TIBCO® Workspace Configuration and Customization

Software Release 4.2
August 2017

Document Update: December 2017

Two-Second Advantage®

Important Information

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

ANY SOFTWARE ITEM IDENTIFIED AS THIRD PARTY LIBRARY IS AVAILABLE UNDER SEPARATE SOFTWARE LICENSE TERMS AND IS NOT PART OF A TIBCO PRODUCT. AS SUCH, THESE SOFTWARE ITEMS ARE NOT COVERED BY THE TERMS OF YOUR AGREEMENT WITH TIBCO, INCLUDING ANY TERMS CONCERNING SUPPORT, MAINTENANCE, WARRANTIES, AND INDEMNITIES. DOWNLOAD AND USE THESE ITEMS IS SOLELY AT YOUR OWN DISCRETION AND SUBJECT TO THE LICENSE TERMS APPLICABLE TO THEM. BY PROCEEDING TO DOWNLOAD, INSTALL OR USE ANY OF THESE ITEMS, YOU ACKNOWLEDGE THE FOREGOING DISTINCTIONS BETWEEN THESE ITEMS AND TIBCO PRODUCTS.

This document contains confidential information that is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of TIBCO Software Inc.

TIBCO, Two-Second Advantage, TIBCO ActiveMatrix BPM, TIBCO Administrator, TIBCO Business Studio, TIBCO Enterprise Message Service, TIBCO General Interface, TIBCO Hawk, TIBCO iProcess, TIBCO JasperReports, TIBCO Spotfire, TIBCO Spotfire Server, and TIBCO Spotfire Web Player are either registered trademarks or trademarks of TIBCO Software Inc. in the United States and/or other countries.

Enterprise Java Beans (EJB), Java Platform Enterprise Edition (Java EE), Java 2 Platform Enterprise Edition (J2EE), and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle Corporation in the U.S. and other countries.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

THIS SOFTWARE MAY BE AVAILABLE ON MULTIPLE OPERATING SYSTEMS. HOWEVER, NOT ALL OPERATING SYSTEM PLATFORMS FOR A SPECIFIC SOFTWARE VERSION ARE RELEASED AT THE SAME TIME. SEE THE README FILE FOR THE AVAILABILITY OF THIS SOFTWARE VERSION ON A SPECIFIC OPERATING SYSTEM PLATFORM.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. TIBCO SOFTWARE INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

Copyright © 2005-2017 TIBCO Software Inc. All rights reserved.

TIBCO Software Inc. Confidential Information

Contents

Figures

TIBCO Documentation and Support Services

How to Access TIBCO Documentation

Documentation for TIBCO products is available on the TIBCO Product Documentation website, mainly in HTML and PDF formats.

The TIBCO Product Documentation website is updated frequently and is more current than any other documentation included with the product. To access the latest documentation, visit https://docs.tibco.com.

Product-Specific Documentation

Documentation for TIBCO products is not bundled with the software. Instead, it is available on the TIBCO Documentation site. To directly access documentation for this product, double-click the following file:

`TIBCO_HOME``/release_notes/TIB_amx-bpm_`*`version`*`_docinfo.html`

where *TIBCO_HOME* is the top-level directory in which TIBCO products are installed. On Windows, the default *TIBCO_HOME* is `C:\tibco`. On UNIX systems, the default *TIBCO_HOME* is `/opt/tibco`.

The following documents for this product can be found on the TIBCO Documentation site:

- TIBCO ActiveMatrix BPM SOA Concepts
- TIBCO ActiveMatrix BPM Concepts
- TIBCO ActiveMatrix BPM Developer's Guide
- TIBCO ActiveMatrix BPM Web Client Developer's Guide
- TIBCO ActiveMatrix BPM Tutorials
- TIBCO ActiveMatrix BPM Business Data Services Developer Guide
- TIBCO ActiveMatrix BPM Case Data User Guide
- TIBCO ActiveMatrix BPM Event Collector Schema Reference
- TIBCO ActiveMatrix BPM - Integration with Content Management Systems
- TIBCO ActiveMatrix BPM SOA Composite Development
- TIBCO ActiveMatrix BPM Java Component Development
- TIBCO ActiveMatrix BPM Mediation Component Development
- TIBCO ActiveMatrix BPM Mediation API Reference
- TIBCO ActiveMatrix BPM WebApp Component Development
- TIBCO ActiveMatrix BPM Administration
- TIBCO ActiveMatrix BPM Performance Tuning Guide
- TIBCO ActiveMatrix BPM SOA Administration
- TIBCO ActiveMatrix BPM SOA Administration Tutorials
- TIBCO ActiveMatrix BPM SOA Development Tutorials
- TIBCO ActiveMatrix BPM Client Application Management Guide
- TIBCO ActiveMatrix BPM Client Application Developer's Guide
- TIBCO Openspace User's Guide
- TIBCO Openspace Customization Guide

- TIBCO ActiveMatrix BPM Organization Browser User's Guide (Openspace)
- TIBCO ActiveMatrix BPM Organization Browser User's Guide (Workspace)
- TIBCO ActiveMatrix BPM Spotfire Visualizations
- TIBCO Workspace User's Guide
- TIBCO Workspace Configuration and Customization
- TIBCO Workspace Components Developer Guide
- TIBCO ActiveMatrix BPM Troubleshooting Guide
- TIBCO ActiveMatrix BPM Deployment
- TIBCO ActiveMatrix BPM Hawk Plug-in User's Guide
- TIBCO ActiveMatrix BPM Installation: Developer Server
- TIBCO ActiveMatrix BPM Installation and Configuration
- TIBCO ActiveMatrix BPM Log Viewer
- TIBCO ActiveMatrix BPM Single Sign-On
- Using TIBCO JasperReports for ActiveMatrix BPM

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

- For an overview of TIBCO Support, visit http://www.tibco.com/services/support.
- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the TIBCO Support portal at https://support.tibco.com.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to https://support.tibco.com. If you do not have a user name, you can request one by clicking Register on the website.

How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the TIBCO Ideas Portal. For a free registration, go to https://community.tibco.com.

Introduction

The information in the *TIBCO® Workspace Configuration and Customization* guide can be used to configure and customize any WCC application—that includes either the *Workspace application* or a *custom WCC application*.

- **Workspace application** - This application is created with, and illustrates the use of, all of the available WCC components. It is installed with:
  - **The BPM Runtime** - BPM runtime software is installed on a server (or “node”), and includes all of the necessary BPM runtime components (for example, Directory Engine, Process Engine, etc.), including the Workspace application.

    The Workspace application that is installed with the BPM runtime is already deployed and can be configured and customized to the extent that deployed applications can be configured/ customized as described in this document.
  - **TIBCO Business Studio** - The Workspace application is also installed in your local development environment when you install Business Studio. This application can be opened with TIBCO General Interface Builder and modified, as desired. After modification, it must be deployed to the node.

    The Workspace application can be configured and customized either prior to deploying it to the node, of after deploying it (to the extent that deployed applications can be configured/ customized).
- **Custom WCC Application** - A custom WCC application that is created with TIBCO General Interface Builder can also be configured and customized either prior to deploying it to the node, of after deploying it (to the extent that deployed applications can be configured/customized).

Methods of Configuration and Customization

The way in which WCC applications are configured and/or customized depends on whether you are configuring/customizing a deployed application or one that is not deployed and is installed in your local development environment.

- **Configuring and Customizing a deployed application** - An application that is already deployed on a BPM runtime machine can be configured and/or customized in one of two ways:
  - Using the **Configuration Administrator** - This is used to configure the deployed application by allowing you to modify configuration settings that are stored in the database on the BPM runtime machine.

    This method of configuration can be used for any WCC application that has been deployed (including the Workspace application).

    For more information, see Using the Configuration Administrator.
  - Directly modifying files on the BPM runtime machine - This method of configuration can only be used for the Workspace application. It cannot be used for a deployed custom WCC application (the specific location in the file system in which the WCC application is deployed is indeterminable—it contains GUIDs determined at deployment time).

    For more information, see Directly Modifying Files on the BPM Runtime Machine.
- **Configuring and customizing a non-deployed application** - Configuring and customizing a non-deployed application is done by making changes to the application in your local development environment, either by directly modifying files, or by using TIBCO General Interface Builder. Then after making the desired changes, the application must be deployed to a node.

  This generally falls into two categories:

- You are creating a new custom application using WCC components.
- You already have the Workspace application deployed, but you want to make changes to it that are beyond the scope of what you can do via configuration with the Configuration Administrator. In this case, you would make changes to the Workspace application that was installed when TIBCO Business Studio was installed. Then after configuring/customizing it, you must deploy it.

  For more information, see Configuring and Customizing a Non-Deployed Application.

Configuring and Customizing a Deployed Application

When a BPM runtime is installed, it automatically includes the Workspace application. You can configure and customize that application. Or you can configure and customize a custom WCC application that was created locally, then deployed to the BPM runtime machine.

This can be done in one of two ways:

- Using the Configuration Administrator
- Directly Modifying Files on the BPM Runtime Machine

Using the Configuration Administrator

You can easily configure certain aspects of a deployed application (either the Workspace application, or a custom WCC application that has been deployed), such as setting default user options, specifying user access to functions in the application, specifying the way in which events are displayed to the user, etc.

These configurations are available via the **Configuration Administrator**. The Configuration Administrator is a WCC component that is available via a toolbar button in the Workspace application.

To access the Configuration Administrator from Workspace, click the **Admin** button in the Workspace toolbar, then select **Configuration**. The Configuration Administrator dialog is displayed.

The Configuration Administrator dialog displays a directory/file structure in the left pane that represents configuration files for the deployed application. Clicking on a file name in the left pane displays the contents of that file in the right pane.

Procedure

1. In the left pane of the Configuration Administrator, click on the file that contains the configuration information you want to change. The following describes the type of configuration information available in each of the configuration files:

   - **config.xml** - This file contains a variety of configuration parameters, such as default user options, session timeout value, etc.

     After selecting the `config.xml` file in the Configuration Administrator, you can choose to edit the configuration parameters using either a graphical editor or an XML editor by clicking on the appropriate tab in the right pane. The use of each of these editors is described in Configuring the Application.

     > The XML view is provided in the Configuration Administrator so that you can easily move XML for one configuration to another (for example, from a test environment to a production environment).

   - **userAccess.xml** - This file is used to specify which functions users can access in the application, based on the privileges held by the user.

After selecting the `userAccess.xml` file in the Configuration Administrator, you can choose to edit user access using either a graphical editor or an XML editor by clicking on the appropriate tab in the right pane. The use of each of these editors is described in Configuring User Access.

- **customInterfaces.xml** - This file is used to configure custom interfaces that allow custom menus and toolbar buttons to be displayed in a WCC application that, when selected, display TIBCO General Interface prototypes, web pages displayed in iFrames, or web pages displayed in TIBCO PageBus Managed Hub iFrames. For more information, see Custom Interfaces.
- **eventRoles.xml** - This file is used to map privileges to *event roles*, which are used to determine how events are displayed to users who hold specific privileges.

  If a role is added using the Configuration Administrator, it causes an entry of the same name to be dynamically added under the `Event Roles` directory in the left pane of the Configuration Administrator. It adds the event configuration files for the newly added role to the database by copying the event configuration files from the default role files that are on the deployed file system.

  This allows you to add roles via configuration (i.e., without requiring a re-deployment).

  You can then configure each of the individual event configuration files for the new role by accessing those files using the Configuration Administrator.

  For more information about configuring events, see Configuring Events.
- **eventLocale.xml** - This file contains text strings that are referenced in the event configuration files (`eventDescriptions.xml`, `eventLinks.xml`, `eventViewTemplates.xml`, and `eventAttributes.xml`). It allows for easier localizing of the event-related text. (Note that if your application has been localized, additional event locale files will appear as well, for example, `eventLocale.es.xml`, `eventLocale.es_MX.xml`, etc. For information about localization, see Localization.)

  For more information about configuring events, see Event-Related Text.
- **eventLocale.en_GB.xml** - This file is a placeholder for localizing event-related text strings for English Great Britain. This file is empty by default, which causes it to fall back to English United States. For more information, see Localization.

  For more information about configuring events, see Event-Related Text.
- **customInterfaceLocale.xml** - This file is used to support localization of text strings associated with custom interfaces. If you localize custom interface-related text strings, this file contains the English (United States) strings, and there will also be a `customInterfaceLocale.`*`ll_CC`*`.xml` resource file that contains the localized text strings for the language identified by the "*`ll_CC`*" in the resource file name. For more information, see Localization for Custom Interfaces
- **customInterfaceLocale.en_GB.xml** - This file is a placeholder for localizing custom interface-related text strings for English (Great Britain). This file is empty by default, which causes it to fall back to English (United States). For more information, see Localization for Custom Interfaces.

  The contents of the selected file are displayed in the right pane.

2. Edit the configuration information as described in the links provided in step 1.

   You can use the standard **Ctrl+C**, **Ctrl+V**, and **Ctrl+A** keys to copy, paste, and select all, respectively.

   > As the text editor in the Configuration Administrator displays all black text, it can be helpful to select all of the text in the right pane (Ctrl+A), copy it, then paste it into an editor that displays XML in colored text. Make your changes in that editor, then copy it and paste it back into the Configuration Administrator.

   If you make a change to a file, an asterisk is shown to the left of the file name in the left pane. This is an indicator that the file has changed and that you have not applied nor saved it yet.

3. After making a change, you can apply changes without closing the Configuration Administrator by clicking the **Apply** button, or you can save changes and close the window by clicking the **OK** button.
4. You must restart the application for changes made via the Configuration Administrator to take effect.

Configuration Administrator Writes Files to the Database

When a deployed application is started, it attempts to read configuration information from the database. If the configuration information is not in the database, it obtains it from the appropriate configuration file in the file system.

This occurs for the following configuration files:

- `config.xml`
- `userAccess.xml`
- `customInterfaces.xml`
- `eventRoles.xml`
- `eventAttributes.xml`
- `eventDescriptions.xml`
- `eventLinks.xml`
- `eventViewTemplates.xml`
- `eventLocale.xml`
- `eventLocale.en_GB.xml`
- `customIntefaceLocale.xml`
- `customIntefaceLocale.en_GB.xml`

The first time the Configuration Administrator is used to modify the information in any of these configuration files, the Configuration Administrator writes that information to the database and modifies the information there. For example, the first time you click on the `config.xml` file in the left pane of the Configuration Administrator, the contents of the `config.xml` file on the file system is written to the database. Changes you make are made in the database—the original configuration file remains unchanged.

After that first time that you modify a configuration file using the Configuration Administrator, it reads the information from the database, not from the file in the file system.

Also, subsequent starts of the application will then read the modified configuration information from the database, rather than from the file on the file system.

For the system to be able to read configuration information from the database for a particular user, the user must have the DE.userAdmin system action. If the user does not have the DE.userAdmin system action, the system *always* reads configuration information from the file system, even if the Configuration Administrator had previously been used, and configuration has been written to the database.

After accessing a configuration file in the Configuration Administrator, causing it to be written to the database, you can revert to the information in the file on the file system by using the **Defaults** button; see Setting Configuration Information to Default Values.

You can also launch a WCC application using the default configuration files, rather than using the files that have been written to the database. For information about doing this, see Launching a WCC Application Using Default Configuration Files.

Setting Configuration Information to Default Values

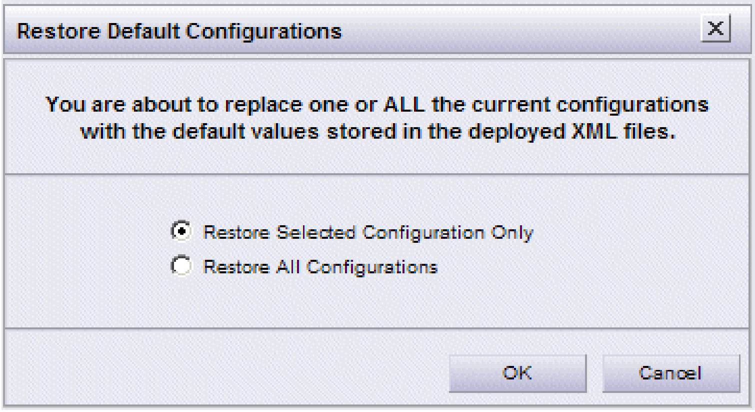
Anytime a change is made to configuration information using the Configuration Administrator, it is written to the database—it does not ever modify the original configuration file on the file system.

The Configuration Administrator also provides a function that allows you to restore the default values for any or all of the configuration files. This causes the values in the original configuration file(s) to be written to the database, overwriting any changes that had been made to those configurations.

Procedure

1. In the Configuration Administrator left pane, select the configuration file whose default values you want to restore (if you want to restore the default values for *all* configuration files, it doesn't matter what is currently selected), then click the **Defaults** button in the lower portion of the left pane.

   The following dialog is displayed:



Restore Default Configurations

You are about to replace one or ALL the current configurations with the default values stored in the deployed XML files.

● Restore Selected Configuration Only
○ Restore All Configurations

2. Select the appropriate option, as follows:
   - **Restore Selected Configuration Only** - Select this to restore only the configuration values from the file that was selected when you clicked the **Default** button.

     Note that this option is disabled if one of the "folders" in the left pane was selected rather than one of the files.
   - **Restore All Configurations** - Select this to restore the default values for all of the configuration files listed in the left pane.

     Note that the way in which default values are restored for the event configuration files depends on whether the event role existed in the deployment.

     For example, if a "LoanManager" event role existed when the application was deployed, event configuration files for that event role will exist in the file system—in this case, the default configuration values will be read from the LoanManager role configuration files.

     If the LoanManager role was added *after* deployment, event configuration files for that role will not exist in the file system. Therefore, if you restore event configuration information for the LoanManager role, the Configuration Administrator will read the event configuration information for the "default" role from the file system, and write that information to the database for the LoanManager role. (The same occurs if files for a deployed role are deleted from the file system—configuration values are read from the files for the default role.)
3. Click **OK** to restore the configuration defaults.

   An asterisk appears to the left of the name of each file that has been modified.
4. Click either **Apply** to save the changes in the database and keep the Configuration Administrator open, or **OK** to save the changes in the database and close the Configuration Administrator.

Directly Modifying Files on the BPM Runtime Machine

You can also configure and customize a deployed Workspace application by directly modifying the files on the BPM runtime machine.

For information about the location of the files you can modify, see Location of Files on a BPM Runtime Machine.

Note, however, that you cannot directly modify the files in a custom WCC application that has been deployed because the specific location in the file system in which the WCC application is deployed is indeterminable—it contains GUIDs determined at deployment time. To configure or customize a custom WCC application beyond the scope of what can be done with the Configuration Administrator, you can make the desired changes in your local WCC development environment, then re-deploy the application.

Using this method, you can perform the following types of configurations and customizations:

- **Configuration changes** - You can make changes directly to configuration files on the BPM runtime file system (as opposed to using the Configuration Administrator to make database changes for deployed applications). The configuration files that can be modified are:

  – **config.xml** - This file contains a variety of configuration parameters, such as default user options, specifying the session timeout value, etc. All of the configuration parameters available in the `config.xml` file are described in Configuring the Application.

  – **userAccess.xml** - This file is used to specify which functions users can access in the application, based on the privileges held by the user. For information about configuring user access, see Configuring User Access.

  – **eventRoles.xml** - This file is used to map privileges to event roles; event roles are used to configure events differently based on a user's privileges. For more information, see Mapping Privileges to Event Roles.

    Note - There are also some configuration files in the `\JSXAPPS\base\locale\` directory that specify how events are displayed in the application. For information about these configuration files, see Configuring Events.

  – **customInterfaces.xml** - This file is used to configure custom interfaces that allow custom menus and toolbar buttons to be displayed in a WCC application that, when selected, display TIBCO General Interface prototypes, web pages displayed in iFrames, or web pages displayed in TIBCO PageBus Managed Hub iFrames. For more information, see Custom Interfaces.

  – **wccConfig.xml** - This file is used to specify things such as whether or not to load previously loaded resources when loading the application in General Interface Builder. For more information, see WCC Configuration File.

  If you are manually modifying files on the BPM runtime machine, and they are files that are accessible from the Configuration Administrator (such as `config.xml` and `userAccess.xml`), you must have an understanding of which files the application is going to use—those in the file system, or those in the database. For information about this, see Configuration Administrator Writes Files to the Database .

- **Code-level changes** - Custom code can be added to perform whatever business logic is required. This requires that you open and modify files on the BPM runtime machine to implement the custom logic desired. Note that code changes should *not* be made to the `...\JSXAPPS\base\base.js` file, as this is TIBCO's private implementation of the WCC components—it is crunched and obfuscated.

  For information about custom code entry points, see the "Building Custom Applications" chapter in the *TIBCO Workspace Components Developer Guide*.

- Other Customizations - There are other various customizations you can perform, such as changing fonts, colors, and images in the application, localizing the application, adding custom menus and toolbar buttons, etc.

For information about these customizations, see Customizations.

Prior to making changes directly in the files on the BPM runtime machine, you should ensure that all users have logged out of all TIBCO BPM applications, then stop the BPM node. After making changes and restarting the BPM node, you should also clear the browser cache before logging back into TIBCO BPM applications.

Location of Files on a BPM Runtime Machine

This topic provides information about the location of files on the BPM runtime machine. These paths can be used when you choose to directly modify the files for the Workspace application.

Note that you cannot directly modify the files in a custom WCC application that has been deployed because the specific location in the file system in which the WCC application is deployed is indeterminable—it contains GUIDs determined at deployment time.

There are actually two locations in which the BPM files (including Workspace files) are installed. They are:

- **TIBCO Configuration Home** - This is where the BPM runtime files are stored. Making changes to Workspace files in this path affects the current runtime system (this is shown in Windows format—it is the same in UNIX, except for the direction of the slashes):

  ```
  %config_home%\tibcohost\Admin-environmentName-adminServer\data_3.2.x\host\plugins
  \com.tibco.n2.rtc.ws_Version\resources\
  ```

  where:

  - *%config_home%* is the root directory for BPM runtime files. This can be specified during the installation. On Windows systems, this defaults to: `C:\ProgramData\amx-bpm\tibco\data`; on UNIX systems, this defaults to: `/opt/tibco/data`.
  - environmentName - The name of the BPM environment. This can be specified during the installation.
  - adminServer - The name of the TIBCO Administrator server. This can be specified during the installation.
  - *Version* is the specific version of the Workspace software installed.

  The `\ProgramData` directory is hidden by default on Windows Server 2008.

- **TIBCO Installation Home** - This is the *shared plug-in component store*. The files in this location are used when a new node is created with the TIBCO Configuration Tool (TCT). Making changes to Workspace files in this path affects only new nodes that are created in the future using the TCT (this is shown in Windows format—it is the same in UNIX, except for the direction of the slashes):

  ```
  %tibco_home%\components\shared\1.0.0\plugins\com.tibco.n2.rtc.
  ws_Version\resources
  ```

  where:

  - *%tibco_home%* is the root directory for installation home. This can be specified during the installation. On Windows systems, this defaults to: `C:\tibco\amx-bpm`; on UNIX, this defaults to `/opt/tibco`.
  - *Version* is the specific version of the Workspace software installed.

Configuring and Customizing a Non-Deployed Application

This topic describes how to configure and customize a non-deployed WCC application.

A non-deployed application is simply one in which the application source is installed on your local development environment. There are two primary reasons you would have the source installed in your local development environment:

- You are creating a new custom application using WCC components. This allows you to design the application to exactly fit your needs. For information about creating a new custom WCC application, including reference information about all of the available components, see the *TIBCO Workspace Components Developer Guide*.
- You already have the Workspace application deployed, but you want to make changes to it that are beyond the scope of what you can do via configuration with the Configuration Administrator. This requires that you install the Workspace application locally, make the desired changes, then re-deploy it. The details of how to do this are provided below.

To accomplish either one of the scenarios described above, you must install TIBCO Business Studio locally. Business Studio provides the tools needed to create processes and organization models used in TIBCO applications. The Business Studio installation also includes the following software, which are used to create and/or customize WCC applications:

- **TIBCO General Interface Builder** - This is the development tool you will use to open and modify WCC applications.
- **WCC components** - These are used as the building blocks for WCC custom applications. The WCC components are added to custom applications using General Interface Builder.
- **The Workspace application/framework** - The entire Workspace application and framework are also installed when you install TIBCO Business Studio.

The following are the types of customization tasks you can perform on your WCC application:

- **Layout / component changes** - These kinds of changes are made by opening your application in General Interface Builder, then changing the layout or functionality using the standard General Interface Builder components or WCC components.

  If you are creating a new custom application, see the *TIBCO Workspace Components Developer Guide* for information about how to create the application and add WCC components to it.

  If you are customizing the Workspace application, see Opening the Workspace Application In General Interface Builder for information about how to open that application in General Interface Builder.
- **Configuration changes** - For non-deployed applications, configuration changes are accomplished by making changes directly to configuration files in your local development environment.

  For information about making configuration changes to the local configuration files, see Configuring a Non-Deployed Application.
- **Code-level changes** - Custom code can be added to perform whatever business logic is required. This requires that you open and modify files locally to implement the custom logic desired. Note that code changes should *not* be made to the `...\JSXAPPS\base\base.js` file, as this is TIBCO's private implementation of the WCC components—it is crunched and obfuscated.

  For information about custom code entry points, see the "Building Custom Applications" chapter in the *TIBCO Workspace Components Developer Guide*.
- Other Customizations - There are other various customizations you can perform, such as changing fonts, colors, and images in the application, localizing the application, adding custom menus and toolbar buttons, etc.

  For information about these customizations, see Customizations.

Opening the Workspace Application In General Interface Builder

This topic describes how to open the Workspace application in TIBCO General Interface Builder so that layout and component changes can be made.

For information about customizing the application using components, see the *TIBCO Workspace Components Developer Guide*.

Procedure

1. Install TIBCO Business Studio. Ensure that the "Workspace Client Components" feature is selected and installed. (You must use the version of TIBCO Business Studio that targets the version of TIBCO ActiveMatrix BPM you are using.)
2. Start General Interface Builder and create a workspace if you haven't created one yet (the General Interface workspace must be set to the directory that contains the `GI_Builder.html` file, as shown below).

   General Interface Builder can be started by executing the following:

   ```
   StudioHome\wcc\version\GI_Builder.html
   ```

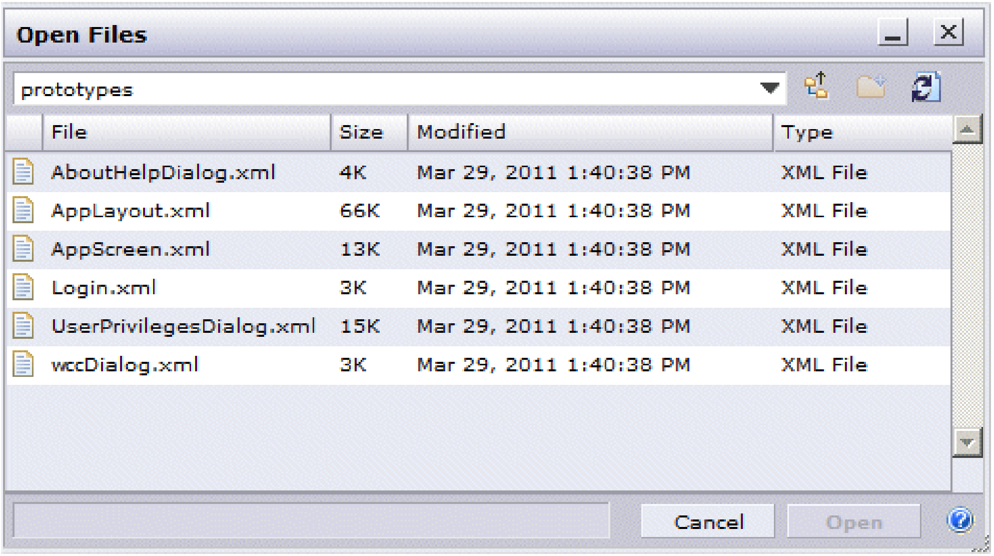
   where:

   - *StudioHome* is the directory in which TIBCO Business Studio was installed.
   - *version* is the version number of Workspace that was installed with TIBCO Business Studio.

     For more information about starting General Interface Builder and creating workspaces, see the "Starting TIBCO General Interface Builder" section in *TIBCO General Interface Getting Started.*
3. From the **Project** menu in General Interface Builder, select **User Projects** > **workspace** .

   This opens the General Interface project in which the Workspace application was built.
4. From the **File** menu, select **Open**.
5. From the **Open Files** dialog, drill down to **workspace** > **application** > **prototypes** .

   The Open Files dialog will now list the available prototypes in the Workspace application:



File	Size	Modified	Type
AboutHelpDialog.xml	4K	Mar 29, 2011 1:40:38 PM	XML File
AppLayout.xml	66K	Mar 29, 2011 1:40:38 PM	XML File
AppScreen.xml	13K	Mar 29, 2011 1:40:38 PM	XML File
Login.xml	3K	Mar 29, 2011 1:40:38 PM	XML File
UserPrivilegesDialog.xml	15K	Mar 29, 2011 1:40:38 PM	XML File
wccDialog.xml	3K	Mar 29, 2011 1:40:38 PM	XML File

6. Open the appropriate prototype to customize the desired part of the Workspace application. For instance, opening `AppLayout.xml`, allows you to customize the main layout of the application, opening `Login.xml` allows you to customize the Login dialog, etc.

Configuring a Non-Deployed Application

For applications in your WCC development environment (such as the Workspace application or a custom WCC application), you can make configuration changes directly in the local configuration files.

The following provides the location of each of the configuration files available:

- **config.xml** - This file contains a variety of configuration parameters, such as default user options, specifying the session timeout value, etc. All of the configuration parameters available in the `config.xml` file are described in Configuring the Application.
- **userAccess.xml** - This file is used to specify which functions users can access in the application, based on the privileges held by the user. For information about configuring user access, see Configuring User Access.
- **eventRoles.xml** - This file is used to map privileges to event roles; event roles are used to configure events differently based on a user's privileges. For more information, see Mapping Privileges to Event Roles.

  Note - There are also some configuration files in the `\JSXAPPS\base\locale\` directory that specify how events are displayed in the application. For information about these configuration files, see Configuring Events.
- **customInterfaces.xml** - This file is used to configure custom interfaces that allow custom menus and toolbar buttons to be displayed in a WCC application that, when selected, display TIBCO General Interface prototypes, web pages displayed in iFrames, or web pages displayed in TIBCO PageBus Managed Hub iFrames. For more information, see Custom Interfaces.
- **wccConfig.xml** - This file is used to specify things such as whether or not to load previously loaded resources when loading the application in General Interface Builder. For more information, see WCC Configuration File.

All of these configuration files are located in the following directory:

`StudioHome\wcc\version\JSXAPPS\WCCProjectName\`

where:

- *`StudioHome`* is the directory in which TIBCO Business Studio was installed.
- *version* is the version number of Workspace that was installed with TIBCO Business Studio.
- *WCCProjectName* is the name of the General Interface Builder project that contains your custom application. If you are working with the Workspace application, this is “workspace”.

WCC Configuration File

There is a WCC configuration file, `wccConfig.xml`, that is created for each WCC project/application.

Note that every WCC application has its own `wccConfig.xml` file, whereas multiple WCC applications can reference the same `config.xml` file (see Multiple Applications Referencing a Single Configuration File).

The WCC configuration file contains parameters that specify things such as whether or not to load previously loaded resources when loading the application in General Interface Builder. This file is located as follows:

`StudioHome\wcc\version\JSXAPPS\WCCProjectName\wccConfig.xml`

where:

- *`StudioHome`* is the directory in which TIBCO Business Studio was installed.
- *version* is the version number of Workspace that was installed with TIBCO Business Studio.
- *WCCProjectName* is the name of the General Interface Builder project that contains your custom application. If you are working with the Workspace application, this is “workspace”.

> The **wccapppath** attribute in the application's launch fragment (see WCC Application Launch Fragment) points to the directory containing the `wccConfig.xml` file. The `wccConfig.xml` file is the only file that is required in the directory specified by **wccapppath**.

For further information, see the comments in the `wccConfig.xml` file.

WCC Application Launch Fragment

When a WCC application is created in General Interface Builder, a launch fragment is created that is used to launch your custom application.

This launch fragment is located as follows:

`StudioHome\wcc\version\WCCProjectName.html`

where:

- `StudioHome` is the directory in which TIBCO Business Studio was installed.
- *version* is the version number of Workspace that was installed with TIBCO Business Studio.
- *WCCProjectName* is the name of the General Interface Builder project that contains your custom application. If you are working with the Workspace application, this is “workspace”.

For example, if you create a custom application called “Accounts”, an `Accounts.html` file is created in your workspace directory. Executing this file starts the custom application.

For more information about launching WCC applications, see Launching a WCC Application.

Deploying an Application After Customizing

After customizing and testing a non-deployed application in your development environment, you can deploy the application to a node using a provided WAR-file creation utility.

The WAR-file command script used in the steps below requires that you have a Java Development Kit (JDK) installed on your system, plus the JAVA_HOME environment variable must point to the directory in which the JDK is installed.

Procedure

1. Execute the following command script:

   `StudioHome\wcc\version\WCCProjectName.create.war.cmd`

   where:

   - `StudioHome` is the directory in which TIBCO Business Studio was installed.
   - *version* is the version number of Workspace that was installed with TIBCO Business Studio.
   - *WCCProjectName* is the name of the General Interface Builder project that contains your custom application. If you are working with the Workspace application, this is “workspace”.

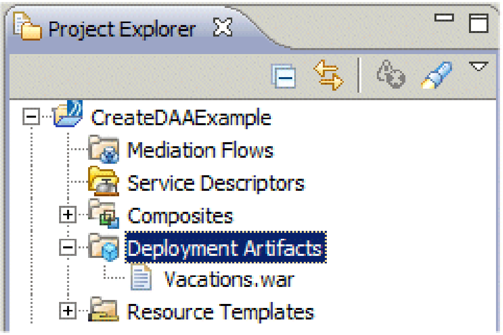
     This command file was created when you saved your custom WCC project in TIBCO General Interface Builder.

     A `WCCProjectName.war` file is created in the same directory containing the command file. This WAR file, which contains the entire custom application, including the Workspace framework, will be used to create a deployment DAA file.

     Note that this can take a few minutes to complete.
2. Start TIBCO Business Studio.
3. From the **File** menu, select **New** > **TIBCO SOA Resources** .
4. From the Select a wizard dialog, select "TIBCO SOA Project", then click **Next**.

   Note - Throughout these steps, "CreateDAAExample" is used as the project name; you can substitute this with the desired name, where applicable.
5. In the **Project Name** field, enter "CreateDAAExample", then click **Next**.

6. Accept the defaults on the **Asset Type Selection** dialog, and click **Next**.
7. On the Composite Project dialog, select "Empty SOA Project", then click **Next**.
8. On the Composite File Name dialog, the **Composite File** field will be prefilled with project name ("CreateDAAExample")—click **Next**.
9. On the Set Special Folders dialog, click **Finish**.
10. Copy the WAR file you created in step 1 into Business Studio by doing the following:
    a) From your local file system, select the WAR file and copy it to the clipboard.
    b) From the Project Explorer in Business Studio, right-click **Deployment Artifacts** and select **Paste**. The WAR file is added to the artifacts:



11. Select the WAR file under **Deployment Artifacts** and drag and drop it into the composite canvas in the right pane.
12. Select the component in the composite, then set the following properties on the **Properties** tab in the **Properties** view:

- **contextRoot**: The value you enter in this property is the root of the URL used to launch the application after deployment; it is the part of the URL between the host name / port number, and the launch fragment. For example, the contextRoot in the following URL is "inventory":

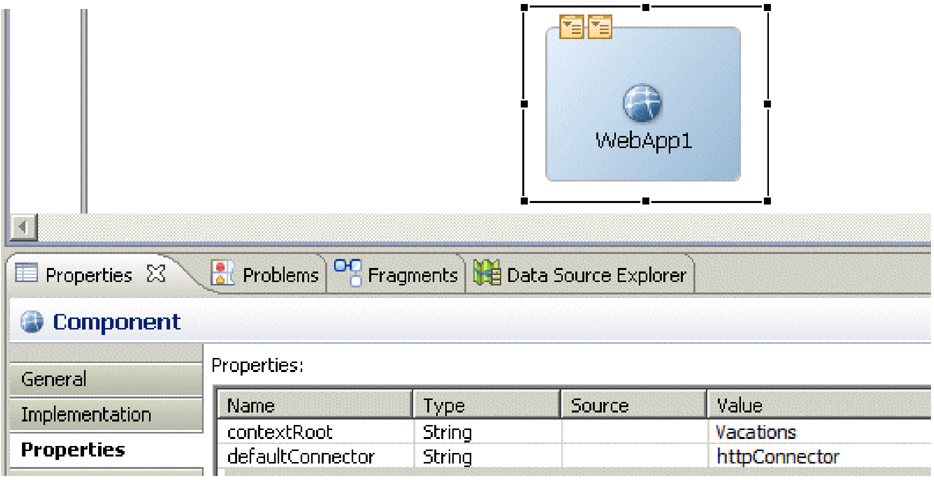
  ```
  http(s)://eastern:8088/inventory/launch.html
  ```

  The contextRoot you enter in the **contextRoot** property must be a simple string (it cannot contain slashes).

  For more information, see Launching a WCC Application.

- **defaultConnector**: The defaultConnector must be the same HTTP connector used by the BPM runtime (which can be seen in TIBCO Administrator; the default is "httpConnector").

  For example:



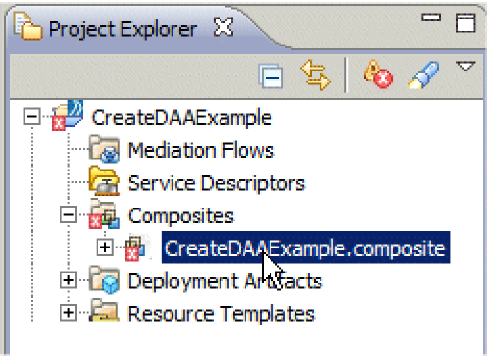
Properties:

Name	Type	Source	Value
contextRoot	String		Vacations
defaultConnector	String		httpConnector

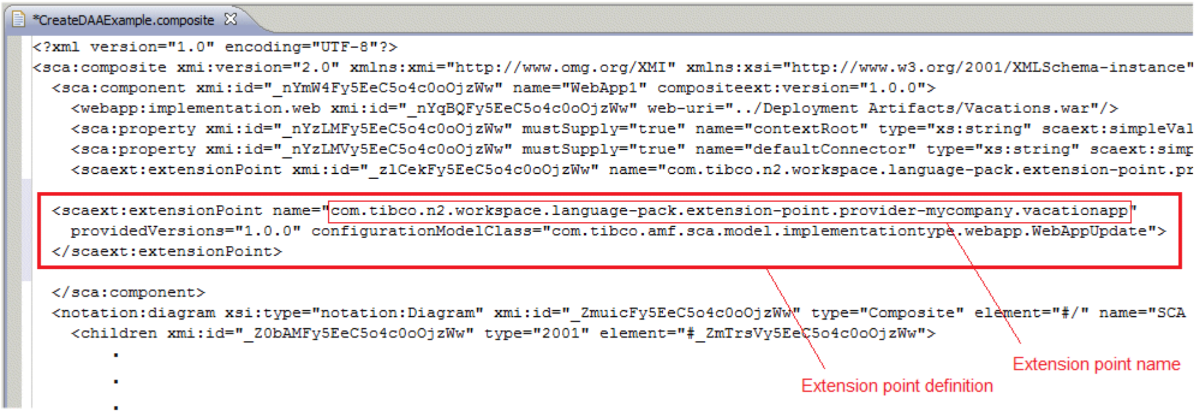
13. Save your project.
14. [Optional step] This step describes how to add an extension point to the composite. This extension point can be used at a later time to add *resource files*, that have been manually localized, to your deployed application. Without an extension point, which allows dynamic update of files in a deployed application, you would need to undeploy the application, add the resource files, then re-deploy the application (resulting in application downtime). For more information, see Localization.

    To add an extension point:

    a) In the **Project Explorer**, right click the composite, then select **Open With** > **Text Editor** from the context menu:



    b) Add an extension point definition to the composite file, inside of the <sca:Componentelement> element, as follows:



```
<?xml version="1.0" encoding="UTF-8"?>
<sca:composite xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <sca:component xmi:id="_nYmW4Fy5EeC5o4c0oOjzWw" name="WebApp1" compositeext:version="1.0.0">
    <webapp:implementation.web xmi:id="_nYqBQFy5EeC5o4c0oOjzWw" web-uri="../Deployment Artifacts/Vacations.war"/>
    <sca:property xmi:id="_nYzLMFy5EeC5o4c0oOjzWw" mustSupply="true" name="contextRoot" type="xs:string" scaext:simpleVal
    <sca:property xmi:id="_nYzLMVy5EeC5o4c0oOjzWw" mustSupply="true" name="defaultConnector" type="xs:string" scaext:simp
    <scaext:extensionPoint xmi:id="_zlCekFy5EeC5o4c0oOjzWw" name="com.tibco.n2.workspace.language-pack.extension-point.pr

  <scaext:extensionPoint name="com.tibco.n2.workspace.language-pack.extension-point.provider-mycompany.vacationapp"
    providedVersions="1.0.0" configurationModelClass="com.tibco.amf.sca.model.implementationtype.webapp.WebAppUpdate">
  </scaext:extensionPoint>

  </sca:component>
  <notation:diagram xsi:type="notation:Diagram" xmi:id="_ZmuicFy5EeC5o4c0oOjzWw" type="Composite" element="#/" name="SCA
    <children xmi:id="_Z0bAMFy5EeC5o4c0oOjzWw" type="2001" element="#_ZmTrsVy5EeC5o4c0oOjzWw">
        .
        .
        .
```

Extension point name

Extension point definition

    Use the following text for the extension point definition, substituting the desired extension point name in the **name** attribute:

```
<scaext:extensionPoint name="
                              com.tibco.n2.workspace.language-pack.extension-
point.provider-<customerId
                               > .<appId
                           " providedVersions="1.0.0"
configurationModelClass="com.tibco.amf.sca.model.implementationtype.webapp.WebA
ppUpdate"> </scaext:extensionPoint>
```

    The recommended convention for the extension point name is to use the value shown in the text above, substituting a customer ID and an application ID in the <**customerId**> and <**appId**> parts of the value. For example:

```
com.tibco.n2.workspace.language-pack.extension-point.provider-
megacorporation.loanapp
```

    Note, however, any value in the **name** attribute is acceptable. Whatever is specified as the name of the extension point must be used when referencing that extension point to dynamically add resource files to the application at a later time. For information about adding resource files using an extension point, see Deploying Language Resource Files.

Also note that multiple WCC applications can define the same extension point. Then if resource files are deployed at a later time to that extension point (as described in Deploying Language Resource Files), all applications that defined that extension point are updated with those resource files.

If you have multiple applications and you expect them to use different translations, you should define a different extension point for each application.

15. Save your project.
16. In the Project Explorer, expand **Composites**, then right-click **CreateDAAExample.composite** and select **Create DAA...**.
17. On the Select Archive Location dialog, accept the default location (CreateDAAExample/ Deployment Artifacts) and the default file name (CreateDAAExample.daa)—click **Next**.
18. On the Select Distribution dialog, ensure the **Do not use a distribution file** option is selected (the default), then click **Next**.
19. On the DAA Specification dialog, ensure the **Save DAA Spec** option is selected (the default), then click **Finish**.

    The DAA file can now be found on your file system in the following directory:

    ```
    workspaceName\CreateDAAExample\Deployment Artifacts
    ```

    where *workspaceName* is the workspace name you specified when starting Business Studio.
20. Using TIBCO Administrator, deploy the DAA file to the appropriate node.

    For information about deploying applications using TIBCO Administrator, see the *TIBCO Administration* guide. (For a tutorial of deploying applications using TIBCO Administrator, see *How to Implement and Deploy the WelcomeUsers Application*.)

Result

After the application is deployed, you can run it using the description in Launching a WCC Application.

Disabling a Currently Deployed Application

You can disable a currently deployed application. The method you use to disable the application depends on whether you are disabling it on a temporary basis (for example, to prevent access during maintenance), or for a longer period of time.

- **Short-term** - To disable an application short-term, use the Configuration Administrator.

  When an application is disabled via the Configuration Administrator, and a user attempts to access it, the **Login** screen is displayed. When the user logs in, configuration data is accessed from the database, then the user is notified that the application is disabled (and the browser is closed if the browser security allows it), or they are redirected to another URL (redirection is described below).
- **Long-term** - To disable an application for a longer term, directly update the `config.xml` file on the BPM runtime machine, rather than via the Configuration Administrator.

  When an application is disabled via an updated `config.xml` file on the BPM runtime machine, the user is immediately notified that the application is disabled (and the browser is closed if the browser security allows it), or they are redirected to another URL (redirection is described below). When the application is disabled in this way, the **Login** screen is not displayed.

Both of the methods described above can be used to disable the Workspace application. However, a custom WCC application can only be disabled by using the Configuration Administrator, as the specific location in the file system in which the WCC application is deployed is indeterminable—it contains GUIDs determined at deployment time. To disable a custom WCC application on a long-term or permanent basis, you would typically stop or undeploy it using TIBCO Administrator. For information about stopping and undeploying applications, see the TIBCO Administrator documentation.

Also note that an application that is disabled can still be accessed by the System Administrator. For information, see Accessing a Disabled Application by the System Administrator.

Redirecting to a URL

When you disable an application, you can also specify that if a user attempts to access the application that they be redirected to a specific URL. Specifying the URL to which they are to be redirected is described in the steps below that describe disabling an application.

Note that if the redirection URL is set to an empty string, instead of redirecting the user, the message specified in the **txtAppDisabled** variable is displayed (this variable is located in the `...\JSXAPPS\base\locale\locale.xml` file). By default, the **txtAppDisabled** variable contains the message “This application is disabled”, but can be changed and/or localized if desired.

Accessing a Disabled Application by the System Administrator

The System Administrator user (out-of-the-box, the System Administrator user’s name and password are “tibco-admin” and “secret”, respectively, but may have been changed) can log into a disabled application by appending the application URL with “?adminBypass=true”. For example:

```
http://hostName:port/workspace/workspace.html?adminBypass=true
```

If accessed using this URL, the **Login** screen is displayed, allowing the user to log in. If a user other than the System Administrator attempts to log in, a message is displayed stating that the application is disabled, then the application reloads and the **Login** screen is re-displayed.

Disabling an Application Using the Configuration Administrator

When an application is disabled via the Configuration Administrator, and a user attempts to access it, the **Login** screen is displayed.

Procedure

1. Log into the deployed application as a user who can access the Configuration Administrator.
2. Open the Configuration Administrator (**Admin** > **Configuration**).
3. From the **Graphical Editor** tab, click **Application**. (Note that these steps describe using the Graphical Editor. You can also use the **XML Editor** in the Configuration Administrator, which is similar to directly updating the `config.xml` file.)
4. Check the **Disable Application** check box.
5. Optionally specify a redirection URL in the **Redirected URL** field. Users who attempt to access the disabled application will be redirected to this URL. For more information, see Redirecting to a URL.
6. Click **OK** to save your changes.
7. Log out of the application.

Disabling an Application by Modifying the config.xml File

When an application is disabled via an updated `config.xml` file on the BPM runtime machine, the user is immediately notified that the application is disabled (and the browser is closed if the browser security allows it), or they are redirected to another URL (redirection is described below).

Procedure

1. Using a text editor, open the `config.xml` file on the BPM runtime machine. The `config.xml` file is located as follows:

   ```
   %config_home%\tibcohost\Admin-environmentName-adminServer\data_3.2.x\host\plugins
   \com.tibco.n2.rtc.ws_Version\resources\JSXAPPS\workspace\config.xml
   ```

   Note - This method can be used only to disable the Workspace application, not a custom WCC application.

2. Locate the **jsxid="Application"** record.
3. Set the **disabled** attribute to "true".
4. Optionally specify a redirection URL in the **redirectUrl** attribute. Users who attempt to access the disabled application will be redirected to this URL. For more information, see Redirecting to a URL.
5. Save and close the `config.xml` file.

Launching a WCC Application

The way in which you launch a WCC application (including the Workspace application) depends on whether the application is deployed to the node, or it is non-deployed (i.e., it is installed in your development environment).

Launching a Deployed Application

A WCC application (either the Workspace application, or a custom application) that is deployed is launched by pointing a browser at the application’s URL.

For example:

```
http://Host:Port/contextRoot/customApp.html
```

where:

- *Host* is the name or IP address of the machine hosting the BPM runtime.
- *Port* is the port number used by the ActiveMatrix WebApp Implementation Type to communicate with web applications.
- *contextRoot* is the root of the deployed application; it is the part of the URL between the host name / port number, and the launch fragment (application name with an html extension).

  If you are launching the Workspace application, the contextRoot is “workspace” by default. If you are launching a custom WCC application, this is the value that was entered in the **contextRoot** field when deploying the application—this is described in Deploying an Application After Customizing (step 12).
- *customApp* is the name of the application. If you are launching the Workspace application, this is “workspace”. If you are launching a custom WCC application, this is the name that was given to the application when it was created in TIBCO General Interface Builder.

Launching a WCC Application Using an External Login

If there is a user already authenticated, you can launch a WCC application in the context of that HTTP session by including “?externalLogin=true” in the URL.

For example:

```
http://Austin:8080/accounts/accounts.html?externalLogin=true
```

The application is initialized with the user name for the authenticated HTTP session.

An example of doing this is provided in the **wccLoginManagedHub** sample application—see the “Using WCC Components in Mash-ups” chapter in the *TIBCO Workspace Components Developer Guide*.

Note that you can also include “?useLDAP=true” in the application URL to specify that the user be authenticated via LDAP. The server needs to be configured for LDAP for this parameter to have an effect. If it is included in the URL, and the server is not configured for LDAP, the parameter is ignored.

Also see the **authenticationMode** configuration parameter, which works in conjunction with the URL overrides described above — see Authentication Mode.

Launching a WCC Application Using Default Configuration Files

A WCC application can be launched using parameters from the default configuration files, rather than the configuration files that have been written to the database. This can be used if data persisted for configuration is somehow corrupted, causing problems loading the application.

You must be the System Administrator (the tibco-admin user in the out-of-the-box system) to use this feature.

To launch the application using the default configuration files, include “`?config=default`” in the URL. For example:

```
http://Austin:8080/accounts/accounts.html?config=default
```

The following default configuration files are loaded:

- `config.xml`
- `userAccess.xml`
- `eventRoles.xml`
- `eventLocale.xml` (and all locale-specific versions)
- For each eventRole defined in eventRoles:
  - `eventsAttributes.xml`
  - `eventsDescriptions.xml`
  - `eventsLinks.xml`
  - `eventsViewTemplates.xml`

Once logged in, the Configuration Administrator can be used to edit the persisted configurations—for information about the Configuration Administrator, see the Using the Configuration Administrator.

Launching the Application in an HTML Frame

To be able to launch the Workspace application, or a custom WCC application, in an HTML frame (for example, an iframe in a portal), you must make some modifications to the launch fragment for the application.

Perform the following procedure prior to launching the application in a frame.

1. Open the launch fragment.
2. Locate the following: “NOTE: To allow display of this application under frames remove the following style and script elements”.
3. Remove (or comment out) the <style> and <script> elements immediately after the note. For example:

```
<!-- NOTE: To allow display of this application under frames remove the following
style and script elements
```

```
<style type="text/css">html{display:none;}</style>
<script language="javascript">
   if (self == top) {
        // Not in frame so show client app
        document.documentElement.style.display='block';
   } else {
        // In a frame so try to show client app outside of a frame
        top.location = self.location;
   }
</script> -->
```

4. Locate the following: “NOTE: To allow display of this application under frames remove the next script element”.
5. Remove (or comment out) the <script> element immediately after the note. For example:

```
<!-- NOTE: To allow display of this application under frames remove the next
script element
<script language="javascript">
   if (self !== top) {
      // Still in a frame so clear body of app and close
      document.getElementsByTagName("body")[0].innerHTML = 'Not allowed in
frames.';
       window.open('close.html', '_self');
   }
</script> -->
```

6. Save and close the launch fragment.

Launching a Non-Deployed Application

When a WCC application is created in General Interface Builder, a launch fragment is created that is used to launch your custom application.

This launch fragment is located as follows:

`*StudioHome*\wcc\*version*\*WCCProjectName*.html`

where:

- `StudioHome` is the directory in which TIBCO Business Studio was installed.
- *version* is the version number of Workspace that was installed with TIBCO Business Studio.
- *WCCProjectName* is the name of the General Interface Builder project that contains your custom application. If you are working with the default Workspace application, this is “workspace”.

For example, if you create a custom application called “Accounts”, an `Accounts.html` file is created in your workspace directory. Executing this file starts the custom application.

Starting Multiple Applications from the Launch Fragment

You can modify the launch fragment to specify that multiple applications load when the file is executed. It allows you to specify a separate <div> element for each application so that each one is displayed in its own area on the screen.

The following provides an example of how you could create two WCC applications, launch them both from a single launch fragment, and have them interact by one application subscribing to the events of the other.

This somewhat simulates a portal environment, except that this example loads two applications within one container, rather than each application being launched within a separate portal. However, you can use this information to help you set up your portal environment.

Assumptions: We will create two applications: **Accounts** and **AccountDetail**. **Accounts** will display a Login dialog; it will then display the work views list when the user logs in. **AccountDetail** will subscribe to the single-click event of the **WorkViews** component in the **Accounts** application, causing the work item list to display in a separate area in the container when the user clicks on a work view in the work views list displayed by **Accounts**.

Procedure

1. Create and save **Accounts**, which should contain two components: **Login** and **WorkViews**. The **WorkViews** component needs to subscribe to the **LoginComplete** event on the **Login** component.

   For information about how to create a custom application using components, see the *TIBCO Workspace Components Developer Guide*, or the *How to Create a BPM Desktop Using Components* tutorial.

2. With **Accounts** open in General Interface Builder, create an event definition file for **Accounts** that **AccountDetail** can import so that it can subscribe to events published by components in **Accounts**.

   For information about creating an event definition file, see the "Creating the External Application Event Definition" section in the *TIBCO Workspace Components Developer Guide.*

3. Create the **AccountDetail** application, which should contain one component: **WorkItems**.

4. With **AccountDetail** open in General Interface Builder, display the Events Editor.

   Note that at this point, there are no events to which the **WorkItems** component can subscribe because it is the only component in the application.

5. In the Events Editor, click the **Import** button.

6. On the **Import Files** dialog, navigate to and select the event definition file that you created in step 2, then click **Import**.

   Event definition files are saved in the `\defs` folder under the application from which you created it, with the name *`WCCProjectName`*`.app.pub.xml`. For example:

   `StudioHome``\wcc\`*`version`*`\JSXAPPS\Accounts\defs\Accounts.app.pub.xml`

   where:

   - *`StudioHome`* is the directory in which TIBCO Business Studio was installed.
   - *version* is the version number of Workspace that was installed with TIBCO Business Studio.

     The Events Editor should now include all of the events from **Accounts**.

7. Subscribe to the **List Item Select (single click)** event on the **WorkViews** component.

8. Save **AccountDetail**.

   At this point, you have created two separate applications, and they each have their own launch fragment that starts the individual application. You will now create a launch fragment that will start both applications and display each in its own region on the screen.

9. Make a copy of the launch fragment created for **Accounts** and save it with whatever name you desire. (The launch fragment for the **Accounts** application is saved in the *`StudioHome`*`\wcc\`*`version`*`\` directory, as file `Accounts.html`.)

10. Modify the new launch fragment as follows:

    a) Change the value of **window.wccAppCount** to 2, which is the number of applications that will be loaded by this launch fragment.

    b) Uncomment the <div> that is located near the bottom of the file.

    c) In the <div> that you uncommented, replace the application name with the name of your second application (**AccountDetail** in this example). It should now appear as follows:

    ```
    <div id="AccountDetail" ...
       <script type="text/javascript" src="JSX/js/JSX30.js"
            jsxappns="AccountDetail" jsxapppath="JSXAPPS/AccountDetail"
            wccapppath="JSXAPPS/AccountDetail" wccloadorder="2" >
       </script>
    <div>
    ```

    Now you need to specify how much of the screen each of the applications will consume—currently, the **Accounts** application is set to use 100% of the screen.

d) Copy the **style** attribute from the <div> element that pertains to the first application, then paste it into the <div> element that pertains to the second application, replacing the **“...”** placeholder.

e) In the <div> element for the first application, change the **height** value in the **style** attribute to 30%. This causes the first application to consume 30% of the container.

f) In the <div> element for the second application, change the **top** value in the **style** attribute to 30%, and the **height** value in the **style** attribute to 70%. This causes the second application to start at 30% from the top, and consume 70% of the container.

The new launch fragment should now look as follows (with comments removed for brevity):

```
<html>
<head>
    <title>TIBCO(R) General Interface</title>
    <script type="text/javascript">
        window.wccAppCount = 2;
    </script>
</head>
<body BGCOLOR="#9898a5" SCROLL="no"
    style="position:absolute;width:100%;height:100%;left:0px;top:0px;padding:
0px;margin:0px;border:0px;overflow:hidden;">
<div id="jsxmain" style="position:absolute;left:0px;top:0px;width:100%;height:
100%;">
    <div id="Accounts" style="position:absolute;left:0px;top:0px;width:
100%;height:30%;">
        <script type="text/javascript" src="JSX/js/JSX30.js"
            jsxappns="Accounts" jsxapppath="JSXAPPS/Accounts"
            wccapppath="JSXAPPS/Accounts" wccloadorder="1" >
        </script>
    </div>
    <div id="AccountDetail" style="position:absolute;left:0px;top:30%;width:
100%;height:70%;">
        <script type="text/javascript" src="JSX/js/JSX30.js"
            jsxappns="AccountDetail" jsxapppath="JSXAPPS/AccountDetail"
            wccapppath="JSXAPPS/AccountDetail" wccloadorder="2" >
        </script>
    <div>
</div>
</body>
</html>
```

11. Save the new launch fragment, then execute it.

    The Login dialog is displayed. When a valid user name and password are entered, the work queue list is displayed in the top 30% of the screen. When a work queue is selected, the work item list is displayed in the bottom 70% of the screen.

Multiple Applications Referencing a Single Configuration File

When modifying the launch fragment to launch multiple WCC applications, you can also specify that more than one application use the same `config.xml` configuration file. To do this, set the **jsxapppath** attribute in the launch fragment to point to the directory that contains the `config.xml` file you want the launch fragment to reference.

It may be beneficial to configure a single base-level WCC application that can then be used by several WCC applications without the need to repeat the same common files in separate project directories.

By setting the **jsxapppath** to this single base-level WCC application path, each application is launching from that path, sharing the `config.xml` file and potentially any other files (class files, GUI component files, etc.) that may have common use across several WCC applications.

In the example above, the first application is currently using the `config.xml` file in the `JSXAPPS\Accounts` directory; the second application is using the `config.xml` file in the `JSXAPPS\AccountDetail` directory.

You can have both applications use the same `config.xml` by setting both **jsxapppath** attributes to point to the same directory.

Also note that if multiple applications are using the same **jsxapppath**, they also share the same userAccess.xml file located in that path.

Multiple applications cannot share the same wccConfig.xml file (see WCC Configuration File). Each **wccapppath** attribute in the launch fragment must point to the directory containing the wccConfig.xml for each WCC application.

Opening Forms when Testing an Application Locally

If you are launching a non-deployed WCC application locally to test it during development, forms will not open without performing the work around described here.

For more information about issues with launching applications from the local file system, also see Cross-Domain Scripting.

To configure your application so that forms will open properly when testing from the local file system, follow the procedure below.

Procedure

1. Open the application's launch fragment with a text editor.

   For information about the location of the launch fragment, see Launching a Non-Deployed Application.

2. Locate the following line:

   ```
   <script type="text/javascript" language="javascript" src="../bpmresources/
   formsclient/formsclient.nocache.js"></script>
   ```

3. In the **src** attribute, replace the '..' with 'http://*Host*:*Port*',

   where:

   - *Host* is the name or IP address of the machine hosting the BPM runtime. *Port* is the port number used by the ActiveMatrix WebApp Implementation Type to communicate with web applications.

     For example:

     ```
     <script type="text/javascript" language="javascript" src="http://Austin:8080/
     bpmresources/formsclient/formsclient.nocache.js"></script>
     ```

     Also note that the httpConnector cannot be running SSL for this procedure to work, that is, the URL must include 'http', not 'https'.

4. Save and close the launch fragment.

   Important: Before deploying your application, ensure that you reverse the changes you made in this procedure.

Cross-Domain Scripting

Cross-domain scripting is a security vulnerability of web applications. If you trigger cross-domain scripting, and your browser doesn't allow it, the web application will not run (in the case of a WCC application, it will state that it is unable to establish a connection to the Action Processor).

Some browsers are more strict about enforcing cross-domain scripting than others; and newer versions of browsers tend to be more strict than older versions. Some browsers also provide methods to allow cross-domain scripting—see your browser's documentation for more information.

Cross-domain scripting affects accessing WCC applications in the following ways:

- **URL used to launch the application** - To prevent cross-domain scripting, it is best practice to ensure that the domain portion of the URL that is entered into the address line of the browser *exactly* matches the domain portion of the Action Processor URL specified in the application's `config.xml` file.

The domain consists of the "`http://`*`Host`*`:`*`Port`*" part of the URL.

The domain used to launch the application cannot differ in any way from the Action Processor's specified domain, otherwise cross-domain scripting may be triggered (depending on your browser). That is, you cannot use "http" in one and "https" in the other; you cannot use a host name in one and an IP address in the other; one host name cannot be unqualified and the other qualified; you cannot use "localhost" in one and "127.0.0.1" in the other.

To determine if cross-domain scripting is being used, the browser simply compares the URL domains as strings.

Note that in a production environment, where the WCC application and Action Processor are deployed to the same node and HTTP connector, the normal practice is to specify an empty string for the Action Processor URL in the application's `config.xml` file. When this is done, the URL of the Action Processor is inferred from the URL used to launch the application. This avoids the issue of comparing URL strings. For more information, see Action Processor.

- **Running the application from the local file system** - Because of the security risk of cross-domain scripting, some browsers will not allow you to run a web application (including a WCC application) from the local file system.

  Note that you would typically only run a WCC application from the file system in a testing and development environment. In a production environment, it is expected that the application will be deployed to a Web server and run from there. For information about deploying, see Deploying an Application After Customizing.

Single Sign-On Authentication

Workspace can be configured to use single sign on (SSO) authentication (for example, Kerberos).

You must configure the following parameters:

- **`showLogoutButton`** - Use this parameter to specify that the **Logout** button not be displayed. The **Logout** button is not needed when using Kerberos because the session is controlled external to the application.
  - To use the Configuration Administrator to configure this parameter, see Application.
  - To configure the parameter in the `config.xml` file, see showLogoutButton.
- **`authenticationMode`** - You must ensure that the **`authenticationMode`** parameter's **`mode`** attribute is set to "useSessionByDefault." This causes the application to use the current session, and prevents the Login dialog from being displayed.

  For information about this parameter, see Authentication Mode.

For more information about using SSO in ActiveMatrix BPM, see the *TIBCO ActiveMatrix® BPM Single-Sign On* guide.

Configuring the Application

The way in which you configure an application depends on whether the application is deployed or not.

- **A deployed application** - If configuring a deployed application, you can configure the application in one of two ways:
  - Use the Configuration Administrator (for more information, see Using the Configuration Administrator).

    When using the Configuration Administrator to configure a deployed application, you can use either of the provided editors—the graphical editor or the XML editor. The graphical editor modifies the application's `config.xml` file for you—it allows you to avoid directly modifying XML. The XML editor opens the application's `config.xml` file in the database—it is provided if you are comfortable directly modifying XML.

    For information about using the graphical editor, see Using the Graphical Editor to Configure the Application.

    For information about using the XML editor, see Modifying XML to Configure the Application.
  - Directly modify files on the BPM runtime machine - For more information, see Directly Modifying Files on the BPM Runtime Machine.

    For information about the location of files when directly modifying files on a BPM runtime machine, see Location of Files on a BPM Runtime Machine.
- **A non-deployed WCC application** - If configuring a non-deployed WCC application (which might be the Workspace application), open the `config.xml` file in your local development environment.

  The `config.xml` file is located in the following directory:

  ```
     StudioHome\wcc\version\JSXAPPS\WCCProjectName\
  ```

  where:

  - *`StudioHome`* is the directory in which TIBCO Business Studio was installed.
  - *version* is the version number of Workspace that was installed with TIBCO Business Studio.
  - *WCCProjectName* is the name of the General Interface Builder project that contains your custom application. If you are working with the Workspace application, this is “workspace”.

    If you are modifying the `config.xml` file to configure a non-deployed application, see Modifying XML to Configure the Application.

Configuration Parameter Summary

This topic provides a summary of each of the application configuration parameters available.

- **Application** - Disables the application and redirects users to specified URL. When using the graphical editor, you can also configure various items such as the location of help files, whether the **Logout** button is displayed, the action the application takes upon logout, etc.
  - To configure via the graphical editor, see Application.
  - To configure via XML, see Application.
- **Action Processor** - Specifies the URL that points to the Action Processor to which the application will connect at runtime.
  - To configure via XML, see Action Processor.

    As this is a pre-login control, it cannot be configured via the Configuration Administrator; see Pre-Login Configuration Parameters.

- **Session Monitor** - Specifies the period of time of inactivity before the user session times out.
  - To configure via the graphical editor, see Session Monitor.
  - To configure via XML, see Session Monitor.
- **HelpFileLocations** - Specifies the location of the Workspace and Organization Browser help files.
  - To configure via the graphical editor, see Application.
  - To configure via XML, see Help File Locations.
- **Options** - These specify the default user options that are applied the first time a user logs into the application.
  - To configure via the graphical editor, see Options.
  - To configure via XML, see Options.
- **User Preference Persistence** - Used to set the maximum number of bytes for the user preference data.
  - To configure via the graphical editor, see User Preference Persistence
  - To configure via XML, see User Preference Persistence.
- **Channel Identifier** - This identifies the work presentation, which controls the appearance of the form displayed for a work item.
  - To configure via the graphical editor, see Channel Identifier.
  - To configure via XML, see Channel Identifier.
- **Pre-login User Access** - This specifies the set of actions that the user has access to before logging in.
  - To configure via XML, see Pre-login User Access.

    As this is a pre-login control, it cannot be configured via the Configuration Administrator; see Pre-Login Configuration Parameters.
- **Login** - Specifies whether or not to display the **Remember User Name next time I login** check box on the Login dialog.
  - To configure via XML, see Login.

    As this is a pre-login control, it cannot be configured via the Configuration Administrator; see Pre-Login Configuration Parameters.
- **Login Link** - Provides the ability to include a hyperlink on the Login dialog.
  - To configure via XML, see Login Link.

    As this is a pre-login control, it cannot be configured via the Configuration Administrator; see Pre-Login Configuration Parameters.
- **Forms Configuration** - Specifies whether TIBCO Forms are reloaded or retrieved from cache, as well as whether or not the work item list is refreshed whenever a work item form is opened.
  - To configure via the graphical editor, see Forms Configuration.
  - To configure via XML, see Forms Configuration.
- **Browser Features -** Used to customize the appearance of the window when displaying work item forms in a separate browser window.
  - To configure via the graphical editor, see Browser Features.

  - To configure via XML, see Browser Features.

- **Post-login Caption** - Specifies the caption that is displayed in the browser window after a user has logged into the application.
  - To configure via the graphical editor, see Post-Login Caption.
  - To configure via XML, see Post-Login Caption.
- **Logging** - Used to control the Application Log, as well as whether or not the log contents are to be echoed to the Application Monitor.
  - To configure via the graphical editor, see Logging.
  - To configure via XML, see Logging.
- **First Quarter Month** - Specifies on which of the first three months of the year a quarter begins for the business using the application. This information is used when using the "This quarter" dynamic time period when filtering a list on a Date/Time attribute.
  - To configure via the graphical editor, see First Quarter Month.
  - To configure via XML, see First Quarter Month.
- **Column Sort** - Specifies whether clicking a column header in the work item, process instance, and event list causes the entire list (all pages) to be sorted on the column values, or whether only the currently displayed page is sorted on the column values.
  - To configure via the graphical editor, see Column Sort.
  - To configure via XML, see Column Sort.
- **Custom Callouts** - Specifies custom callout handlers, which allow you to *callout*, or *inject*, custom specifications in WCC applications.
  - To configure via the graphical editor, see Application.
  - To configure via XML, see Custom Callouts.
- **Get Binary LDAP Attributes** - Specifies whether or not LDAP attributes that contain binary data will appear in the LDAP Attributes list on the Organization Browser LDAP Container Editor dialog, as well as when you are mapping LDAP attributes to resource attributes.
  - To configure via the graphical editor, see Application.
  - To configure via XML, see Getting Binary LDAP Attributes.
- **Facade Display Names** - Specifies whether or not "display names" are shown for the custom attributes 1-40 in the client application. This requires a facade to be deployed to the node.
  - To configure via the graphical editor, see Application.
  - To configure via XML, see Facade Display Names.
- **Event Paging** - Specifies whether or not to retrieve and display a total count of events in the Event Viewer.
  - To configure via the graphical editor, see Application.
  - To configure via XML, see Event Paging Model.
- **authenticationMode** - Specifies the method used to authenticate users.
  - To configure via XML, see Authentication Mode.

    As this is a pre-login control, it cannot be configured via the Configuration Administrator; see Pre-Login Configuration Parameters.

- **showLogoutButton** - Controls whether or not the **Logout** button is displayed.
  - To configure via the graphical editor, see Application.
  - To configure via XML, see showLogoutButton.
- **logoutPath** - Specifies the action taken upon logout.
  - To configure via the graphical editor, see Application.
  - To configure via XML, see Logout Path.

Pre-Login Configuration Parameters

There are configuration parameters that pertain to configuration that is used *before* the user has logged in.

Those parameters are:

- **Action Processor** - Specifies the URL that points to the Action Processor to which the application will connect at runtime.
- **Pre-Login User Access** - Specifies the set of actions that the user has access to before logging in.
- **Login** - Specifies whether or not to display the **Remember User Name next time I login** check box on the Login dialog.
- **LoginLink** - Provides the ability to include a hyperlink on the Login dialog.
- authenticationMode - Specifies the method used to authenticate users.

If you are configuring an application that has not yet been deployed to a runtime machine, these parameters can be specified as needed in the `config.xml` file in your development environment; they will take effect when the application is deployed.

However, if you are configuring an application that has already been deployed to a runtime machine, these parameters *must* be configured in the `config.xml` file on the file system of the runtime machine. You cannot use the Configuration Administrator to modify the pre-login parameters because the Configuration Administrator is used to modify the parameters that are stored in the database. These pre-login parameters are not stored in the database—they are always read from the file system because the application does not access the database before the user logs in.

The pre-login parameters are still visible in the XML Editor portion of the Configuration Administrator, but modifying them using that editor will have no effect. They do not appear in the Configuration Administrator Graphical Editor.

As described in Directly Modifying Files on the BPM Runtime Machine, you can directly modify the `config.xml` file on the runtime machine if you are using the Workspace application. However, if you are using a custom WCC application, you cannot directly modify the `config.xml` file on the runtime machine because you cannot determine its location (its location, which is determined when the application is deployed, contains GUIDs).

Therefore, to make a change to any of the four pre-login configuration parameters for a deployed custom WCC application, you must make the change to the `config.xml` file in your development environment file system. You will then need to create a new `.war` file for your custom application, and redeploy the application using ActiveMatrix Administrator. For information about the process of creating the `.war` file and deploying an application, see Deploying an Application After Customizing.

Using the Graphical Editor to Configure the Application

When configuring a deployed application, you can use the graphical editor provided in the Configuration Administrator.

Procedure

1. In the Workspace application, click the **Admin** button in the application toolbar, then select **Configuration**.
2. On the Configuration Administrator dialog, click **config.xml** in the left pane.
3. Click the **Graphical Editor** tab in the right pane.

   The right pane is sub-divided into two sections: the left section represents records in the `config.xml` file for configuring the application; the right section contains fields, check boxes, etc., that represents the attributes in the `config.xml` records.

Application

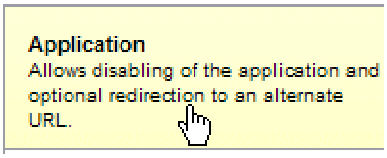
The Application configuration is used to configure application-level functionality.

It can be used to configure the following:

- Disable the application, and optionally redirect users who access the disabled application to a specified URL. This is described below.
- Specify the location of Workspace and Organization Browser help files.
- Get LDAP attributes that contain binary data when creating and configuring LDAP containers using the Organization Browser.
- Specify whether or not display names are used for attributes 1-40 when a work list facade is deployed.
- Specify whether or not the total number of events is retrieved from the server and displayed in the Event Viewer.
- Specify whether or not the **Logout** button is displayed.
- Specify the action taken upon logout.
- Configure custom callout handlers, which allow you to *callout*, or *inject*, custom specifications in WCC applications.

Procedure

1. In the Configuration Administrator, click **config.xml** in the left pane.
2. In the **Graphical Editor** tab, click in the **Application** section:



Application
Allows disabling of the application and optional redirection to an alternate URL.

3. Configure the application using the following fields that appear in the right pane:

   - **Disable Application** - Checking this box disables the application, making it inaccessible. For more information, see Disabling a Currently Deployed Application.

     Note that the System Administrator can still log into a disabled application—see Accessing a Disabled Application by the System Administrator.

   - **Redirect URL** - A URL that you would like users who access the disabled application to be redirected.

     If the **redirectUrl** attribute is set to an empty string, instead of redirecting the user, the message specified in the *txtAppDisabled* variable is displayed (this variable is located in the `...\JSXAPPS`

`\base\locale\locale.xml` file). By default, the *txtAppDisabled* variable contains the message "This application is disabled.", but can be changed and/or localized if desired.

- **Help File Locations** - The following fields are used to specify the location of help files. By default, the help files are located on the TIBCO Documentation website, https://docs.tibco.com. Some users, however, do not have Internet access. In those situations, the documentation can be downloaded and stored locally, then the path to the documentation can be specified in these fields.

  - **Workspace User's Guide URL** - This points to the *TIBCO Workspace User's Guide*. The default value points to the TIBCO ActiveMatrix BPM documentation on the TIBCO Documentation web site. If you have downloaded and stored the help files locally as described in Customizing Location of Help Files, enter the path/URL to the local *TIBCO Workspace User's Guide* in this field.
  - **Organization Browser User's Guide URL** - This points to the TIBCO Organization Browser User's Guide. The default value points to the TIBCO ActiveMatrix BPM documentation on the TIBCO Documentation web site. If you have downloaded and stored the help files locally as described in Customizing Location of Help Files, enter the path/URL to the local Organization Browser User's Guide in this field.

- **Get Binary LDAP resource attributes** - Checking this box causes LDAP attributes that contain binary data to appear in the **LDAP Attributes** list on the Organization Browser LDAP Container Editor dialog, as well as when you are mapping LDAP attributes to resource attributes. As the binary data in the LDAP attributes is base-64 encoded, binary LDAP attributes can be mapped to resource attributes of type String.

  Unchecking this box causes LDAP attributes containing binary data to not be displayed.

- **Use facade display names for attributes 1-40** - Checking this box causes "display names" to be shown for custom attributes 1-40 in the client application.

  A work list facade must be deployed to the node. Work list facades are defined using TIBCO Business Studio.

- **Retrieve total count of events in Event Viewer** - Checking this box causes the total number of events to be retrieved from the server and displayed in the Event Viewer. This can be used to prevent the counts from being retrieved to improve performance when there are a large number of events.

  If the total count is not retrieved and displayed, the user can still page through the list of events. When the end of the list is reached, a "You have reached the end of the list" message is displayed when the last page is reached.

- **Show the Logout Button** - Checking this box causes the **Logout** button to be displayed in the application. Uncheck this box if you are using single sign on (SSO) authentication (for example, Kerberos). In an SSO implementation using Kerberos, the session is controlled external to the application, therefore the **Logout** button is not needed.

  Also note that if you are hiding the **Logout** button in an SSO implementation, you should also ensure that the authenticationMode parameter's **`mode`** attribute is set to "useSessionByDefault" so that the existing external session is used, and the Login dialog is not displayed.

- **logoutPath** - This controls the action taken when the **Logout** button is pressed in the application. For details and a list of valid values, see Logout Path.

- **Enable Callout Interfaces** - Checking this box allows you to enable and add the classes for each callout handler to implement. For information about callout handlers in general, see the "WCC Public Methods" chapter in the *TIBCO Workspace Components Developer Guide*. And for more information about configuring callout handlers using the Configuration Administrator, see the "Using Configuration Administrator to Configure Callout Handlers" section in the same chapter.

Session Monitor

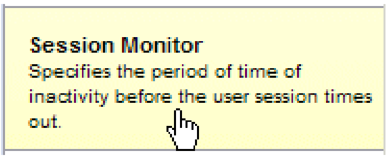
You can specify that if a user of the application is inactive for a certain period of time, the user's session will time out and automatically log the user out.

You can also specify when a warning dialog is to be displayed, informing the user that the session is about to time out. The user can click **OK** on this warning dialog to continue the session. If the user does not respond to the warning message, the session will time out in the specified period of time.

Also note that a time-out period can be specified on the server. The server time-out overrides the time-out specified here. That is, if the server time-out period is less than the time-out specified here, the server time-out is used instead.

Procedure

1. In the Configuration Administrator, click **config.xml** in the left pane.
2. In the **Graphical Editor** tab, click in the **Session Monitor** section**:**



Session Monitor
Specifies the period of time of inactivity before the user session times out.

3. Configure the Session Monitor using the following fields that appear in the right pane:

- **Disable Session Monitor** - Check this check box to disable the Session Monitor—when disabled, the application will not time out.
- **Time Out (minutes)** - The number of minutes of user inactivity before the session will time out. The user is automatically logged out upon timing out.

  Minimum: 5

  Maximum: none

  Default: 30
- **Warning (minutes)** - The number of minutes *before* the time out will occur that a warning dialog is displayed informing the user that the session is about to time out.

  Minimum: 1

  Maximum: 1/3 of the value specified for the time-out period.

  Default: 5

Options

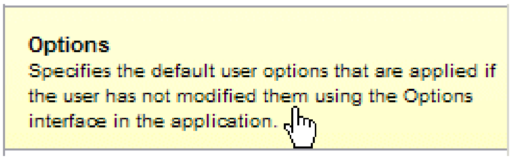
The Workspace application contains an Options dialog from which each user can specify personal *user options*. User options establish default settings for each user who logs into the application. These include things such as the list to display first, the size and location of forms, the language to display, etc.

For information about setting user options from the Options dialog, see the *TIBCO Workspace User's Guide*.

There are *default* user options defined in the system that each new user inherits until they specify their own user options on the Options dialog. The following procedure describes how to specify these default user options. (Note that the Options dialog also contains a **Defaults** button that sets all of the options for the user to the default user options specified here.)

Procedure

1. In the Configuration Administrator, click **config.xml** in the left pane.
2. In the **Graphical Editor** tab, click in the **Options** section**:**



Options

Specifies the default user options that are applied if the user has not modified them using the Options interface in the application.

3. Configure the default user options using the following fields that appear in the right pane:

Result

- Display Section
  - **Initial Display** - Specifies which list to display immediately after login.
  - **Language** - Specifies the default language for the application. The available localized languages are listed, as well as "Use the browser's language", which causes the default language to be set to that of the browser being used.
  - **Default Work View** - The name of the work view to display initially. An empty string causes the first work view in the list to be displayed.
  - **Default Process View** - The name of the process view to display initially. An empty string causes the first process view in the list to be displayed.
  - **Default Event View** - The name of the event view to display initially. An empty string causes the first event view in the list to be displayed.
  - **Work View Caption** - Specifies whether the work view name or description is to be displayed in the caption of the work item list after selecting a work view.
  - **Process View Caption** - Specifies whether the process view name or description is to be displayed in the caption of the process instance list after selecting a process view.
  - **Event View Caption** - Specifies whether the event view name or description is to be displayed in the caption of the event list after selecting an event view.
  - **Display milliseconds in work item and process instance lists** - Checking this box causes milliseconds to be displayed in date/times in the work item and process instance lists.
- Work Items Section
  - **Auto-refresh lists of work items** - Checking this box causes the auto-refresh feature to be on by default on work item lists. If this box is checked, the **Auto-refresh interval** field in enabled, in which you can specify the number of seconds between automatic refreshes of work item lists—the minimum is 5 seconds.
  - **Preview** - The drop-down list in this area allows you to specify whether or not to display the preview pane by default, as well as whether work item forms should be docked to the preview pane, or whether they should be floating.

    If the selection is made to show the preview pane and dock work item forms to the preview pane, the **Resize the preview pane...** check box is enabled. Checking this box allows you to enter a percentage in the field below the check box that specifies the percentage of the preview pane the work item form will occupy.
  - **Forms** - The drop-down list allows you to specify whether floating forms should be opened in a dialog or a separate browser window.

The **Default position and size (px) for floating windows** fields allow you to specify the position and size, in pixels, of floating windows. Note that the **Left** and **Top** fields are disabled if the **Center floating windows on the screen** box is checked, and all of these fields are disabled if the **Display floating windows fullscreen** box is checked.

The **Remember floating window position** box causes the system to remember the size and position of the floating window if it is manually moved.

The **Display floating windows fullscreen** box causes floating windows to be fullscreen, disregarding the position and size field specifications.

The **Center floating windows on the screen** box causes floating windows to be centered, disregarding the **Left** and **Top** field specifications.

- Business Services Section

  - **Number of most recently started services** - The number of business services that can appear in the **Recent** section in the business service list. These are the recently started business sevices.
  - **Forms** - The first field drop-down list in this section allows you to specify whether business service forms should be docked in the preview pane, or whether they should be floated in either a dialog or a separate browser window.

    The second field drop-down list is used to specify whether you want floating windows to display in a dialog or a separate browser window.

    The **Default position and size (px) for floating windows** fields allow you to specify the position and size, in pixels, of floating windows. Note that the **Left** and **Top** fields are disabled if the **Center floating windows on the screen** box is checked, and all of these fields are disabled if the **Display floating windows fullscreen** box is checked.

    The **Remember floating window position** box causes the system to remember the size and position of the floating window if it is manually moved.

    The **Display floating windows fullscreen** box causes floating windows to be fullscreen, disregarding the position and size field specifications.

    The **Center floating windows on the screen** box causes floating windows to be centered, disregarding the **Left** and **Top** field specifications.

- Processes Section

  - **Number of most recently started processes** - The number of processes that can appear in the **Start Process Instance** button drop-down list. This drop-down list shows the user the processes that have been started recently.

- Appearance Section

  - **Screen layout** - Specifies whether lists are displayed side-by-side, stacked top-to-bottom, or displayed in floating dialogs.
  - **Font Size** - Specifies the size of the font to display in the application. The options are:

    - Small - 10px
    - Medium - 12px
    - Large - 14px
    - Default - The font size is determined by the setting in the `workspaceCSS.xml` file.

User Preference Persistence

When using a WCC application, user preference data is always stored on the server. This includes things like view definitions, column settings, information on the Options dialog, list filter and sort settings, etc.

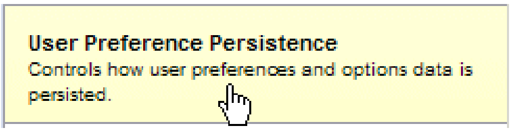
This configuration is used to set the maximum number of bytes for the user preference data. This value needs to be set at or below the field size supported by the database used on the server.

If this value is too small, processing the data at the server will be inefficient; if it's too large, the database will throw an exception when is attempts to parse the message containing user preference data.

Note that the character encoding used should be taken into consideration when determining the maximum data size.

Procedure

1. In the Configuration Administrator, click **config.xml** in the left pane.
2. In the **Graphical Editor** tab, click in the **User Preference Persistence** section**:**



User Preference Persistence
Controls how user preferences and options data is persisted.

3. Configure the maximum data size using the following field that appears in the right pane:
    - **Maximum number of bytes for user preference data** - Set according to the description above. The default is 1992; the minimum is 10.

Channel Identifier

When a project is created in TIBCO Business Studio, *presentation channels* can be created for the project. Each presentation channel specifies the way in which forms that are part of the project will be presented to the user at runtime.

The Presentation Channel editor is displayed in TIBCO Business Studio by selecting **Windows** > **Preferences** > **Presentation Channels** .

There is only a single presentation channel defined in Workspace at this time:

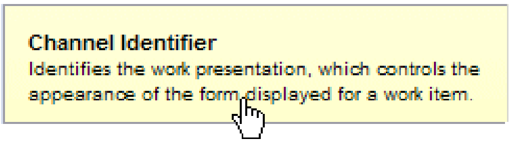
- **GIGWTPull_DefaultChannel** - This is the default channel for Google Web Toolkit (GWT) forms.

The application's `config.xml` file contains a **channelId** parameter that identifies which presentation channel to use when displaying work item forms when using an instance of that application. (At this time, each instance of the application can be bound to a single presentation channel.)

By default, Workspace is configured to use GWT forms (the **channelId** attribute is set to **GIGWTPull_DefaultChannel** by default).

Procedure

1. In the Configuration Administrator, click **config.xml** in the left pane.
2. In the **Graphical Editor** tab, click in the **Channel Identifier** section**:**



Channel Identifier
Identifies the work presentation, which controls the appearance of the form displayed for a work item.

3. Configure the presentation channel using the following field that appears in the right pane:

   - **Channel Identifier** - Set to the value of the channel ID defined in TIBCO Business Studio for the presentation channel you want to use for this Workspace.

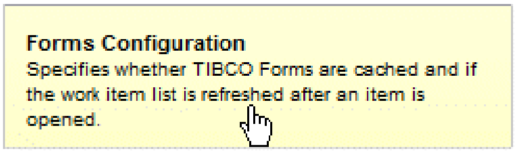
     Note that if there is no channel ID specified here, or the channel ID specified does not exist, the default presentation channel (GIGWTPull_DefaultChannel) is used.

Forms Configuration

This configuration specifies whether TIBCO Forms are reloaded or retrieved from cache, as well as whether or not the work item list is refreshed whenever a work item form is opened.

Procedure

1. In the Configuration Administrator, click **config.xml** in the left pane.
2. In the **Graphical Editor** tab, click in the **Forms Configuration** section**:**



Forms Configuration
Specifies whether TIBCO Forms are cached and if the work item list is refreshed after an item is opened.

3. Configure the forms configuration using the following fields that appear in the right pane:

   - **Reload TIBCO Forms on each request** - Checking this box causes forms and their resources to be reloaded on each form request, rather than retrieving the resources from the browser cache.
   - **Refresh work item list after an item is opened** - If this option is checked, a full refresh of the work item list is performed whenever a work item is opened (which can have an impact on performance). If this option is not checked, the refresh is not performed, but the status icon for the opened work item is changed to reflect that it is currently open.

Browser Features

In a WCC application, the user can specify whether work item forms are opened in the Preview Pane or in a *floating* work item form. This can be set from the **View** > **Preview** menu on the work item list.

If the user chooses to display floating work item forms, he can further specify that the floating form be displayed in either a *dialog* or a *separate browser window*. This can be specified in User Options. Note, however, that the user can control whether the form is displayed in a dialog or separate browser window via User Options *only* if the configuration in this section gives the user the ability to do that (see the description of the **dialog** check box below).

The configurations specified here allow you to set the behavior of the window (things such as whether the window is resizable, whether or not a status bar is displayed, etc.) when the form is opened in a separate browser window.

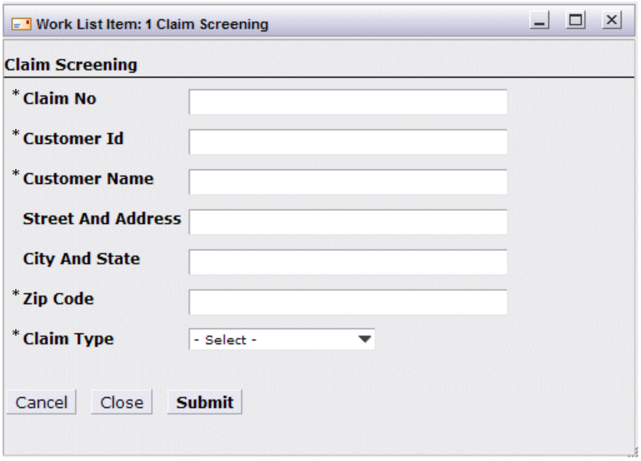
The following describes the differences in behavior between dialogs and separate browser windows:

- **Floating Window Outside Application Window**:
  - Separate browser windows can be floated outside the parent application's window.
  - Dialogs cannot be floated outside the parent application's window.
- **Close as child window**:
  - Separate browser windows do not close (or minimize) when the parent is closed (or minimized).

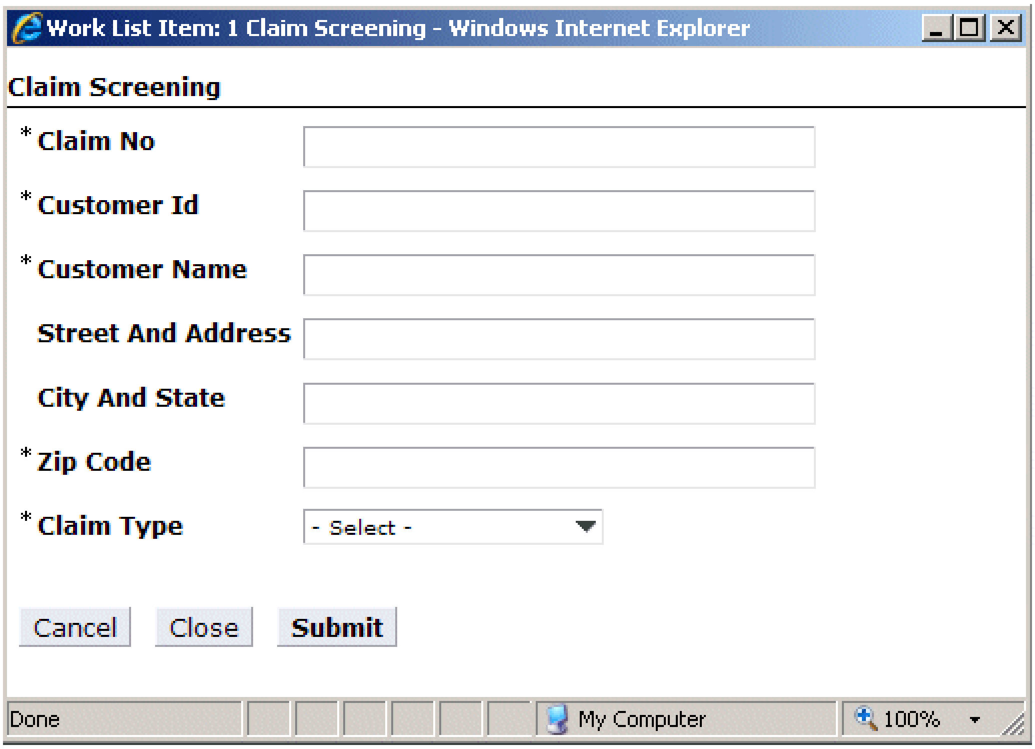
  - Dialogs are children of the parent window. Therefore if the parent window is closed (or minimized), the dialog is also closed (or minimized).
- **Browser Feature Attributes**:
  - Separate browser windows can be further controlled using the configuration options described in this section.
  - Dialogs*cannot* be controlled using the configuration options described in this section.

The following illustrations show you the general differences in appearance between a form in a dialog and in a separate browser window.

Work Item Form in a Dialog



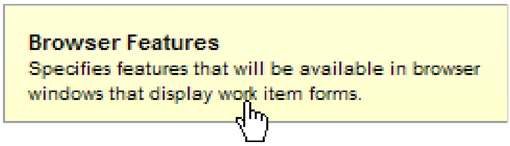
Work Item Form in a Separate Browser Window



Also, the extent to which you can customize the browser window appearance depends on the type of browser (Internet Explorer, Firefox, Safari, or Chrome) you are using, as described below.

Procedure

1. In the Configuration Administrator, click **config.xml** in the left pane.
2. In the **Graphical Editor** tab, click in the **Browser Features** section**:**



Browser Features
Specifies features that will be available in browser windows that display work item forms.

3. Configure the browser features using the following selections that appear in the right pane:

   - **channelMode** - (applicable to IE only) Specifies whether or not to display the window in “theater mode”, that is, as a maximized window.
   - **dialog** - (applicable to all browsers) Specifies whether or not to display the window as a “dialog”. If checked, it forces work item forms to be displayed in a dialog, regardless the setting in User Options. If unchecked, the user can control whether or not the form is opened in a dialog or separate browser window via a setting in User Options.
   - **directories** - (applicable to Firefox only) Specifies whether or not to display the "Bookmarks Toolbar". (Firefox users can force new windows to always render the Bookmarks Toolbar by setting `dom.disable_window_open_feature.directories` to true in **about:config** or in their `user.js` file.)
   - **location** - (applicable to all browsers) Specifies whether or not to display the "Navigation Toolbar" in Internet Explorer, the "Location Bar" in Firefox, or the "Toolbar" and "Bookmarks Bar" in Safari. (Firefox users can force new windows to always render the Navigation Toolbar by setting `dom.disable_window_open_feature.location` to true in **about:config** or in their `user.js` file.
   - **menubar**[1] - (applicable to all browsers) Specifies whether or not the browser window should display a “Menu Bar”. (Firefox users can force new windows to always render the Menu Bar by setting `dom.disable_window_open_feature.menubar` to true in **about:config** or in their `user.js` file.)
   - **resizable** - (applicable to IE only) Specifies whether or not the browser window can be manually resized using the lower right corner of the window.

     Note - This configuration may not work as expected. Tests on various systems has shown that on some the window can be resized, while on others, it cannot. The exact cause(s) of the unexpected behavior remains unknown, although it is thought to be a combination of the browser being used, the browser version, and browser security settings.
   - **scrollbars**1 - (applicable to IE and Firefox only; scrollbars are always displayed, if needed, in Safari and Chrome) Controls the display of scrollbars, on work item forms opened in a separate browser window, when the form content overflows the browser dimensions.
   - **status**1 - (applicable to IE only) Specifies whether or not the browser window displays a status bar on the bottom of the window. (Firefox always displays the status bar.)
   - **toolbar**1 - (applicable to all browsers) Specifies whether or not to display the Toolbar across the top of the window. This bar contains buttons/icons for Back, Forward, Refresh, Home, etc.

     In Internet Explorer this bar is referred to as the “Command Bar”, in Firefox it’s the “Tab Bar”, and in Safari it’s the “Toolbar” and “Bookmarks Bar”.

     Firefox users can force new windows to always render the Tab Bar by setting `dom.disable_window_open_feature.toolbar` to true in **about:config** or in their `user.js` file.

[1] Selecting menubar, scrollbars, status, or toolbar when using Chrome causes the form to open in a new tab rather than a separate browser window.

Post-Login Caption

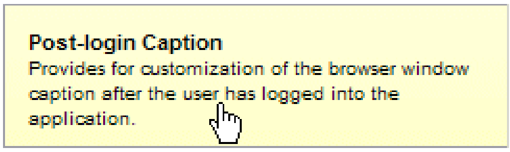
This configuration specifies the caption that is displayed on the top of the browser window after a user has logged into the application.

By default, the caption is set to:

```
"Workspace - <Username>"
```

Procedure

1. In the Configuration Administrator, click **config.xml** in the left pane.
2. In the **Graphical Editor** tab, click in the **Post-login Caption** section**:**



Post-login Caption

Provides for customization of the browser window caption after the user has logged into the application.

3. Configure the post-login caption using the following field that appears in the right pane:

- **Post-login caption pattern** - Specify the pattern for the caption you would like displayed. The following placeholders can be used in the pattern string to display various information:
- **%productname%** - This placeholder is replaced with the name of the product.
- **%username%** - This placeholder is replaced with the name of the logged-in user.

  Any or all of the placeholders can be specified or omitted.

  If a placeholder is specified that is not available, it will be replaced with a zero-length string.

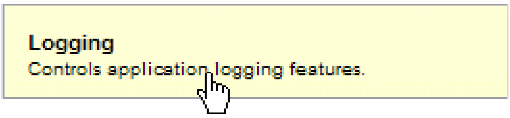
Logging

This configuration is used to control the Application Log.

For information about the Application Log, see Application Log.

Procedure

1. In the Configuration Administrator, click **config.xml** in the left pane.
2. In the **Graphical Editor** tab, click in the **Logging** section**:**



Logging

Controls application logging features.

3. Configure the logging option using the following fields that appear in the right pane:

- **Application Log Level** - Specifies the level of log messages that will be written to the Application Log. Select OFF from the drop-down list to turn off the Application Log.
- **Echo application log contents to the application monitor** - Checking this box causes the log contents to be echoed to the Application Monitor (see Application Monitor).

First Quarter Month

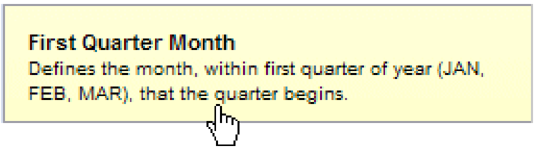
This configuration is used to specify on which of the first three months of the year a quarter begins for the business using the WCC application.

This information is used when using the “This quarter” dynamic time period when filtering a list on a Date/Time attribute. For information about using dynamic time periods, see the “Filtering Lists” chapter in the *TIBCO Workspace User’s Guide.*

This is also used if the **~THISQUARTER~** dynamic time-period variable has been used when creating custom event filters. For more information, see Event Filter String**.**

Procedure

1. In the Configuration Administrator, click **config.xml** in the left pane.
2. In the **Graphical Editor** tab, click in the **First Quarter Month** section**:**



First Quarter Month
Defines the month, within first quarter of year (JAN, FEB, MAR), that the quarter begins.

3. Configure the first quarter month using the following field that appears in the right pane:

   - Specify the beginning month of the first business quarter - Select “Jan”, “Feb”, “or “Mar” to indicate on which of the first three months of the year a quarter begins.

     Note that this is not meant to indicate the month on which the first quarter of the year begins (although it might)—it indicates on which of those months any quarter begins. For example, if the company’s first quarter actually starts in December and runs through February, you would set the value attribute to “MAR” because that is the month in which a quarter begins in the first three months of the year.

Column Sort

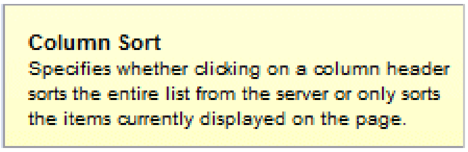
This configuration controls how list contents are sorted when a column header is clicked in the lists of work items, process instances, and events.

If this parameter is set to "server", a request is sent to the server, which causes *all* available data to be sorted, including pages that have not been accessed. If set to "page", only the page that is currently displayed is sorted (a request is not sent to the server).

Note that when this parameter is set to "server," not all columns are sortable, as some attributes cannot be sorted by the server. If the user attempts to sort a non-sortable column by clicking on the header, a message is displayed.

Procedure

1. In the Configuration Administrator, click **config.xml** in the left pane.
2. In the **Graphical Editor** tab, click in the **Column Sort** section**:**



Column Sort
Specifies whether clicking on a column header sorts the entire list from the server or only sorts the items currently displayed on the page.

3. Configure the column sort option using the following optoins that appear in the right pane:

- **server** - Causes *all* available data to be sorted by the values in the column, including pages that have not been accessed.
- **page** - Only the page that is currently displayed is sorted by the values in the column.

Modifying XML to Configure the Application

You can directly modify the XML in the application’s `config.xml` file to configure the application.

This can be done via the Configuration Administration (for deployed applications) or via the file system. For more information, see Introduction.

Application

This configuration allows you to disable the application, and optionally redirect users who access the disabled application to a specified URL.

Note that the System Administrator can still log into a disabled application—see Accessing a Disabled Application by the System Administrator.

For more information, see Disabling a Currently Deployed Application.

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system, see Introduction).

2. Locate the **Application** record in the `config.xml` file.

```
<record jsxid="Application" type="Workspace"
        disabled="false"
        redirectUrl=""/>
```

3. Modify the attributes as follows:

   - **disabled** - “true” disables the application, making it inaccessible.
   - **redirectUrl** - A URL that you would like users who access the disabled application to be redirected.

     If the **redirectUrl** attribute is set to an empty string, instead of redirecting the user, the message specified in the *txtAppDisabled* variable is displayed (this variable is located in the `...\JSXAPPS\base\locale\locale.xml` file). By default, the *txtAppDisabled* variable contains the message “This application is disabled.”, but can be changed and/or localized if desired.

Action Processor

This configuration specifies the URL that points to the Action Processor to which the application will connect at runtime.

Note that in a production environment, the application and the Action Processor must be running on the same machine. Therefore, the **baseUrl** attribute for the **ActionProcessors** record in the `config.xml` file is set to an empty string by default:

```
<record jsxid="ActionProcessors" type="workspace">
    <ActionProcessor
       weighting="100"
       baseUrl="">
    </ActionProcessor>
</record>
```

This causes the URL to the Action Processor to be determined at runtime based on the URL entered in the browser to start the application.

In a development/testing environment, however, you will likely be running the application on a local machine. Therefore, you will need to specify the URL to the Action Processor in the **baseUrl** attribute.

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system, see Introduction).

2. Locate the **ActionProcessors** record in the `config.xml` file.

3. Set the **baseUrl** attribute to the URL of the Action Processor. The string in the **baseUrl** attribute must be in the form:

   ```
      http://Host:Port/bpm/actionprocessor/actionprocessor.servlet
   ```

   where:

   - `Host` is the name or IP address of the machine hosting the Action Processor. This must be the same machine on which the application is running (i.e., the Action Processor and the application must be running on the same machine).
   - `Port` is the port number used by the ActiveMatrix WebApp Implementation Type to communicate with web applications.

     Note - The **weighting** attribute is not used at this time.

     For example:

   ```
   <record jsxid="ActionProcessors" type="workspace">
       <ActionProcessor
          weighting="100"
          baseUrl="http://Austin:8080/bpm/actionprocessor/actionprocessor.servlet">
       </ActionProcessor>
   </record>
   ```

4. Save and close the `config.xml` file.

Session Monitor

You can specify that if a user of the application is inactive for a certain period of time, the user’s session will time out and automatically log the user out.

You can also specify when a warning dialog is to be displayed, informing the user that the session is about to time out. The user can click **OK** on this warning dialog to continue the session. If the user does not respond to the warning message, the session will time out in the specified period of time.

Also note that a time-out period can be specified on the server. The server time-out overrides the time-out specified here. That is, if the server time-out period is less than the time-out specified here, the server time-out is used instead.

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system, see Introduction).

2. Locate the **SessionMonitor** record in the `config.xml` file:

```
<record jsxid="SessionMonitor" timeout="30" warning="5" disable="false"
type="Workspace" />
```

3. Specify the record’s attributes as follows:

- **timeout** - The number of minutes of user inactivity before the session will time out. The user is automatically logged out upon timing out.

  Minimum: 5

  Maximum: none

  Default: 30

- **warning** - The number of minutes *before* the time out will occur that a warning dialog is displayed informing the user that the session is about to time out.

  Minimum: 1

  Maximum: 1/3 of the value specified for the time-out period.

  Default: 5

- **disable** - Set to “true” to disable session monitoring—the application will not time out; set to “false” to enable session monitoring.

  Default: false

4. Save and close the `config.xml` file.

Help File Locations

This configuration is used to specify the location of Workspace and Organization Browser help files.

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (that is, via the Configuration Administrator or via the file system), see Introduction.

2. Locate the **`HelpFileLocations`** record in the `config.xml` file.

   ```
   <record jsxid="HelpFileLocations"
      workspace="https://docs.tibco.com/pub/amx-bpm/n.n.n/doc/html/bpmhelp/
   ClientOSWS/GUID-BEC930CC-206C-4026-8DAB-95AA8DF9A56F.html"
      orgBrowser="https://docs.tibco.com/pub/amx-bpm/n.n.n/doc/html/bpmhelp/
   ClientOSWS/GUID-654ED7AC-E0A2-4815-8727-5047C53F872B1.html"
   </record>
   ```

   where *n.n.n* is the current version number of Workspace and the Organization Browser.

3. Modify the attributes as follows:

   - **`workspace`** - This points to the *TIBCO Workspace User's Guide*. The default value points to the TIBCO ActiveMatrix BPM documentation on the TIBCO Documentation website. If you have downloaded and stored the help files locally as described in Customizing Location of Help Files, enter the path/URL to the local *TIBCO Workspace User's Guide* in this field.

     If **Help** is selected from the Workspace Login dialog, the setting in this attribute is always used, rather than the setting specified in the Configuration Administrator because settings in the Configuration Administrator are stored in the database, which are not loaded until after login.

   - **`orgBrowser`** - This points to the *TIBCO ActiveMatrix BPM Organization Browser User's Guide*. The default value points to the TIBCO ActiveMatrix BPM documentation on the TIBCO Documentation web site. If you have downloaded and stored the help files locally as described in Customizing Location of Help Files, enter the path/URL to the local Organization Browser User's Guide in this field.

Options

The Workspace application contains an Options dialog from which each user can specify their personal *user options*. User options establish default settings for each user who logs into the application. These include things such as the list to display first, the size and location of forms, the language to display, etc.

For information about setting user options from the Options dialog, see the *TIBCO Workspace User's Guide*.

There are *default* user options defined in the system that each new user inherits until they specify their own user options on the Options dialog. These default user options are defined in the application's configuration file, `config.xml`. (Note that the Options dialog also contains a **Defaults** button that sets all of the options for the user to the default user options specified in the `config.xml` file.)

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system, see Introduction).

2. Locate the **Options** record:

```
<record jsxid="Options" type="Workspace">
  <Options>
    <Display>
      <InitialDisplay initialList="WorkViews"></InitialDisplay>
           .
           .
           .
```

3. Using the information in the table shown below, change the values of the appropriate element attributes to the desired values.

4. When finished, save and close the `config.xml` file.

Element	Attribute	Possible Values	Meaning
<Display> <InitialDisplay>	initialList	WorkViews	Displays the work views list after login.
		PTViews	Displays the process views list after login (the "PT" is in reference to the former "process template views" name).
		EventViews	Displays the event views list after login.
<Display> <DefaultView>	workView	string	The name of the work view to display initially. An empty string causes the first work view in the list to be displayed.

Element	Attribute	Possible Values	Meaning
	processView	*string*	The name of the process view to display initially. An empty string causes the first process view in the list to be displayed.
	eventView	*string*	The name of the event view to display initially. An empty string causes the first event view in the list to be displayed.
<Display> <Captions>	workView	name	Causes the work view name to be displayed in the caption of the work item list after selecting a work view.
		description	Causes the work view description to be displayed in the caption of the work item list after selecting a work view.
	processView	name	Causes the process view name to be displayed in the caption of the process instance list after selecting a process view.
		description	Causes the process view description to be displayed in the caption of the process instance list after selecting a process view.
	eventView	name	Causes the event view name to be displayed in the caption of the event list after selecting an event view.
		description	Causes the event view description to be displayed in the caption of the event list after selecting an event view.
<Display> <Times>	showMilliseconds	true	Milliseconds are displayed in date/times in the work item and process instance lists.

Element	Attribute	Possible Values	Meaning
		false	Milliseconds are not displayed in date/times in the work item and process instance lists.
<Display> <Language>	localeKey	string	The default locale to use in the application(1) . If this string is empty, or the <Language> element or localeKey attribute is missing, the language defaults to that of the browser. Also, if the locale file specified by the localeKey value cannot be found on the system, "en_US" becomes the default locale.
<WorkItems> <AutoRefresh>	refresh	true	Causes the auto-refresh feature to be on by default on work item lists.
		false	Causes the auto-refresh feature to be off by default on work item lists.
	refreshInterval	integer	Number of seconds between automatic refreshes of work item lists when AutoRefresh = true. Minimum = 5 seconds; Maximum of 8 digits.
<WorkItems> <Preview>	previewForms	dock	Work item forms are docked in the Preview Pane.
		float	Preview Pane is displayed, but work item forms are displayed in floating windows.
		off	Preview Pane is turned off. Work item forms are displayed in floating windows.

Element	Attribute	Possible Values	Meaning
	previewResize	true	Causes the Preview Pane to automatically resize when a work item is opened in the Preview Pane. It is resized to the percentage specified in the **previewSize** attribute.
		false	The Preview Pane is not resized when a work item is opened in the Preview Pane.
	previewSize	integer	The percentage of the viewing area the Preview Pane will occupy when it is automatically resized (**previewResize** must be set to "true").
<WorkItems> <Forms>	float	dialog	Floating windows containing a work item form are displayed in a separate dialog.
		browser	Floating windows containing a work item form are displayed in a separate browser window.
	floatLeft	integer	The floating window is positioned this number of pixels from the left. (Only applicable if both the **floatFullscreen** and **floatCenter** attributes are false.)
	floatRight	integer	The floating window is positioned this number of pixels from the top. (Only applicable if both the **floatFullscreen** and **floatCenter** attributes are false.)
	floatWidth	integer	The width (in pixels) of the floating window. (Only applicable if the **floatFullscreen** attribute is false.)

Element	Attribute	Possible Values	Meaning
	floatHeight	integer	The height (in pixels) of the floating window. (Only applicable if the **floatFullscreen** attribute is false.)
	floatFullscreen	true	Floating windows are displayed full screen.
		false	Floating windows are not displayed full screen (other attributes specify position/ size.)
	floatCenter	true	Floating windows are displayed centered. (Use **floatWidth** and **floatHeight** attributes to determine size.)
		false	Floating windows are not displayed centered (other attributes specify position/ size.)
	floatRememberPosition	true	The system remembers the size and position of the floating window if you manually move it on your screen.
		false	Future floating windows are opened in the position and size specified by the other *float* attributes.
<BusinessServices> <RecentlyStarted>	number	integer	The number of business services that can appear in the **Recent** section in the business service list. These are the recently started business sevices. Valid entries: 0-10
<BusinessServices> <Forms>	openForms	dock	Business service forms are docked in the Preview Pane.
		float	Business service forms are displayed in floating windows.

Element	Attribute	Possible Values	Meaning
	float	dialog	Floating windows containing a business service form are displayed in a separate dialog.
		browser	Floating windows containing a business service form are displayed in a separate browser window.
	floatLeft	integer	The floating window is positioned this number of pixels from the left. (Only applicable if both the **floatFullscreen** and **floatCenter** attributes are false.)
	floatRight	integer	The floating window is positioned this number of pixels from the top. (Only applicable if both the **floatFullscreen** and **floatCenter** attributes are false.)
	floatWidth	integer	The width (in pixels) of the floating window. (Only applicable if the **floatFullscreen** attribute is false.)
	floatHeight	integer	The height (in pixels) of the floating window. (Only applicable if the **floatFullscreen** attribute is false.)
	floatFullscreen	true	Floating windows are displayed full screen.
		false	Floating windows are not displayed full screen (other attributes specify position/ size.)

Element	Attribute	Possible Values	Meaning
<Processes> <RecentlyStarted>	number	integer	The number of processes that can appear in the **Start Process Instance** button drop-down list. This drop-down list shows the user the processes that have been started recently. Valid entry: 0-10
<Appearance> <Layout>	default	side	Lists are displayed in a side-by-side format.
		stacked	Lists are displayed in stacked (top and bottom) format.
		floating	List are displayed in floating windows.
<Appearance> <Font>	fontSize	default	The font size in the application is determined by the setting in the `workspaceCSS.xml` file.
		small	10px
		medium	12px
		large	14px

(1) The locale string must be in the form of "*ll*" or "*ll_CC*", where "*ll*" is a lowercase, two-letter ISO 639-1 language code and "CC" is an uppercase, two-letter ISO 3166 country code. For example "es" for Spanish, or "es_MX" for Spanish Mexico. For more information, see Language Resource Files .

User Preference Persistence

When using a WCC application, user preference data is always stored on the server. This includes things like view definitions, column settings, information on the Options dialog, list filter and sort settings, etc.

The **UserPreferencePersistence** parameter is used to set the maximum number of bytes for the user preference data. This value needs to be set at or below the field size supported by the database used on the server.

If this value is too small, processing the data at the server will be inefficient; if it's too large, the database will throw an exception when is attempts to parse the message containing user preference data.

Note that the character encoding used should be taken into consideration when determining the maximum data size.

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system, see Introduction).

2. Locate the **UserPreferencePersistence** record in the `config.xml` file. For example:

   ```
   <record jsxid="UserPreferencePersistence"
   type="Workspace"        maxDataSize="1992"/>
   ```

3. Set the **maxDataSize** attribute to the maximum number of bytes for the user preference data.

   Default = 1992 if attribute is absent

   Minimum value = 10

4. Save and close the `config.xml` file.

Channel Identifier

When a project is created in TIBCO Business Studio, *presentation channels* can be created for the project. Each presentation channel specifies the way in which forms that are part of the project will be presented to the user at runtime.

The Presentation Channel editor is displayed in TIBCO Business Studio by selecting **Windows** > **Preferences** > **Presentation Channels** .

There is only a single presentation channel defined in Workspace at this time:

- **GIGWTPull_DefaultChannel** - This is the default channel for Google Web Toolkit (GWT) forms.

The application's `config.xml` file contains a **channelId** parameter that identifies which presentation channel to use when displaying work item forms when using an instance of that application. (At this time, each instance of the application can be bound to a single presentation channel.)

By default, Workspace is configured to use GWT forms (the **channelId** attribute is set to **GIGWTPull_DefaultChannel** by default).

Procedure

1. Open the `config.xml` file.

2. Locate the **ChannelIdentifier** record in the `config.xml` file. For example:

   ```
   <record jsxid="ChannelIdentifier" channelId="GIGWTPull_DefaultChannel"/>
   ```

3. Set the **channelId** attribute to the value of the channelId defined in TIBCO Business Studio for the presentation channel you want to use for this Workspace.

   Note that if there is no channelId specified here, or the channelId specified does not exist, the default presentation channel (GIGWTPull_DefaultChannel) is used.

4. Save and close the `config.xml` file.

Pre-login User Access

This configuration allows you to control the access users have before logging in.

The **PreLoginUserAccess** record includes <**access**/> elements that allow you to specify the functions to which the user will have access pre-login.

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system, see Introduction).

2. Locate the **PreLoginUserAccess** record in the `config.xml` file:

```
<record jsxid="PreLoginUserAccess">
   <access name="ApplicationLog"/>
   <access name="ShowErrorDetail">
      <!--<access name="ShowStackTrace"/>-->
   </access>
   <access name="Help">
      <access name="Help"/>
      <access name="About"/>
   </access>
</record>
```

3. Configure the pre-login user access as follows:

   - **ApplicationLog** - If included, this provides access to the Application Log. The Application Log is used to troubleshoot the application. It provides detailed debug information generated by the application, as well as information about communications between the application and Action Processor. For more information, see Application Log.

   - **ShowErrorDetail** - If included, details about error conditions are displayed to the user.

   - **ShowStackTrace** - If included, a stack trace is shown when error information is displayed. This is commented out by default, as this information could be used by an attacker to gain insight into internal processes.

   - **Help** - If included, the **Help** selection is available from the **Help** button menu on the Login screen. This provides access to the application user's guide.

   - **About** - If included, the **About** selection is available from the **Help** button menu on the Login screen. This provide access to a dialog that include the software legal notice, as well as application version information.

     Note that if neither **Help** nor **About** are included, the **Help** button does not display on the Login screen.

     You can customize the **PreLoginUserAccess** record as needed:

   - include <**access**/> elements for each function to which you want all users to have access pre-login.

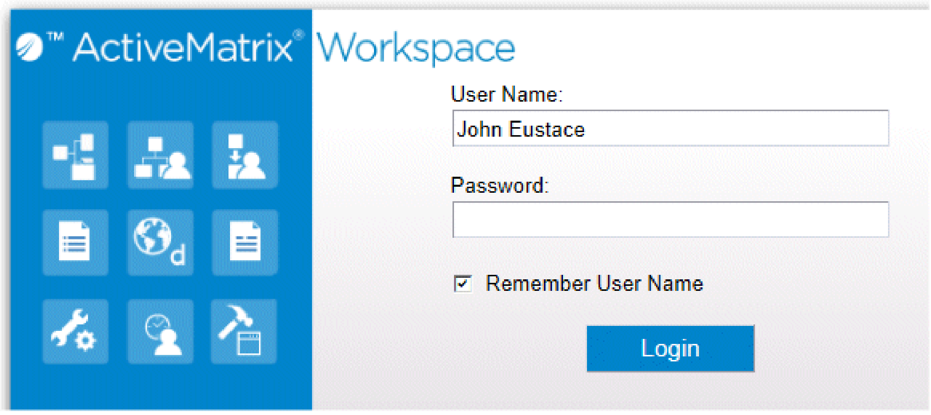
   - remove, or comment out, all <**access**/> elements for the functions to which you do *not* want users to have access pre-login.

     Once a user logs in, access authority to these specific functions can be overridden with the **ApplicationLog**, **ShowErrorDetail**, **ShowStackTrace**, **Help** > **Help** , and **Help** > **About** access entries—see Available Functions.

     Note that the pre-login access controls are not overridden by system actions.

Login

You can configure whether or not to display the **Remember User Name** check box on the Login dialog.



By default, the check box is displayed.

Procedure

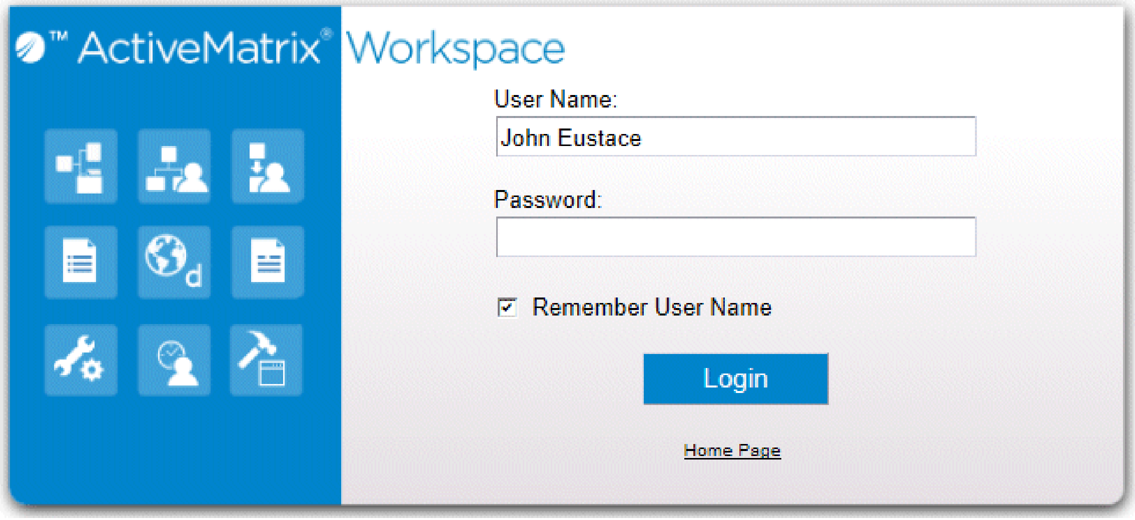
1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system, see Introduction).
2. Locate the **Login** record in the `config.xml` file:
   ```
   <record jsxid="Login" type="Workspace" useRemember="true"></record>
   ```
3. Modify the **useRemember** attribute as follows:
   - "true" causes the check box to be displayed.
   - "false" causes the check box to not be displayed.
4. Save and close the `config.xml` file.

Login Link

You can specify that a hyperlink appear on the Login dialog.

For example, on the Login dialog below, the "Home Page" link has been added:



Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system, see Introduction).

2. Locate the **LoginLink** record in the `config.xml` file:

```
<record jsxid="LoginLink" text="txtLoginLink" hoverText="txtLoginLinkHoverText"
address="" windowFeatures=""></record>
```

3. Modify the **LoginLink** attributes as follows:

   - **text** - This specifies the text that appears on the **Login** dialog—for example, “Home Page”. The value in this attribute points to a key in the `JSXAPPS/base/locale.locale.xml` file, in which the text is defined.

   - **hoverText** - This specifies the text that appears if the mouse pointer is hovered over the hyperlink. The value in this attribute points to a key in the `JSXAPPS/base/locale.locale.xml` file, in which the text is defined.

   - **address** - This specifies an HTML file path (for example, `file://C:/Welcome.html`) or a URL (for example, `http://www.tibco.com`) to open when the hyperlink is clicked.

     If this attribute contains an empty string, a hyperlink does not appear on the Login dialog.

   - **windowFeatures** - This allows you to specify various features of the window that is opened when the hyperlink is clicked. You can enter one or more of the **BrowserFeatures** attributes that are available for specifying how forms are displayed when a work item is opened. For information about the **BrowserFeatures** attributes, see Browser Features. Note that the available attributes depend on the browser being used.

     Also note that when included in the **windowsFeatures** attribute string, each of the attribute values must be enclosed in *single* quotes (not double quotes). For example:

     windowFeatures="dialog='yes' toolbar='yes'"

4. Save and close the `config.xml` file.

Forms Configuration

The **formsConfig** parameter is used to specify whether TIBCO Forms are reloaded or retrieved from cache, as well as whether or not the work item list is refreshed whenever a work item form is opened.

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system, see Introduction).

2. Locate the **formsConfig** record in the `config.xml` file. For example:

```
<record
jsxid="formsConfig"    ignoreCache="false    refreshListOnFormOpen="false"/>
```

3. Set the forms configuration parameters as follows:

   - **ignoreCache** - True = TIBCO Forms and their resources are reloaded on each request. False = TIBCO Forms and their resources are retrieved from the browser cache.

   - **refreshListOnFormOpen** - True = A full refresh of the work item list is performed whenever a work item is opened (which can have an impact on performance). False = A refresh is not performed on the work item list when a work item is opened (however, the status icon for the opened work item is changed to reflect that it is currently open).

4. Save and close the `config.xml` file.

Browser Features

The user can specify whether work item forms are opened in the Preview Pane or in a *floating* work item form. This can be set from the **View** > **Preview** menu on the work item list.

If the user chooses to display floating work item forms, he can further specify that the floating form be displayed in either a *dialog* or a *separate browser window*. This can be specified in User Options. Note, however, that the user can control whether the form is displayed in a dialog or separate browser window via User Options *only* if the **dialog** attribute in the <BrowserFeatures> record in `config.xml` is set to "no". If the **dialog** attribute is set to "yes", it forces the work item form to be displayed in a dialog. This allows you to control whether or not the user can choose a dialog or separate browser window.

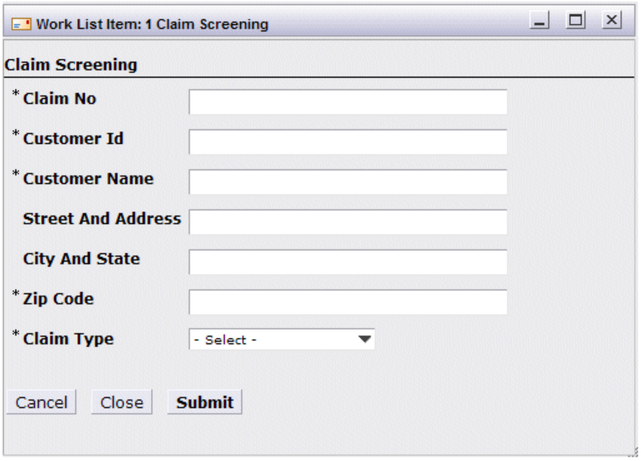
The following describes all of the attributes available in the <**BrowserFeatures**> record. They allow you to specify the behavior of the window (things such as whether the window is resizable, whether or not a status bar is displayed, etc.) when the form is opened in a separate browser window.

The following describes the differences in behavior between dialogs and separate browser windows:

- **Floating Window Outside Application Window**:
  - Separate browser windows can be floated outside the parent application's window.
  - Dialogs cannot be floated outside the parent application's window.
- **Close as child window**:
  - Separate browser windows do not close (or minimize) when the parent is closed (or minimized).
  - Dialogs are children of the parent window. Therefore if the parent window is closed (or minimized), the dialog is also closed (or minimized).
- **Browser Feature Attributes**:
  - Separate browser windows can be further controlled using the <BrowserFeatures> record in the `config.xml` file.
  - Dialogs cannot be controlled using the <BrowserFeatures> record in the `config.xml` file.

The following illustrations show you the general differences in appearance between a form in a dialog and in a separate browser window.

Work Item Form in a Dialog

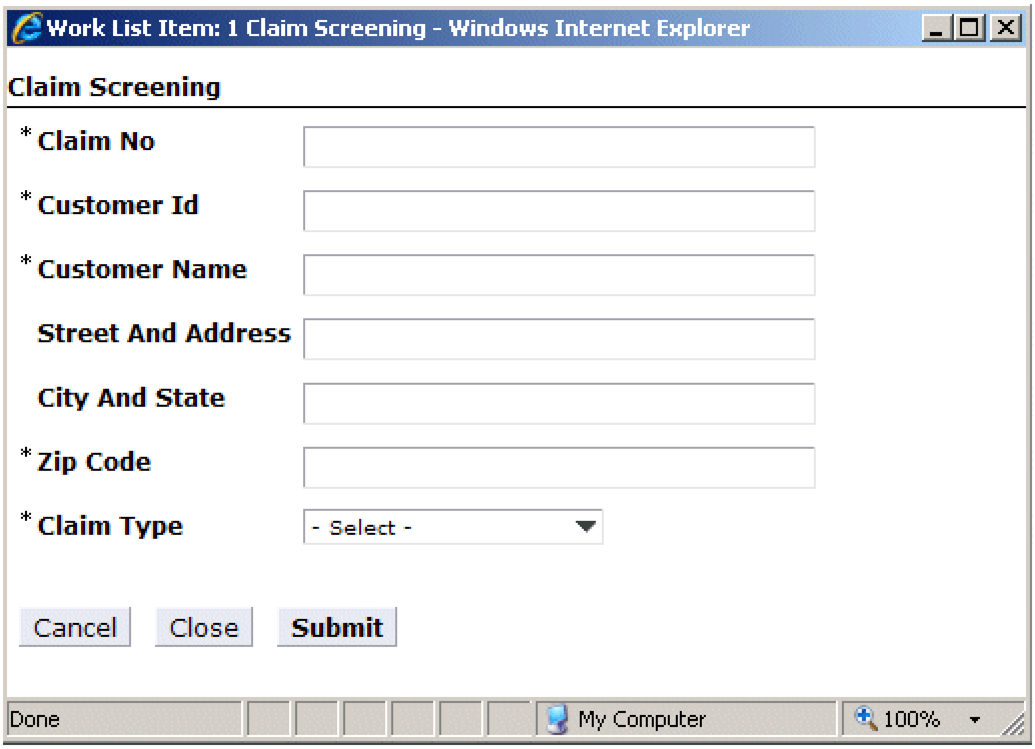


Work List Item: 1 Claim Screening

Claim Screening

* Claim No	
* Customer Id	
* Customer Name	
Street And Address	
City And State	
* Zip Code	
* Claim Type	- Select -

Work Item Form in a Separate Browser Window



Also, the extent to which you can customize the browser window appearance depends on the type of browser (Internet Explorer, Firefox, Safari, or Chrome) you are using, as shown in the table below.

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system, see Introduction).

2. Locate the <BrowserFeatures> record in the `config.xml` file:

```
<record jsxid="BrowserFeatures" type="Workspace" channelmode="no" dialog="no"
        directories="no" location="no" menubar="no" resizable="yes"
        scrollbars="yes" status="yes" toolbar="no">
</record>
```

3. Modify the desired attributes to control browser window behavior. Each attribute can be set to either “yes” or “no” to indicate whether or not to display/enable that feature.

   The following table lists the available attributes for the <BrowserFeatures> record. The **Browser** columns indicate the browsers to which the attribute applies.

	Browser				
Attribute	**IE**	**Firefox**	**Safari**	**Chrome**	**Description**
channelmode	X				Specifies whether or not to display the window in “theater mode”, that is, as a maximized window.

	Browser				
Attribute	IE	Firefox	Safari	Chrome	Description
dialog	X	X	X	X	Specifies whether or not to display the window as a “dialog”. • If **dialog** = "yes", it forces work item forms to be displayed in a dialog, regardless the setting in User Options. • If **dialog** = "no", the user can control whether or not the form is opened in a dialog or separate browser window via a setting in User Options.
directories		X			Specifies whether or not to display the "Bookmarks Toolbar" (in Firefox). Firefox users can force new windows to always render the Bookmarks Toolbar by setting `dom.disable_window_open_feature.directories` to true in **about:config** or in their **user.js** file. This setting is ignored in Internet Explorer.
location	X	X	X		Specifies whether or not to display the "Navigation Toolbar" in Internet Explorer, the "Location Bar" in Firefox, or the "Toolbar" and "Bookmarks Bar" in Safari. Firefox users can force new windows to always render the Navigation Toolbar by setting `dom.disable_window_open_feature.location` to true in **about:config** or in their **user.js** file.
menubar	X	X	X	X(1)	Specifies whether or not the browser window should display a “Menu Bar”. Firefox users can force new windows to always render the Menu Bar by setting `dom.disable_window_open_feature.menubar` to true in **about:config** or in their **user.js** file.

	Browser				
Attribute	**IE**	**Firefox**	**Safari**	**Chrome**	**Description**
resizable	X				Specifies whether or not the browser window can be manually resized using the lower right corner of the window. Note - This attribute may not work as expected. Tests on various systems has shown that on some the window can be resized, while on others, it cannot. The exact cause(s) of the unexpected behavior remains unknown, although it is thought to be a combination of the browser being used, the browser version, and browser security settings.
scrollbars	X	X		X1	Controls the display of scrollbars, on work item forms opened in a separate browser window, when the form content overflows the browser dimensions. Note - Scrollbars are always displayed, if needed, in Safari.
status	X			X1	Specifies whether or not the browser window displays a status bar on the bottom of the window. Firefox always displays the status bar.
toolbar	X	X	X	X1	Specifies whether or not to display the Toolbar across the top of the window. This bar contains buttons/icons for Back, Forward, Refresh, Home, etc. In Internet Explorer this bar is referred to as the “Command Bar”, in Firefox it’s the “Tab Bar”, and in Safari it’s the “Toolbar” and “Bookmarks Bar”. Firefox users can force new windows to always render the Tab Bar by setting `dom.disable_window_open_feature.toolbar` to true in **about:config** or in their **user.js** file.

(1) Setting menubar, scrollbars, status, or toolbar to “yes” when using Chrome causes the form to open in a new tab rather than a separate browser window.

Post-Login Caption

You can customize the caption that is displayed on the top of the browser window after a user has logged into the application.

By default, the caption is set to:

"Workspace - <*Username*>"

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system, see Introduction).

2. Locate the **postLoginCaption** record in the `config.xml` file:

   ```
   <record jsxid="postLoginCaption" pattern="%productname% - %username%"></record>
   ```

3. Modify the **pattern** attribute for the caption you would like displayed. The following placeholders can be used in the pattern string to display various information:

   - **%productname%** - This placeholder is replaced with the name of the product.
   - **%username%** - This placeholder is replaced with the name of the logged-in user.

     Any or all of the placeholders can be specified or omitted.

     If a placeholder is specified that is not available, it will be replaced with a zero-length string.

4. Save and close the `config.xml` file.

Logging

This configuration is used to control the Application Log.

For information about the Application Log, see Application Log.

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system, see Introduction).

2. Locate the **logging** record:

   ```
   <record jsxid="logging" type="Workspace"
       appLogLevel="ERROR"
       echoToJsxLog="false">
   </record>
   ```

3. Set the **logging** record's attributes as follows:

   a) Set the **appLogLevel** attribute to indicate the default value for the **Log Level** drop-down list in the Application Log. The valid entries are:

   - OFF
   - FATAL
   - ERROR
   - WARN
   - INFO
   - DEBUG
   - TRACE

   b) Set the **echoToJsxLog** attribute to indicate if the log contents should be echoed to the Application Monitor (see Application Monitor), as follows:

   - "true" causes the contents of the Application Log to be echoed to the Application Monitor.

- “false” causes the contents of the Application Log to not be echoed to the Application Monitor.

4. Save and close the `config.xml` file.

First Quarter Month

This parameter is used to specify on which of the first three months of the year a quarter begins for the business using the application.

This information is used when using the “This quarter” dynamic time period when filtering a list on a Date/Time attribute. For information about using dynamic time periods, see the “Filtering Lists” chapter in the *TIBCO Workspace User’s Guide.*

This parameter is also used if the **~THISQUARTER~** dynamic time-period variable has been used when creating custom event filters. For more information, see Event Filter String**.**

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system, see Introduction).

2. Locate the **FilterMaskQuarterStart** record in the `config.xml` file. For example:

```
<record jsxid="FilterMaskQuarterStart" value="JAN"/>
```

3. Set the **value** attribute to “JAN”, “FEB”, “or “MAR” to indicate on which of the first three months of the year a quarter begins.

   Note that this is *not* meant to indicate the month on which the *first* quarter of the year begins (although it might)—it indicates on which of those months *any* quarter begins. For example, if the company’s first quarter actually starts in December and runs through February, you would set the **value** attribute to “MAR” because that is the month in which a quarter begins in the first three months of the year.

4. Save and close the `config.xml` file.

Column Sort

This configuration controls how list contents are sorted when a column header is clicked in the lists of work items, process instances, and events.

If this parameter is set to "server", a request is sent to the server, which causes *all* available data to be sorted, including pages that have not been accessed. If set to "page", only the page that is currently displayed is sorted (a request is not sent to the server).

Note that when this parameter is set to "server," not all columns are sortable, as some attributes cannot be sorted by the server. If the user attempts to sort a non-sortable column by clicking on the header, a message is displayed.

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system), see Introduction.

2. Locate the **columnSort** record in the `config.xml` file. For example:

   ```
   <record jsxid="columnSort" value="server"/record>
   ```

3. Set the **value** attribute as follows:

- **server** - Causes *all* available data to be sorted by the values in the column, including pages that have not been accessed.
- **page** - Only the page that is currently displayed is sorted by the values in the column.

4. Save and close the `config.xml` file.

Custom Callouts

This configuration parameter specifies custom callout handlers, which allow you to *callout*, or *inject*, custom specifications in WCC applications.

For information about callout handlers in general, see the “WCC Public Methods” chapter in the *TIBCO Workspace Components Developer Guide*. And for information about specifying callout handlers using this parameter, see the “Callout Handler Configuration” section in the same chapter.

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system), see Introduction.

2. Locate the **customCallout** record in the `config.xml` file. For example:

```
<record jsxid="customCallout" type="Workspace">
    <Classes>
       <!--<Class class="com.tibco.wcc.workspace.CalloutHandler"/>-->
    </Classes>
</record>
```

3. Specify a <Class> element for each callout handler to implement. This must include the fully qualified name of the callout handler. For example:

```
<record jsxid="customCallout" type="Workspace">
    <Classes>
        <Class class="com.tibco.wcc.accounts.CalloutHandler"/>
    </Classes>
</record>
```

4. Save and close the `config.xml` file.

Getting Binary LDAP Attributes

This configuration parameter specifies whether or not LDAP attributes that contain binary data will appear in the **LDAP Attributes** list on the Organization Browser LDAP Container Editor dialog, as well as when you are mapping LDAP attributes to resource attributes. As the binary data in the LDAP attributes is base-64 encoded, binary LDAP attributes can be mapped to resource attributes of type String.

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system), see Introduction.

2. Locate the **getBinaryLDAPAttributes** record in the `config.xml` file. For example:

```
<record jsxid="getBinaryLDAPAttributes" enable="true">
</record>
```

3. Set the **enable** attribute to “true” to cause the binary LDAP attributes to appear, or “false” to not display them.

4. Save and close the `config.xml` file.

Facade Display Names

The **displayAttributeFacade** parameter is used to specify whether or not “display names” are shown for the custom attributes 1-40 in the client application. A work list facade must be deployed to the node. Work list facades are defined using TIBCO Business Studio.

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system), see Introduction.

2. Locate the **displayAttributeFacade** record in the `config.xml` file. For example:

   ```
   <record jsxid="displayAttributeFacade" useFacade="true">
   </record>
   ```

3. Set the **useFacade** attribute to “true” to cause the client application to display the attribute “display names”.

4. Save and close the `config.xml` file.

Event Paging Model

The **eventPagingModel** parameter is used to specify whether or not the total number of events is retrieved from the server and displayed in the Event Viewer. This can be used to prevent the count from being retrieved to improve performance when there are a large number of events.

If the total count is not retrieved and displayed, the user can still page through the list of events. When the end of the list is reached, a “You have reached the end of the list” message is displayed when the last page is reached.

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system), see Introduction.

2. Locate the **eventPagingModel** record in the `config.xml` file. For example:

   ```
   <record jsxid="eventPagingModel" retrieveCount="true">
   </record>
   ```

3. Set the **retrieveCount** attribute to “true” to cause the count to be retrieved and displayed in the Event Viewer. Or set it to “false” to prevent the count from being retrieved and displayed.

4. Save and close the `config.xml` file.

Authentication Mode

The **`authenticationMode`** parameter specifies the method used to authenticate users. You can use it to specify whether or not to use a current user session, or to force a Login dialog even if there is a current user session. It also allows you to specify either LDAP or single sign-on (SiteMinder or Kerberos) authentication.

This parameter can work in conjunction with the “externalLogin” URL override, which can be included in the URL when invoking Workspace. For more information about this override, see Launching a WCC Application Using an External Login.

Note that any changes made to this parameter must be made in the `config.xml` file on disk; it is not available via the Configuration Administrator. This is because this parameter configures behavior prior to login, so values stored in the database (updated via Configuration Administrator) are not accessible as the user has not logged in yet.

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system), see Introduction.

2. Locate the **authenticationMode** record in the `config.xml` file. For example:

   ```
   <record jsxid="authenticationMode" mode="useSessionByDefault" useLDAP="false">
   </record>
   ```

3. Set the **mode** and **useLDAP** attributes as follows:

   - **mode**

     – **loginByDefault** - If "externalLogin=true" is specified in the application URL, use the existing session if valid. If no valid session exists, display the Login dialog.

       If "externalLogin=false" is specified in the application URL, invalidate the session and display the Login dialog.

       If the "externalLogin" parameter is not specified in the application URL, default to invalidating the session and display the Login dialog.

     – **useSessionByDefault** - By default, use the existing session if it is valid, and do not display the Login dialog. If no valid session exists, display the Login dialog. Use this value in single sign on (SSO) implementations, such as SiteMinder or Kerberos. In SSO implementations, the session is controlled externally to the application, therefore the Login dialog is not needed.

     – **alwaysLogin** - Always invalidate an existing session and display the Login dialog.

   - **useLDAP**

     – **true** - Perform authentication using LDAP.

     – **false** - Perform authentication using single sign-on -- SiteMinder or Kerberos.

       If the server is configured to do LDAP authentication, and **useLDAP** is set to "false" in `config.xml`, clients by default will authenticate via SiteMinder or Kerberos. However, a particular invocation of the client can specify to use LDAP authentication (instead of single sign-on) by including the "ldap=true" URL override on the application URL (for more information, see Launching a WCC Application Using an External Login). If the server is not configured for LDAP authentication, any use of "ldap=true" on the URL or "useLDAP=true" in `config.xml` is ignored; in this case authentication must be via single-sign-on.

4. Save and close the `config.xml` file.

Show Logout Button

The **`showLogoutButton`** parameter controls whether or not the **Logout** button is displayed.

This parameter can be used to hide the **Logout** button in single sign on (SSO) implementations, for example, when using Kerberos, as the session is controlled externally from the application. Therefore, the **Logout** button has no meaning in that situation.

Also note that if you are hiding the **Logout** button in an SSO implementation, you should also ensure that the authenticationMode parameter's **`mode`** attribute is set to "useSessionByDefault" so that the existing external session is used, and the Login dialog is not displayed.

Procedure

1. Open the `config.xml` file.

For information about how this file should be opened (that is, via the Configuration Administrator or via the file system), see Introduction.

2. Locate the **`showLogoutButton`** record in the `config.xml` file. For example:

```
<record jsxid="showLogoutButton" showLogout="true">
</record>
```

3. Set the **`showLogout`** attribute to one of the following:

   - "true" - Displays the **Logout** button.
   - "false" - Hides the **Logout** button.

4. Save and close the `config.xml` file.

Logout Path

The **`logoutPath`** parameter controls the action taken when the **Logout** button is pressed in the application.

Procedure

1. Open the `config.xml` file.

   For information about how this file should be opened (i.e., via the Configuration Administrator or via the file system), see Introduction.

2. Locate the **logoutPath** record in the `config.xml` file. For example:

```
<record jsxid="logoutPath" path="logout">
</record>
```

3. Set the **path** attribute to one of the following:

   - "logout" - This causes the out-of-the-box logout URL to be used, which is:

     `http://`*`hostname`*`:8080/bpm/logout`

     The **`redirectURL`**[2] is then appended to the logout URL. For example, if you are running the Workspace application, the URL, `http://`*`hostname`*`:8080/workspace/workspace.html`, is appended to the logout URL. This would yield the following upon logout:

     `http://`*`hostname`*`:8080/bpm/logout?redirectURL=http://hostname:8080/workspace/workspace.html`

     Specifying "logout" in the **`path`** attribute causes the application to do a GET on the URL, which invalidates the current session. It loads a new session, then redirects to the redirectURL.

   - "*someURL*" - This works similar to "logout" except that the out-of-the-box logout URL (`http://`*`hostname`*`:8080/bpm/logout`) is not used. Instead, the custom URL specified in the **`path`** attribute is used for the logout (the custom URL can perform any custom processing that is needed (including clearing the session, if desired)). The **`redirectURL`**[1] is then appended to the custom URL. For example, if the value in the **`path`** attribute is "`http://`*`myserver`*`:90/myCustomLogout`", and you are running the Workspace application, the following URL is used upon logout:

     `http://`*`myserver`*`:90/myCustomLogout?redirectURL=http://`*`hostname`*`:8080/workspace/workspace.html`

     Upon logout, the application does a GET on this URL.

[2] Note that this redirectURL is not the same as the **`redirectURL`** attribute in the **`Application`** record, which is used when the application is being disabled. In this context, the redirectURL is the original URL used to launch the application.

The URL used to launch the application is passed in the "redirectURL" parameter. The custom URL can choose to use the redirectURL value, or not.

Also note that the custom URL must be a full URL starting with either **`http`** or **`https`**; it cannot be a relative URL.

- "" - An empty string causes the session to be invalidated and the application is reloaded. It does not do a GET on the URL.

4. Save and close the `config.xml` file.

Configuring User Access

User access is controlled in WCC applications using *system actions* and *user access controls*.

- **System actions** - These provide access control to services of components on the TIBCO ActiveMatrix BPM node. Specifying access to these services is modeled in the organization model that is built in TIBCO Business Studio.

  Access to some functions in the application is controlled by system actions (some are controlled by a single system action, some are controlled by multiple system actions, whereas some are not controlled by system actions).
- **User access controls** - These are specified at the application level. There is a user access control for most functions in the application. User access controls are grouped into *user access sets*, which allow you to give access to a group of functions defined in the user access set.

Both system actions and user access sets are privilege driven. Users gain privileges by being members of groups and positions in the organization model. Privileges can also be mapped to system actions and user access sets. Therefore, a user's privileges determine whether or not that user has access to a particular function, based on which privileges have been mapped to the system action and user access set that controls access to that function.

Note that system actions have the overriding control over access to functions, as follows:

- If a system action gives a user access to a particular function, the user access set can either give or deny access to that function.
- If a system action denies access to a particular function, the user access set cannot give access to that function.

If a user is denied access to a specific function, the UI corresponding to that function is removed from the application.

Also note that there is not a one-to-one correspondence between system actions in the organization model and the user access controls in the application. For a list of all system actions and how they map to the user access controls, see User Access Control to System Action Mapping.

For a more detailed description of system actions, see the "Security" chapter in the *BPM Concepts* guide.

Overview of User Access Authority

This topic summarizes the steps that determine user access in the application.

1. Privileges are defined in the organization model. Using the TIBCO Business Studio Organization Modeler, privileges can be assigned to groups, organization units and positions.

   For information about defining privileges in the organization model, see the *TIBCO Business Studio™ Organization Modeler User's Guide*.
2. Users are mapped to groups and positions, using the Organization Browser, thus inheriting the privileges of those entities, as follows:

   - **Groups** - Members of groups inherit the privileges of the group, as well as all parent groups.
   - **Positions / organization units** - Members of a position inherit the privileges of the position, as well as the organization unit that is the immediate parent of the position. If organization units are nested, members of the position do not inherit privileges from organization units further up the tree—only the immediate parent.

     For information about mapping users to groups and positions, see the *Organization Browser User's Guide.*
3. When the application is started, it looks at the system actions in the deployed organization model to determine which functions the user has access.

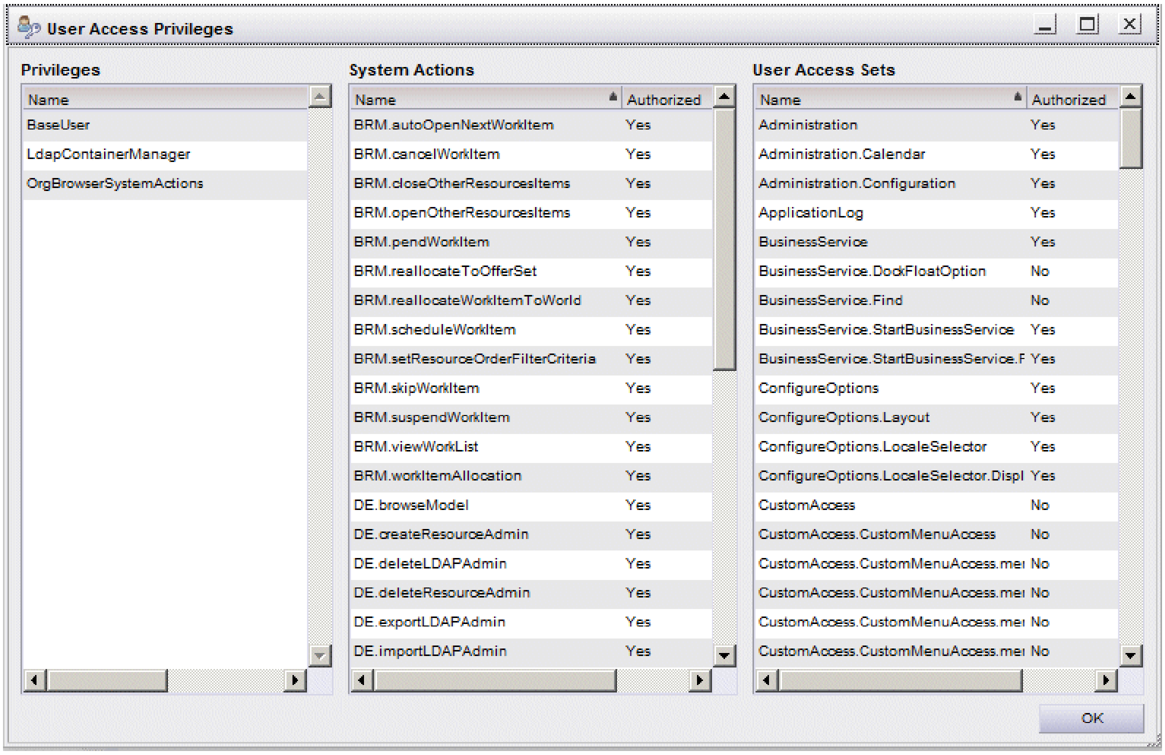
- If the system action denies access to a particular function, the user is not given access to the function, regardless which user access sets their privilege(s) are mapped.
- If the system action allows access to a particular function, it then looks at the **useAccessDefaults** attribute in the `userAccess.xml` file to determine if the user's privileges should be used to control access at the application level.
- If **useAccessDefaults** is set to "true", the default access permissions are used; see Turning Access Control On and Off.
- if **useAccessDefaults** is set to "false" user access sets containing the user's privileges are used to control access.

Determining a User's System Actions and User Access Controls

A function is provided in the Workspace application (it's also provided as a WCC component) that allows you to determine the system actions and user access controls of the currently logged-in user.

To display user access information, select **User Access Privileges** from the **Help** button menu on the Workspace application's main toolbar.

The following dialog is displayed:



Privileges

Name
BaseUser
LdapContainerManager
OrgBrowserSystemActions

System Actions

Name	Authorized
BRM.autoOpenNextWorkItem	Yes
BRM.cancelWorkItem	Yes
BRM.closeOtherResourcesItems	Yes
BRM.openOtherResourcesItems	Yes
BRM.pendWorkItem	Yes
BRM.reallocateToOfferSet	Yes
BRM.reallocateWorkItemToWorld	Yes
BRM.scheduleWorkItem	Yes
BRM.setResourceOrderFilterCriteria	Yes
BRM.skipWorkItem	Yes
BRM.suspendWorkItem	Yes
BRM.viewWorkList	Yes
BRM.workItemAllocation	Yes
DE.browseModel	Yes
DE.createResourceAdmin	Yes
DE.deleteLDAPAdmin	Yes
DE.deleteResourceAdmin	Yes
DE.exportLDAPAdmin	Yes
DE.importLDAPAdmin	Yes

User Access Sets

Name	Authorized
Administration	Yes
Administration.Calendar	Yes
Administration.Configuration	Yes
ApplicationLog	Yes
BusinessService	Yes
BusinessService.DockFloatOption	No
BusinessService.Find	No
BusinessService.StartBusinessService	Yes
BusinessService.StartBusinessService.F	Yes
ConfigureOptions	Yes
ConfigureOptions.Layout	Yes
ConfigureOptions.LocaleSelector	Yes
ConfigureOptions.LocaleSelector.Displ	Yes
CustomAccess	No
CustomAccess.CustomMenuAccess	No
CustomAccess.CustomMenuAccess.mei	No
CustomAccess.CustomMenuAccess.mei	No
CustomAccess.CustomMenuAccess.mei	No
CustomAccess.CustomMenuAccess.mei	No

This dialog displays the following columns of information:

- **Privileges** - These are the privileges held by the logged-in user, based on the groups and/or positions to which the user has been mapped.
- **System Actions** - This is an alphabetical list of all systems actions. The **Authorized** column indicates whether or not the logged-in user has that system action.
- **User Access Sets** - This is an alphabetical list of all user access controls. The **Authorized** column indicates whether or not the logged-in user has that user access control.

To determine whether or not the logged-in user has the required system action(s) and user access control needed for a particular function, see User Access Control to System Action Mapping.

Configuring User Access

User access is controlled via the `userAccess.xml` file. The way in which you modify the `userAccess.xml` file depends on whether you are configuring a deployed or non-deployed application.

- **A deployed application** - If configuring a deployed application, use the Configuration Administrator to modify the `userAccess.xml` file.

  After selecting the `userAccess.xml` file in the Configuration Administrator, you can choose to edit user access using either a graphical editor or an XML editor by clicking on the appropriate tab in the right pane. The graphical editor provides a user interface with dialogs, check boxes, etc. The XML editor allows you to directly modify the XML in the `userAccess.xml` file, for those that are comfortable with XML.

  For more information about configuring deployed applications, see Configuring and Customizing a Deployed Application.

- **A non-deployed WCC application** - If configuring a non-deployed WCC application (which might be the Workspace application), open the `userAccess.xml` file in your local development environment.

  The `userAccess.xml` file is located in the following directory:

  ```
  StudioHome\wcc\version\JSXAPPS\WCCProjectName\
  ```

  where:

  – *StudioHome* is the directory in which TIBCO Business Studio was installed.

  – *version* is the version number of Workspace that was installed with TIBCO Business Studio.

  – *WCCProjectName* is the name of the General Interface Builder project that contains your custom application. If you are working with the Workspace application, this is “workspace”.

  For more information about configuring non-deployed applications, see Configuring a Non-Deployed Application.

User Access Sets

User access sets are used to specify a set of functions that can be accessed by users who hold one or more of the privileges specified by that user access set. These are used in conjunction with *system actions* to control which functions each user can access.

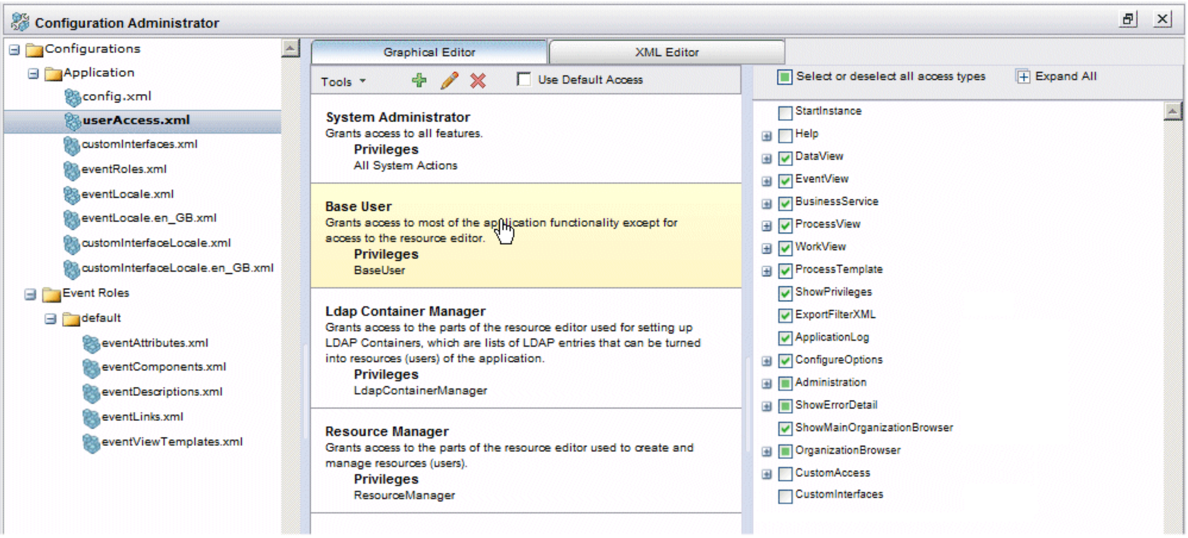
For information about system actions, see Introduction.

The way in which you create or modify a user access set depends on whether you are using the graphical editor in the Configuration Administrator, or modifying XML.

Configuring User Access Sets Using the Graphical Editor

In the Configuration Administrator graphical editor, the available user access sets are listed on the left side of the **Graphical Editor** tab.

For example:



□ Use Default Access

System Administrator
Grants access to all features.
Privileges
All System Actions

Base User
Grants access to most of the application functionality except for access to the resource editor.
Privileges
BaseUser

Ldap Container Manager
Grants access to the parts of the resource editor used for setting up LDAP Containers, which are lists of LDAP entries that can be turned into resources (users) of the application.
Privileges
LdapContainerManager

Resource Manager
Grants access to the parts of the resource editor used to create and manage resources (users).
Privileges
ResourceManager

▣ Select or deselect all access types

□ StartInstance
□ Help
☑ DataView
☑ EventView
☑ BusinessService
☑ ProcessView
☑ WorkView
☑ ProcessTemplate
☑ ShowPrivileges
☑ ExportFilterXML
☑ ApplicationLog
☑ ConfigureOptions
▣ Administration
▣ ShowErrorDetail
☑ ShowMainOrganizationBrowser
▣ OrganizationBrowser
□ CustomAccess
□ CustomInterfaces

This example shows the four default user access sets, with the “Base User” user access set selected. (Note that these four default user access sets are displayed only if the **Use Default Access** check box is unchecked—for more information, see Turning Access Control On Off Using the Graphical Editor.)

The privileges assigned to the user access set is shown under the **Privileges** heading. For example, the Base User user access set has one privilege assigned to it, BaseUser.

Users who possess a privilege listed for a particular user access set have access to functions that are selected in the check boxes listed on the right.

As you select different user access sets, the list of check boxes on the right changes to reflect the functions that are available to users with that user access set.

If a user has access to a particular function, the appropriate buttons and/or menu selections for that function are accessible in the application; if the user does not have access authority to a particular function, the buttons and/or menu selections for that function are not displayed in the application.

Creating a User Access Set Using the Graphical Editor

Click the ✚ button on the **Graphical Editor** tab, or select **New** from the **Tools** menu.

Procedure

1. On the **Add Access Set** dialog, enter a name for the user access set in the **Name** field.
2. Enter a description for the user access set in the **Description** field. This description is displayed in the list of user access sets on the **Graphical Editor** tab.
3. From the list of privileges in the right pane, select the privileges that you would like assigned to the new user access set. This list shows all privileges that are defined in your organization model. Your’s will probably be different than the one shown in this example.

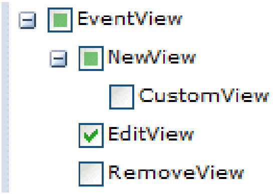
   Users who possess one of the selected privileges are given access to the functions to which this user access set provides access.
4. Click **OK**.
5. Using the check boxes in the right pane, select the functions to which you would like the new user access set to have access.

   For a description of all of the available functions, see Available Functions.

   - Double-click a check box to select or deselect it. Check boxes with a green check mark indicates access is given to that function.

- Checking a box that has subordinate boxes causes all of the subordinate boxes to be checked as well.
- Checking the **Select or deselect all access types** box on the top of the dialog causes all of the functions to become selected or deselected.

  A green square inside a check box indicates some of the subordinate entries below that box are not selected. Note, however, that the access privilege with the green box itself is selected. For example, in the following, EventView and NewView and both selected—they contain a green square because they each have an unselected child privilege:



  You can expand all of the check boxes by clicking on the **Expand All** button on the top of the dialog. If clicked, it changes to a **Collapse All** button, which can be used to collapse all of the check boxes.

  > Note that removing access to a function by deselecting a function’s check box does not *deny* the user access to the function; it just does not grant it through this user access set—the user may gain access to the function through another user access set that lists a privilege the user possesses.

6. When you are finished configuring the new user access set, click the **Apply** button to save the configuration and keep the Configuration Administrator open, or **OK** to save the configuration and close the Configuration Administrator.

Editing an Existing User Access Set Using the Graphical Editor

Procedure

1. From the **Graphical Editor** tab, select the existing user access set you would like to edit.
2. Click the ✏ button on the **Graphical Editor** tab, or select **Edit** from the **Tools** menu.
3. Optionally change the name or description of the user access set in the **Name** or **Description** fields.
4. From the list of privileges in the right pane, select the privileges that you would like assigned to the edited user access set. This list shows all privileges that are defined in your organization model. Your’s will probably be different than the one shown in this example.

   Users who possess one of the selected privileges are given access to the functions to which this user access set provides access.
5. Click **OK** to save the changes to the user access set.

Deleting a User Access Set Using the Graphical Editor

Procedure

1. Select the access set on the **Graphical Editor** tab.
2. Click the ✖ button on the **Graphical Editor** tab, or select **Remove** from the **Tools** menu.

   A confirmation dialog is displayed.

3. Respond to the confirmation.
4. Click the **Apply** button to save the configuration and keep the Configuration Administrator open, or **OK** to save the configuration and close the Configuration Administrator.

Configuring Access Sets by Editing XML

The XML in user access sets can be configured via the XML editor in the Configuration Administrator (for deployed applications) or directly in the `userAccess.xml` file on the file system (for non-deployed applications).

User access sets are specified in the `userAccess.xml` file using the <PrivilegeAccessSet/> element, as follows:

```
<PrivilegeAccessSets>
<PrivilegeAccessSet name="Loan processors">
     <privileges>
       <privilege name="LoanReviewers"/>
        <privilege name="LoanApprovers"/>
     </privileges>
     <access name="EventView">
        <access name="NewView">
            <access name="CustomView"/>
        </access>
        <access name="EditView"/>
        <access name="RemoveView"/>
        <access name="BaseFilter"/>
        <access name="CreateSystemView"/>
        <access name="AuthorSystemView"/>
        <access name="EventViewer"/>
            <access name="SaveView">
            <access name="SaveViewAs"/>
            <access name="CorrelatedEvents"/>
            <access name="EventAttributes"/>
               .
               .
               .
```

This example user access set specifies two privileges: LoanReviewers and LoanApprovers. This means that any user who has one of these privileges is given access to all of the functions listed in the <access/> elements in this user access set. Note that one of these privileges could also be listed in another user access set in the `userAccess.xml` file; users with that privilege would be given access permissions granted by the other user access set also.

Access authority can be removed from a user access set by either removing or commenting out the <access/> elements for the desired functions.

Note that removing or commenting out an <access/> element does not *deny* the user access to the function; it just does not grant it through this user access set. The user may gain access to the function through another user access set that lists a privilege the user possesses.

If a user has access authority to a particular function in the application, the appropriate buttons and/or menu selections for that function are accessible to the user; if the user does not have access authority to a particular function, the buttons and/or menu selections for that function are not displayed in the application.

Turning Access Control On and Off

The use of user access sets and privileges to control access in the application can be turned on or off.

- If access control is turned on, the application determines each user's access authority by looking at the privileges that that user possesses. It then looks in the `userAccess.xml` file to determine which user access sets contain those privileges. The user is given access to the functions listed in *every* user access set in which his privileges are listed.

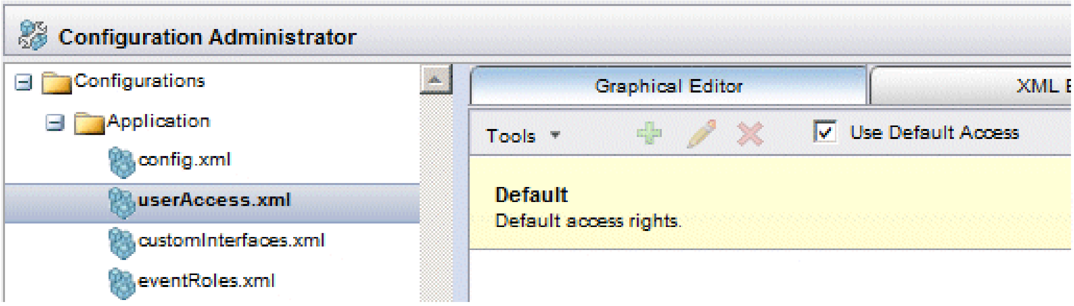
- If access control is turned off, access for all users is controlled using a list of default access permissions specified in the <AccessDefaults/> element in the `userAccess.xml` file—that is, user access sets are not used to control user access. This provides a means of controlling access for all users without regard to privileges possessed by individual users (although system actions are still enforced, even if the default user access controls are used).

The way in which you turn access control on or off depends on whether you are using the graphical editor in the Configuration Administrator, or modifying XML.

Turning Access Control On Off Using the Graphical Editor

If you are configuring a deployed application, you can use the Configuration Administrator graphical editor to turn access control on or off.

The **Use Default Access** check box on the Configuration Administrator **Graphical Editor** tab specifies access control.



☑ Use Default Access

Default
Default access rights.

- Checking the **Use Default Access** check box turns off access control. This causes the *access defaults* to be used, rather than the user’s privileges to determine access.
- Unchecking the **Use Default Access** check box turns on access control. The user’s privileges and user access sets are used to control access.

Note - Checking the **Use Access Control** box sets the **useAccessDefaults** attribute to "true" in the **UserAccess** record in the `userAccess.xml` file; unchecking the **Use Access Control** box sets the **useAccessDefaults** attribute to "false".

If the **Use Access Control** box is checked, a set of check boxes is displayed in the right pane:

These check boxes control access to functions when user access control is turned off, i.e., they define the default accesses.

For information about selecting functions in this list for default access, see step 6 in Creating a User Access Set Using the Graphical Editor.

When you are finished configuring default access, click the **Apply** button to save the configuration and keep the Configuration Administrator open, or **OK** to save the configuration and close the Configuration Administrator.

Turning Access Control On Off by Editing XML

You can modify the XML to turn access control on or off. The XML can be modified via the XML editor in the Configuration Administrator (for deployed applications) or directly in the `userAccess.xml` file on the file system (for non-deployed applications)

Access control can be turned on or off using the following record/attribute in the application’s `userAccess.xml` file:

```
<record jsxid="UserAccess" useAccessDefaults="true">
```

- If the **useAccessDefaults** attribute is set to "true" (the default), access for all users is controlled using a list of default access permissions specified in the <**AccessDefaults**/> element in the userAccess.xml file—that is, user access sets are not used to control user access (system actions still override access permissions given by the default user access controls).

- If the **useAccessDefaults** attribute is set to “false”, the application determines each user’s access authority by looking at the privileges that that user possesses. It then looks in the userAccess.xml file to determine which user access sets contain those privileges. The user is given access to the functions listed in *every* user access set in which his privileges are listed.

By default, the <**AccessDefaults**/> element includes <**access**/> entries for all available functions:

```
<AccessDefaults>
   <!--<access name="StartInstance"/>-->
<access name="DataView">
   <access name="NewView"/>
    <access name="EditView"/>
    <access name="RemoveView"/>
    <access name="NewCategory"/>
    <access name="DataViewList">
       <access name="PageSize"/>
       <access name="DataViewResults">
          <access name="GlobalDataPreview"/>
          <access name="WorkItems"/>
          <access name="ProcessInstances"/>
          <access name="EventViewer"/>
         .
         .
         .
```

You can customize the <**AccessDefaults**/> element as needed:

- include <**access**/> elements for each function to which you want all users to have access.
- remove, or comment out, all <**access**/> elements for the functions to which you do *not* want users to have access.

For a description of all of the available functions, see Available Functions.

Provided User Access Sets

By default, the userAccess.xml file contains a number of example user access sets.

They are:

- **System Administrator** - This is an example user access set that provides access to all available functionality in the application.
- **Base User** - This example user access set grants access to all available functions in the application *except* those related to using the Organization Browser (for creating/editing LDAP containers and mapping resources to the organization model).
- **LDAP Container Manager** - This example user access set grants access to only the functions in the application related to creating/editing LDAP containers using the Organization Browser.
- **Resource Manager** - This example user access set grants access to only the functions in the application related to creating and managing resources using the Organization Browser.

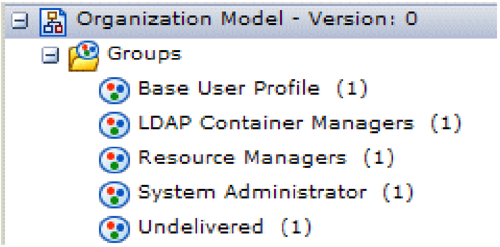
All of the example user access sets can be modified as needed by adding the appropriate privilege names, as well as adding or removing access to specific functions.

Version 0 of the Organization Model

A "Version 0" of the organization model is built into the system by default that contains a number of groups.

For more information, see the *Organization Browser User’s Guide.*

These groups are:



Organization Model - Version: 0

Groups

Base User Profile (1)

LDAP Container Managers (1)

Resource Managers (1)

System Administrator (1)

Undelivered (1)

Note that the **Base User Profile**, **LDAP Container Managers**, and **Resource Managers** groups are not used at this time**.**

The **System Administrator** and **Undelivered** groups are described below:

- **System Administrator** - Users that are mapped to this group inherit the All System Actions privilege. The All System Actions privilege is a special privilege that gives the user *all* system actions.

  The **All System Actions** privilege is also mapped to the **System Administrator** user access set in the `userAccess.xml` file, which by default provides access to all available functions in the application:

  ```
  <PrivilegeAccessSet name="System Administrator">
     <privileges>
        <privilege name="All System Actions"/>
     </privileges>
            .
            .
            .
  ```

  By default, the System Administrator user name and password are “tibco-admin” and “secret”, respectively, but may have been changed.

- **Undelivered** - This is a special group to which work items that for some reason could not be delivered to another user, are sent. The System Administrator user is a member of this group, and cannot be removed. You can add additional users to this group, if desired. Note, however, that you cannot distinguish undeliverable work items from work items that would be received because of membership in other groups or positions, so only a user who would deal with those types of work items should be mapped to this group.

Pre-Login Access

Pre-login access authority to the application is controlled using the **PreLoginUserAccess** record in the application’s configuration file, `config.xml`.

The **PreLoginUserAccess** record includes <**access**/> elements that allow you to specify the functions to which the user will have access before logging in:

```
<record jsxid="PreLoginUserAccess">
   <access name="ApplicationLog"/>
   <access name="ShowErrorDetail">
      <!--<access name="ShowStackTrace"/>-->
   </access>
   <access name="Help">
      <access name="Help"/>
      <access name="About"/>
   </access>
</record>
```

where:

- **ApplicationLog** - If included, this provides access to the Application Log. The Application Log is used to troubleshoot the application. It provides detailed debug information generated by the application, as well as information about communications between the application and Action Processor. For more information, see Application Log.

- **ShowErrorDetail** - If included, details about error conditions are displayed to the user.
- **ShowStackTrace** - If included, a stack trace is shown when error information is displayed. This is commented out by default, as this information could be used by an attacker to gain insight into internal processes.
- **Help** - If included, the **Help** selection is available from the **Help** button menu on the Login screen. This provides access to the application user's guide.
- **About** - If included, the **About** selection is available from the **Help** button menu on the Login screen. This provide access to a dialog that include the software legal notice, as well as application version information.

Note that if neither **Help** nor **About** are included, the **Help** button does not display on the Login screen.

You can customize the **PreLoginUserAccess** record as needed:

- include <**access**/> elements for each function to which you want all users to have access pre-login.
- remove, or comment out, all <**access**/> elements for the functions to which you do *not* want users to have access pre-login.

Once a user logs in, access authority to these specific functions can be overridden with the **ApplicationLog**, **ShowErrorDetail**, **ShowStackTrace**, **Help** > **Help** , and **Help** > **About** access entries—see Available Functions.

Note that the pre-login access controls are not overridden by system actions.

Available Functions

This topic lists all of the functions available through the <**access**/> elements in the `userAccess.xml` file.

User Access Control	Description
StartInstance	Provides access to the **Start Process Instance** button, which provides the ability to start an instance of a process template. Note - This is commented out by default, as it is a function that will normally not be used—process instances are normally started via a business service.
DataView	Provides access to the **Data Views** button, and the ability to display a list of data views. As of version 4.0 of Workspace, "data views" are now called "case views". However, the user access controls for case views still use the term "DataView".
DataView NewView	Provides access to the **New Data View** button and menu selection (i.e., data view wizard) on the data view list.
DataView EditView	Provides access to the **Edit Data View** button and menu selection on the data view list.

User Access Control	Description
DataView RemoveView	Provides access to the **Remove Data View** button and menu selection on the data view list.
DataView NewCategory	Provides access to the **Add Category** button and menu selection on the data view list.
DataView DataViewList	Provides access to the data view (the upper-right pane, which is entitled **Data View Results**).
DataView DataViewList PageSize	Provides access to the **View** > **Page Size** menu selection on the data view pane.
DataView DataViewList DataViewResults	Provides access to the lower-right pane that displays details about the case reference that is selected in the data view.
DataView DataViewList DataViewResults GlobalDataPreview	Provides access to the **Global Data Preview** tab when a case reference is selected in the data view.
DataView DataViewList DataViewResults WorkItems	Provides access to the **Work Items** tab when a case reference is selected in the data view.
DataView DataViewList DataViewResults ProcessInstances	Provides access to the **Process Instances** tab when a case reference is selected in the data view.
DataView DataViewList DataViewResults EventViewer	Provides access to the **Events** tab when a case reference is selected in the data view.
EventView	Provides access to the **Event Views** button, and the ability to display a list of event views.
EventView NewView	Provides access to the **New Event View** button and menu selection (i.e., event view wizard) on the event view list.

User Access Control	Description
EventView NewView CustomView	Provides access to the **Create a custom view by specifying a filter** option on the New Event View dialog. Denying access to this option forces the user to select one of the filter templates, rather than allowing a custom filter and sort for the event view.
EventView EditView	Provides access to the **Edit Event View** button and menu selection on the event view list.
EventView RemoveView	Provides access to the **Remove Event View** button and menu selection on the event view list.
EventView BaseFilter	Provides the ability to edit an event view *base filter*, which is the filter defined in the view wizard (as opposed to the *refined filter*, which is the filter specified through the Filter function on the event list). Note that this user access also affects the **Save View** function on the event list. If BaseFilter access is denied, access to the **Save View** function on the event list is also denied because the **Save View** function causes a refined filter to be added to the base filter, which you cannot do without BaseFilter authority. (Note that the **Save View** function also has its own user access control.)
EventView CreateSystemView	Provides the authority to create an event system view. If you have this user access, the **This is my own user view** and **This is a system view assigned to groups/positions or users** selections appear on the New Event View dialog when you are creating a new event view.
EventView AuthorSystemView	Provides the authority to author an event system view, that is, an event system view for which you were designated an author. Note that if you were designated an author for an event system view, and you don't have this user access, you can still edit *properties* of the event system view, such as description, filter, sort, columns, and template, via the Edit Event View function. However, you cannot edit the system view-specific elements, such as recipients, authors, and effective dates.
EventView EventViewer	Provides access to the Event Viewer (event list).
EventView EventViewer SetRetrieveCount	Provides access to the **Event Paging Model** option on the Options dialog.

User Access Control	Description
EventView EventViewer SaveView	Provides access to the **Save View** button and menu selection on the Event Viewer. Note that access to the **Save View** function is also denied if you do not have the **BaseFilter** user access control. That's because the **Save View** function causes a refined filter to be added to the base filter, which you cannot do without BaseFilter authority.
EventView EventViewer SaveViewAs	Provides access to the **Save View As** button and menu selection on the Event Viewer.
EventView EventViewer CorrelatedEvents	Provides access to correlated events links on the Event Viewer.
EventView EventViewer EventAttributes	Provides the ability to view the event attribute list in the Event Viewer.
EventView EventViewer EventLinks	Provides the ability to use event links in the Event Viewer. Note that commenting out this access attribute prevents the **Event Links** column from being displayed, although the user can add the column back into the event list (if they have permission to do that). However, if the user does this, the **Links** drop-down menu will not provide any selections.
EventView EventViewer WorkItemParticipants	Provides access to the **Work Item Participants** button and menu selection in the Event Viewer.
EventView EventViewer Filter	Provides access to the **Filter** button and menu selection in the Event Viewer.
EventView EventViewer Sort	Provides access to the **Sort** button and menu selection in the Event Viewer.
EventView EventViewer Find	Provides access to the **Find** button and menu selection in the Event Viewer.

User Access Control	Description
EventView EventViewer SelectAttributes	Provides access to the **Select Attributes** button and menu selection in the Event Viewer.
EventView EventViewer SelectColumns	Provides access to the **Select Columns** button and menu selection in the Event Viewer.
EventView EventViewer SelectColumns DefaultColumns	Provides access to the **Set as Default Columns** button and menu selection in the Event Viewer.
EventView EventViewer PageSize	Provides access to the **PageSize** menu selection in the Event Viewer.
BusinessService	Provides access to the **Business Services** button, and the ability to display a list of business services.
BusinessService StartBusinessService	Provides access to the **Start Business Service** button in the business service list.
BusinessService StartBusinessService Favorites	Provides the ability to add favorites to the business service list.
BusinessService Find	Provides access to the **Find Business Service** button and menu selection on the business service list.
BusinessService DockFloatOption	Provides access to the **Dock Forms** and **Float Forms** menu selections on the business service list.
ProcessView	Provides access to the **Process Views** button, and the ability to display a list of process views.
ProcessView ShowAllInstances	Controls whether the **All Instances** process view is displayed by default in the process view list.
ProcessView NewView	Provides access to the **New Process View** button and menu selection (i.e., process view wizard) on the process views list.

User Access Control	Description
ProcessView EditView	Provides access to the **Edit Process View** button and menu selection on the process views list.
ProcessView RemoveView	Provides access to the **Remove Process View** button and menu selection on the process views list.
ProcessView BaseFilter	Provides the ability to edit a process view *base filter*, which is the filter defined in the view wizard (as opposed to the *refined filter*, which is the filter specified through the Filter function on the process instance list). Note that this user access also affects the **Save View** function on the process instance list. If BaseFilter access is denied, access to the **Save View** function on the process instance list is also denied because the **Save View** function causes a refined filter to be added to the base filter, which you cannot do without BaseFilter authority. (Note that the **Save View** function also has its own user access control.)
ProcessView CreateSystemView	Provides the authority to create a process system view. If you have this user access, the **This is my own user view** and **This is a system view assigned to groups/positions or users** selections appear on the New Process View dialog when you are creating a new process view.
ProcessView AuthorSystemView	Provides the authority to author a process system view, that is, a process system view for which you were designated an author. Note that if you were designated an author for a process system view, and you don't have this user access, you can still edit *properties* of the process system view, such as description, filter, sort, columns, and processes, via the Edit Process View function. However, you cannot edit the system view-specific elements, such as recipients, authors, and effective dates.
ProcessView ProcessInstance	Provides access to the list of process instances for the selected process view.
ProcessView ProcessInstance DefineHaltedView	Provides access to the **Standard instance view** and **Halted instance view** selections on the process view definition dialog. This user access control is required to create halted process instance views.
ProcessView ProcessInstance ShowTerminalInstances	Provides access to the options in the process view wizard that allows you to specify that the standard instance view contain only active instances, only terminal-state instances or both active and terminal-state instances.

User Access Control	Description
ProcessView ProcessInstance SaveView	Provides access to the **Save View** button and menu selection on the process instance list. Note that access to the **Save View** function is also denied if you do not have the **BaseFilter** user access control. That's because the **Save View** function causes a refined filter to be added to the base filter, which you cannot do without BaseFilter authority.
ProcessView ProcessInstance SaveViewAs	Provides access to the **Save View As** button and menu selection on the process instance list.
ProcessView ProcessInstance Cancel	Provides access to the **Cancel** button and menu selection on the process instance list.
ProcessView ProcessInstance Resume	Provides access to the **Resume** button and menu selection on the process instance list.
ProcessView ProcessInstance Suspend	Provides access to the **Suspend** button and menu selection on the process instance list.
ProcessView ProcessInstance ResumeHalted	Provides access to the **Resume Halted Process Instance(s)** button and menu selection on the process instance list.
ProcessView ProcessInstance Retry	Provides access to the **Retry Halted Process Instance(s)** button and menu selection on the process instance list.
ProcessView ProcessInstance Ignore	Provides access to the **Ignore Fault for Halted Process Instance(s)** button and menu selection on the process instance list.
ProcessView ProcessInstance Filter	Provides access to the **Filter** button and menu selection on the process instance list.
ProcessView ProcessInstance Sort	Provides access to the **Sort** button and menu selection on the process instance list.

User Access Control	Description
ProcessView ProcessInstance Find	Provides access to the **Find** button and menu selection on the process instance list.
ProcessView ProcessInstance ShowOutstanding	Provides access to the **Show Outstanding Work Items** button and menu selection on the process instance list.
ProcessView ProcessInstance ShowOutstandingAdmin	Provides access to the **Show Supervised List of Outstanding Work Items** button and menu selection on the process instance list.
ProcessView ProcessInstance EventViewer	Provides access to the **Event Viewer** button and menu selection on the process instance list.
ProcessView ProcessInstance SelectColumns	Provides access to the **Select Columns** button and menu selection on the process instance list.
ProcessView ProcessInstance SelectColumns DefaultColumns	Provides access to the **Set as Default Columns** button and menu selection on the process instance list.
ProcessView ProcessInstance ShowCustomAttributes	Provides access to custom attributes when filtering process instances. This also provides access to the **View Details** button on the Process Instance Summry.
ProcessView ProcessInstance PageSize	Provides access to the **PageSize** menu selection on the process instance list.
WorkView	Provides access to the **Work Views** button, and the ability to display a list of work views.
WorkView ShowInbox	Controls whether the Inbox is displayed by default in the work view list.
WorkView NewView	Provides access to the **New Work View** button and menu selection (i.e., work view wizard) in the work views list.

User Access Control	Description
WorkView EditView	Provides access to the **Edit Work View** button and menu selection on the work views list.
WorkView RemoveView	Provides access to the **Remove Work View** button and menu selection on the work views list.
WorkView BaseFilter	Provides the ability to edit a work view *base filter*, which is the filter defined in the view wizard (as opposed to the *refined filter*, which is the filter specified through the Filter function on the work item list). Note that this user access also affects the **Save View** function on the work item list. If BaseFilter access is denied, access to the **Save View** function on the work item list is also denied because the **Save View** function causes a refined filter to be added to the base filter, which you cannot do without BaseFilter authority. (Note that the **Save View** function also has its own user access control.)
WorkView CreateSystemView	Provides the authority to create a work system view. If you have this user access, the **This is my own user view** and **This is a system view assigned to groups/positions or users** selections appear on the New Work View dialog when you are creating a new work view.
WorkView AuthorSystemView	Provides the authority to author a work system view, that is, a work system view for which you were designated an author. Note that if you were designated an author for a work system view, and you don't have this user access, you can still edit *properties* of the work system view, such as description, filter, sort, columns, and processes, via the Edit Work View function. However, you cannot edit the system view-specific elements, such as recipients, authors, and effective dates.
WorkView SupervisedWorkItem	Provides access to supervised work views (i.e., those displayed under **Supervised Work** in the work view list).
WorkView SupervisedWorkItem AllWorkItems	Provides the ability to create supervised work views that contain all work items irrespective of the resource or organizational entity to which the work items are offered or allocated. With this access, the **All Work Items** option appears on the New Work View - Supervised Work dialog when creating a supervised work view.
WorkView SupervisedWorkItem Cancel	Provides access to the **Cancel Work Item(s)** button and menu selection on a supervised work item list.

User Access Control	Description
WorkView SupervisedWorkItem Skip	Provides access to the **Skip Work Item(s)** button and menu selection on a supervised work item list.
WorkView SupervisedWorkItem PrioritizeAny	Provides access to the **Prioritize Work Item(s)** button and menu selection on a supervised work item list.
WorkView SupervisedWorkItem Reoffer	Provides access to the **Re-Offer Work Item(s)** button and menu selection on a supervised work item list.
WorkView SupervisedWorkItem AllocateToSelf	Provides access to the **Allocate Work Item(s) To Self** button and menu selection when a supervised work list is selected.
WorkView SupervisedWorkItem AllocateToAnother	Provides access to both the **Allocate Work Item(s) To Offer Set** and the **Allocate Work Item(s) To World** functions when a supervised work list is selected.
WorkView SupervisedWorkItem AllocateToAnother CanAllocateFromOfferSet	Provides access to the **Allocate Work Item(s) To Offer Set** function when a supervised work list is selected.
WorkView SupervisedWorkItem AllocateToAnother CanAllocateFromOrganization	Provides access to the **Allocate Work Item(s) To World** function when a supervised work list is selected.
WorkView SupervisedWorkItem AllocateToAnother ShowResourcePreview	Provides access to the **Toggle Preview** button and menu selection on both the **Allocate Work Item(s) To Offer Set** and the Allocate Work Item(s) To World dialogs when a supervised work list is selected.

User Access Control	Description
WorkView SupervisedWorkItem EventViewer	Provides access to the **Open Event Viewer** button and menu selection on a supervised work item list (that is, one displayed by selecting a work view under **Supervised Work**). Note that the user also will not have access to the **Open Event Viewer** button and menu selection if the user does not have access to event views (see the **EventView** attribute on page 19).
WorkView SupervisedWorkItem Filter	Provides access to the **Filter** button and menu selection on a supervised work item list.
WorkView SupervisedWorkItem Sort	Provides access to the **Sort** button and menu selection on a supervised work item list.
WorkView SupervisedWorkItem Find	Provides access to the **Find** button and menu selection on a supervised work item list.
WorkView SupervisedWorkItem AutoRefresh	Provides access to the **Auto-Refresh** button and menu selection on a supervised work item list.
WorkView SupervisedWorkItem SelectColumns	Provides access to the **Select Columns** button and menu selection on a supervised work item list.
WorkView SupervisedWorkItem SelectColumns DefaultColumns	Provides access to the **Set as Default Columns** button and menu selection on a supervised work item list.
WorkView SupervisedWorkItem Preview	Provides access to the **View** > **Preview** menu selection on a supervised work item list.
WorkView SupervisedWorkItem PreviewData	Provides access to the **Data Preview** tab on the preview pane for a supervised work item list.

User Access Control	Description
WorkView SupervisedWorkItem Preview PreviewOn	Provides access to the **View** > **Preview** >**Show Preview - Dock Forms in Preview Pane** menu selection on a supervised work item list.
WorkView SupervisedWorkItem Preview PreviewFloat	Provides access to the **View** > **Preview** >**Show Preview - Float Forms** menu selection on a supervised work item list.
WorkView SupervisedWorkItem Preview PreviewOff	Provides access to the **View** > **Preview** > **Preview Off - Float Forms** menu selection on a supervised work item list.
WorkView SupervisedWorkItem PageSize	Provides access to the **PageSize** menu selection on a supervised work item list.
WorkView WorkItem	Provides access to personal work views (i.e., those displayed under **My Work** in the work view list).
WorkView WorkItem SaveView	Provides access to the **Save View** button and menu selection on the work item list. Note that access to the **Save View** function is also denied if you do not have the **BaseFilter** user access control. That's because the **Save View** function causes a refined filter to be added to the base filter, which you cannot do without BaseFilter authority.
WorkView WorkItem SaveViewAs	Provides access to the **Save View As** button and menu selection on the work item list.
WorkView WorkItem Open	Provides access to the **Open Selected Work Item(s)** button and menu selection when a personal work item list is selected.
WorkView WorkItem OpenNext	Provides access to the **Open Next Work Item** button and menu selection on the work item list.

User Access Control	Description
WorkView WorkItem OpenAuto	Provides access to the **Auto-Repeat Open Work Item** button and menu selection when a personal work item list is selected. Note that if access to the WorkView/ WorkItem / OpenNext access control is denied, access to this is automatically denied also.
WorkView WorkItem Cancel	Provides access to the **Cancel Work Item(s)** button and menu selection when a personal work item list is selected.
WorkView WorkItem Skip	Provides access to the **Skip Work Item(s)** button and menu selection when a personal work item list is selected.
WorkView WorkItem Pend	Provides access to the **Pend Work Item(s)** button and menu selection when a personal work item list is selected.
WorkView WorkItem PrioritizeAllocated	Provides access to the **Prioritize Work Item(s)** button and menu selection on a personal work item list when allocated or pended work items are selected.
WorkView WorkItem PrioritizeAny	Provides access to the **Prioritize Work Item(s)** button and menu selection on a personal work item list when allocated, pended, or offered work items are selected.
WorkView WorkItem AllocateToSelf	Provides access to the **Allocate Work Item(s) To Self** button and menu selection when a personal work item list is selected.
WorkView WorkItem Reoffer	Provides access to the **Re-Offer Work Item(s)** button and menu selection when a personal work item list is selected.
WorkView WorkItem AllocateToAnother	Provides access to both the **Allocate Work Item(s) To Offer Set** and the **Allocate Work Item(s) To World** functions when a personal work list is selected.
WorkView WorkItem AllocateToAnother CanAllocateFromOfferSet	Provides access to the **Allocate Work Item(s) To Offer Set** function when a personal work list is selected.

User Access Control	Description
WorkView WorkItem AllocateToAnother CanAllocateFromOrganization	Provides access to the **Allocate Work Item(s) To World** function when a personal work list is selected.
WorkView WorkItem AllocateToAnother ShowResourcePreview	Provides access to the **Toggle Preview** button and menu selection on both the **Allocate Work Item(s) To Offer Set** and the Allocate Work Item(s) To World dialogs when a personal work list is selected.
WorkView WorkItem EventViewer	Provides access to the **Open Event Viewer** button and menu selection when a personal work list is selected. Note that the user also will not have access to the **Open Event Viewer** button and menu selection if the user does not have access to event views (see the **EventView** attribute on page 19).
WorkView WorkItem Filter	Provides access to the **Filter** button and menu selection when a personal work list is selected.
WorkView WorkItem Sort	Provides access to the **Sort** button and menu selection when a personal work list is selected.
WorkView WorkItem Find	Provides access to the **Find** button and menu selection when a personal work list is selected.
WorkView WorkItem AutoRefresh	Provides access to the **Auto-Refresh** button and menu selection when a personal work list is selected.
WorkView WorkItem SelectColumns	Provides access to the **Select Columns** button and menu selection when a personal work list is selected.
WorkView WorkItem SelectColumns DefaultColumns	Provides access to the **Set as Default Columns** button and menu selection on a personal work item list.

User Access Control	Description
WorkView WorkItem Preview	Provides access to the **View** > **Preview** menu selection on a personal work item list.
WorkView WorkItem PreviewData	Provides access to the **Data Preview** tab on the preview pane for a personal work item list.
WorkView WorkItem Preview PreviewOn	Provides access to the **View** > **Preview** >**Show Preview - Dock Forms in Preview Pane** menu selection on a personal work item list.
WorkView WorkItem Preview PreviewFloat	Provides access to the **Preview** >**Show Preview - Float Forms** menu selection on a personal work item list.
WorkView WorkItem Preview PreviewOff	Provides access to the **Preview** > **Preview Off - Float Forms** menu selection on a personal work item list.
WorkView WorkItem PageSize	Provides access to the **PageSize** menu selection on a personal work item list.
ProcessTemplate	Provides access to the list of process templates that is displayed when creating or editing a process view, or when the list of process templates is displayed via the WCC component.
ProcessTemplate Filter	Provides access to the **Filter** button and menu selection on the process template list.(1)
ProcessTemplate Sort	Provides access to the **Sort** button and menu selection on the process template list.1
ProcessTemplate Select Columns	Provides access to the **Select Columns** selection on the **View** menu on the process template list.1
ShowPrivileges	Provides access to the **User Access Privileges** selection on the **Help** menu.

User Access Control	Description
ExportFilterXML	Provides access to the **Export Filter XML** function on the Filter dialog, which is used in conjunction with callout interface methods that create or modify filters definitions in the application. For more information about this function, see the *TIBCO Workspace Components Developer Guide*.
ApplicationLog	Provides access to the Application Log that is displayed when the **F12** function key is pressed. (For more information about the Application Log, see Application Log .)
ConfigureOptions	Provides access to the **Options** button.
ConfigureOptions FontSize	Provides access to the **Font** section on the **Appearance** page of the Options dialog (which allows you to choose the font size for the application). Users who do not have access, cannot change font size, in which case, the size is determined by the **`fontSize`** setting in the **`Options`** record in `config.xml`.
ConfigureOptions Layout	Provides access to the **Layout** section on the **Appearance** page of the Options dialog (which allows you to choose the screen layout options: side-by-side, stacked, or floating windows).
ConfigureOptions WorkItemFloatOverride	Provides access to the **Work Item Forms** section on the **Work Items** page of the Options dialog, which allows the user to control work item forms in a floating window.
ConfigureOptions BusinessServiceFloatOverride	Provides access to the "Float" portion of the **Business Service Forms** section on the **Business Services** page of the Options dialog, which allows the user to control business service forms in a floating window.
ConfigureOptions LocaleSelector	Provides access to the **Language** selection field on the Options dialog.
ConfigureOptions LocaleSelector DisplayInAppHeader	Provides access to the **Language** selection field in the application header.
Administration	Provides access to the **Admin** button on the toolbar, which provides access to the Configuration Administrator.
Administration Configuration	Provides access to the **Configuration** selection on the **Admin** button drop-down list. This selection provides access to the Configuration Administrator.

User Access Control	Description
ShowErrorDetail	If enabled, details about error conditions are displayed to the user. (If error details are disabled, the text in the **txtErrorDetailsRestricted** string in `locale.xml` is displayed.)
ShowErrorDetail ShowStackTrace	If enabled, a stack trace is shown when error information is displayed. Be aware that information that is displayed when this access control is enabled could be used by an attacker to gain insight into internal processes.
ShowMainOrganizationBrowser	If enabled, the **Organization Browser** button is displayed in the application, allowing access to the Organization Browser.
OrganizationBrowser	Provides access to the Organization Browser for the purpose of performing organization model maintenance.
OrganizationBrowser ShowContainersTree	Provides access to the containers tree in the left pane of the Organization Browser.
OrganizationBrowser ShowGroupsTree	Provides access to the groups tree in the left pane of the Organization Browser.
OrganizationBrowser ShowOrganizationTree	Provides access to the organization model tree in the left pane of the Organization Browser.
OrganizationBrowser ListPotentialResources	If enabled, *potential* resources are shown in the resource list (potential resources are those that are in the resource list, but have not been created nor mapped to a group or position; they are greyed out in the resource list).
OrganizationBrowser EventViewerForOrganization	Provides access to the "**View Events**" button and menu selection in the organizational model pane in the Organization Browser.
OrganizationBrowser EventViewerForResource	Provides access to the "**View Events**" button and menu selection in the resource list pane in the Organization Browser.
OrganizationBrowser ShowResourceAttributesInResource List	Determines whether or not resource attribute columns are displayed in the resource list.
OrganizationBrowser ManageLDAPContainers	Provides access to all of the LDAP container tools in the Organization Browser.
OrganizationBrowser ManageLDAPContainers NewContainer	Provides access to the **New LDAP Container** button and menu selection in the Organization Browser.

User Access Control	Description
OrganizationBrowser ManageLDAPContainers EditContainer	Provides access to the **Edit LDAP Container** button and menu selection in the Organization Browser.
OrganizationBrowser ManageLDAPContainers DeleteContainer	Provides access to the **Delete LDAP Container** button and menu selection in the Organization Browser.
OrganizationBrowser ShowOrganizationPreview	Provides access to the details pane (lower-right pane) to view information (privileges, required capabilities, and push destinations) for the selected organizational entity.
OrganizationBrowser ShowOrganizationPreview PreviewPrivileges	Provides access to privilege information in the details pane for the selected organizational entity.
OrganizationBrowser ShowOrganizationPreview PreviewCapabilities	Provides access to capability information in the details pane for the selected organizational entity.
OrganizationBrowser ShowOrganizationPreview PreviewPushDestinations	Provides access to push destination information in the details pane for the selected organizational entity.
OrganizationBrowser ShowResourcePreview	Provides access to the details pane (lower-right pane) to view information (group membership, privileges, etc.) for the selected resource.
OrganizationBrowser ShowResourcePreview PreviewGroupMembership	Provides access to group membership information in the details pane for the selected resource.
OrganizationBrowser ShowResourcePreview PreviewPositionsHeld	Provides access to positions-held information in the details pane for the selected resource.
OrganizationBrowser ShowResourcePreview PreviewResourceAttributes	Provides access to resource attribute information in the details pane for the selected resource.
OrganizationBrowser ShowResourcePreview PreviewPrivileges	Provides access to privilege information in the details pane for the selected resource.

User Access Control	Description
OrganizationBrowser ShowResourcePreview PreviewCapabilities	Provides access to capability information in the details pane for the selected resource.
OrganizationBrowser ShowResourcePreview PreviewPushDestinations	Provides access to push destination information in the details pane for the selected resource.
OrganizationBrowser EditOrganization	Provides access to all of the edit organizational entity tools (Edit Organizational Entity Push Destinations, Import, and Export) in the Organization Browser.
OrganizationBrowser EditOrganization EditOrgPushDestinations	Provides access to all of the **Edit Organizational Entity Push Destinations** button and menu selection in the Organization Browser.
OrganizationBrowser EditOrganization Import	Provides access to all of the **Import LDAP Containers** button and menu selection in the Organization Browser.
OrganizationBrowser EditOrganization Export	Provides access to all of the **Export LDAP Containers** button and menu selection in the Organization Browser.
OrganizationBrowser EditResources	Provides access to all of the edit resource tools (Edit Groups and Positions Held, Edit Privileges, etc.) in the Organization Browser.
OrganizationBrowser EditResources EditGroupMembership	Provides access to group membership editing for the selected resource(s). If access to this function is denied, and not to "EditPositionsHeld" (see below), the Group and Position Assignment Editor dialog is still displayed when the **Edit Groups and Positions Held** function is selected. However, the buttons that allow you to grant or remove group membership for the selected resources(s) are not displayed. If access to this function *and* "EditPositionsHeld" are both denied, the **Edit Groups and Positions Held** function is not available, i.e., you cannot access the Group and Position Assignment Editor dialog.

User Access Control	Description
OrganizationBrowser EditResources EditPositionsHeld	Provides access to positions-held information editing for the selected resource(s). If access to this function is denied, and not to "EditGroupMembership" (see above), the Group and Position Assignment Editor dialog is still displayed when the **Edit Groups and Positions Held** function is selected. However, the buttons that allow you to grant or remove positions held for the selected resources(s) are not displayed. If access to this function *and* "EditGroupMembership" are both denied, the **Edit Groups and Positions Held** function is not available, i.e., you cannot access the Group and Position Assignment Editor dialog.
OrganizationBrowser EditResources EditResourceAttributes	Provides access to the **Edit Resource Attributes** button and menu selection in the Organization Browser.
OrganizationBrowser EditResources EditCapabilities	Provides access to the **Edit Capabilities** button and menu selection in the Organization Browser.
OrganizationBrowser EditResources EditPushDestinations	Provides access to the **Edit Resource Push Destination** button and menu selection in the Organization Browser.
OrganizationBrowser EditResources CreateResources	Provides access to the **Create Resources** button and menu selection in the Organization Browser.
OrganizationBrowser EditResources DeleteResources	Provides access to the **Delete Resources** button and menu selection in the Organization Browser.
OrganizationBrowser EditResources RenameResource	Provides access to the **Rename / Move Resource** menu selection in the Organization Browser.
Help	Provides access to the **Help** button on the main toolbar. Note that the **User Access Privileges** selection on the **Help** menu has its own top-level user access control (see ShowPrivileges). If you do not want the **Help** button to be displayed, you also need to remove access to the **User Access Privileges** menu selection using the **ShowPrivileges** control.

User Access Control	Description
Help Help	Provides access to the **Help** selection on the **Help** menu.
Help About	Provides access to the **About** selection on the **Help** menu.
Help OrganizationBrowser	Provides access to the **Organization Browser Help** selection on the **Help** menu.
CustomAccess(2)	This is used as a parent element to the <**CustomMenuAccess**> element, as the name of the <**CustomMenuAccess**> element may be modified if access to multiple custom menus/toolbars is configured.
CustomMenuAccess	Provides access to custom menus and toolbar buttons that have been added to the application.
CustomMenuAccess menu	Provides access to custom menus that have been added to the application.
CustomMenuAccess menu AuditEventList	Provides access to custom menus that have been added to the audit event list.
CustomMenuAccess menu BusinessServiceList	Provides access to custom menus that have been added to the business service list.
CustomMenuAccess menu EventViewList	Provides access to custom menus that have been added to the event view list.
CustomMenuAccess menu MainAppToolbar	Provides access to custom menus that have been added to the main application toolbar. (Note that this is applicable only to the Workspace application. Custom menus and toolbar buttons cannot be added to the main toolbar in a custom application.)
CustomMenuAccess menu ProcessInstanceList	Provides access to custom menus that have been added to the process instance list.
CustomMenuAccess menu ProcessTemplateList	Provides access to custom menus that have been added to the process template list.

User Access Control	Description
CustomMenuAccess menu ProcessViewList	Provides access to custom menus that have been added to the process view list.
CustomMenuAccess menu WorkItemList	Provides access to custom menus that have been added to the work item list.
CustomMenuAccess menu WorkViewList	Provides access to custom menus that have been added to the work view list.
CustomMenuAccess toolbar	Provides access to custom toolbar buttons that have been added to the application.
CustomMenuAccess toolbar AuditEventList	Provides access to custom toolbar buttons that have been added to the audit event list.
CustomMenuAccess toolbar BusinessServiceList	Provides access to custom toolbar buttons that have been added to the business service list.
CustomMenuAccess toolbar EventViewList	Provides access to custom toolbar buttons that have been added to the event view list.
CustomMenuAccess toolbar MainAppToolbar	Provides access to custom toolbar buttons that have been added to the main application toolbar. (Note that this is applicable only to the Workspace application. Custom menus and toolbar buttons cannot be added to the main toolbar in a custom application.)
CustomMenuAccess toolbar ProcessInstanceList	Provides access to custom toolbar buttons that have been added to the process instance list.
CustomMenuAccess toolbar ProcessTemplateList	Provides access to custom toolbar buttons that have been added to the process template list.

User Access Control	Description
CustomMenuAccess toolbar ProcessViewList	Provides access to custom toolbar buttons that have been added to the process view list.
CustomMenuAccess toolbar WorkItemList	Provides access to custom toolbar buttons that have been added to the work item list.
CustomMenuAccess toolbar WorkViewList	Provides access to custom toolbar buttons that have been added to the work view list.
CustomInterfaces	Provides access to toolbar buttons, menus, and menu items added to the Workspace application using custom interfaces. This user access control will contain subordinate entries for each toolbar button, menu, and menu item that has been defined. For more information, see User Access Elements for Custom Interfaces .

(1) Only applicable when using the Process Templates component in a custom WCC application.

(2) For additional information about controlling access to custom menus and toolbar buttons, see Controlling User Access to Custom Menus and Toolbar Buttons .

User Access Control to System Action Mapping

This topic lists each of the user access controls, and all of the system actions that you need to have access to the function provided by the user access control.

Select the user access control name in the table to link to a description of the function.

The **DE.userAdmin** system action does not control access to a function like other system actions. Although its name implies it has to do with user administration, it doesn't. It is used to control whether or not the application will persist user settings. If the user does not have this system action, user settings (e.g., any views that are created, columns that are changed, any changes made via the User Options dialog, etc.) are not persisted. The changes can be made by the user (if they have the appropriate access to make the changes), but they will persist only for the current login session.

Note that the DE.userAdmin system action also controls where the application reads configuration information (including user access):

- If the user has the DE.userAdmin system action, configuration information (including user access) is read from the database (if the Configuration Administrator had previously been used, which causes configuration to be written to the database).
- If the user does not have the DE.userAdmin system action, configuration information (including user access) is always read from disk (even if the Configuration Adminstrator had previously been used).

Some functions do not require a system action (e.g., all of the event view functions).

If the user is mapped to the **System Administrator** group in the version 0 organization model (i.e., the user has the "All System Actions" privilege), the application does not check the system actions—the user is given access to all system actions. For more information, see Provided User Access Sets.

Note that the system action names provided in the table below are the names shown in the User Access Privileges dialog that is available from the application (see Determining a User’s System Actions and User Access Controls). However, these are *not* the labels that are shown for the system actions in TIBCO Business Studio—those labels are not available to the application. They are, however, very similar. For example, the BRM.autoOpenNextWorkItem system action is labelled “Auto Open Next Work item” in Business Studio, the DE.LDAPAdmin system action is labelled “LDAP Admin” in Business Studio, and so on. One notable exception is the DE.userAdmin system action, which is labelled “User Settings” in Business Studio.

User Access Control	System Actions Needed
StartInstance	None
DataView	BDS.readGlobalData
DataView NewView	BDS.manageDataViews, BDS.readGlobalData
DataView EditView	BDS.manageDataViews, BDS.readGlobalData
DataView RemoveView	BDS.manageDataViews, BDS.readGlobalData
DataView NewCategory	BDS.manageDataViews, BDS.readGlobalData
DataView DataViewList	BDS.readGlobalData
DataView DataViewList PageSize	BDS.readGlobalData
DataView DataViewList DataViewResults	BDS.readGlobalData
DataView DataViewList DataViewResults GlobalDataPreview	BDS.readGlobalData
DataView DataViewList DataViewResults WorkItems	BRM.viewGlobalWorkList, BDS.readGlobalData

User Access Control	System Actions Needed
DataView DataViewList DataViewResults ProcessInstances	BDS.readGlobalData PE.queryProcessTemplate PE.queryProcessInstance
DataView DataViewList DataViewResults EventViewer	EC.queryAudit, BDS.readGlobalData
EventView	EC.queryAudit
EventView NewView	EC.queryAudit
EventView NewView CustomView	EC.queryAudit
EventView EditView	EC.queryAudit
EventView RemoveView	EC.queryAudit
EventView BaseFilter	EC.queryAudit
EventView CreateSystemView	EC.queryAudit
EventView AuthorSystemView	EC.queryAudit
EventView EventViewer	EC.queryAudit

User Access Control	System Actions Needed
EventView EventViewer SaveView	EC.queryAudit
EventView EventViewer SetRetrieveCount	*None*
EventView EventViewer SaveViewAs	EC.queryAudit
EventView EventViewer CorrelatedEvents	EC.queryAudit
EventView EventViewer EventAttributes	EC.queryAudit
EventView EventViewer EventLinks	EC.queryAudit
EventView EventViewer Filter	EC.queryAudit
EventView EventViewer Sort	EC.queryAudit
EventView EventViewer Find	EC.queryAudit

User Access Control	System Actions Needed
EventView EventViewer SelectAttributes	EC.queryAudit
EventView EventViewer SelectColumns	EC.queryAudit
EventView EventViewer SelectColumns DefaultColumns	EC.queryAudit
EventView EventViewer PageSize	EC.queryAudit
BusinessService	BIZSVC.listBusinessService(1)
BusinessService StartBusinessService	BIZSVC.listBusinessService(1) BIZSVC.executeBusinessService
BusinessService StartBusinessService Favorites	BIZSVC.listBusinessService(1) BIZSVC.executeBusinessService
BusinessService Find	BIZSVC.listBusinessService(1)
BusinessService DockFloatOption	BIZSVC.listBusinessService(1)
ProcessView	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView ShowAllInstances	PE.queryProcessTemplate PE.queryProcessInstance

User Access Control	System Actions Needed
ProcessView NewView	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView EditView	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView RemoveView	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView BaseFilter	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView CreateSystemView	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView AuthorSystemView	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView ProcessInstance	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView ProcessInstance DefineHaltedView	PE.queryProcessTemplate PE.queryProcessInstance PE.haltedProcessAdministration
ProcessView ProcessInstance ShowTerminalInstances	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView ProcessInstance SaveView	PE.queryProcessTemplate PE.queryProcessInstance

User Access Control	System Actions Needed
ProcessView ProcessInstance SaveViewAs	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView ProcessInstance Cancel	PE.queryProcessTemplate PE.queryProcessInstance PE.cancelProcessInstance
ProcessView ProcessInstance Resume	PE.queryProcessTemplate PE.queryProcessInstance PE.resumeProcessInstance
ProcessView ProcessInstance Suspend	PE.queryProcessTemplate PE.queryProcessInstance PE.suspendProcessInstance
ProcessView ProcessInstance ResumeHalted	PE.queryProcessTemplate PE.queryProcessInstance PE.haltedProcessAdministration
ProcessView ProcessInstance Retry	PE.queryProcessTemplate PE.queryProcessInstance PE.haltedProcessAdministration
ProcessView ProcessInstance Ignore	PE.queryProcessTemplate PE.queryProcessInstance PE.haltedProcessAdministration

User Access Control	System Actions Needed
ProcessView ProcessInstance Filter	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView ProcessInstance Sort	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView ProcessInstance Find	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView ProcessInstance ShowOutstanding	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView ProcessInstance ShowOutstandingAdmin	PE.queryProcessTemplate PE.queryProcessInstance DE.browseModel
ProcessView ProcessInstance EventViewer	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView ProcessInstance SelectColumns	PE.queryProcessTemplate PE.queryProcessInstance

User Access Control	System Actions Needed
ProcessView ProcessInstance SelectColumns DefaultColumns	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView ProcessInstance PageSize	PE.queryProcessTemplate PE.queryProcessInstance
ProcessView ProcessInstance ShowCustomAttributes	PE.queryProcessTemplate PE.queryProcessInstance
WorkView	None
WorkView ShowInbox	None
WorkView NewView	None
WorkView EditView	None
WorkView RemoveView	None
WorkView BaseFilter	None
WorkView CreateSystemView	None
WorkView AuthorSystemView	None

User Access Control	System Actions Needed
WorkView SupervisedWorkItem	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check(2))
WorkView SupervisedWorkItem AllWorkItems	DE.browseModel DE.resolveResource BRM.viewGlobalWorkList
WorkView SupervisedWorkItem Cancel	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2) BRM.closeOtherResourcesItems (scope check2) Note - This function is available only from a supervised work view for a resource; it is not available from a supervised work view for an organizational entity.
WorkView SupervisedWorkItem Skip	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2) BRM.skipWorkItem (scope check2) Note - This function is available only from a supervised work view for a resource; it is not available from a supervised work view for an organizational entity.
WorkView SupervisedWorkItem PrioritizeAny	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2) BRM.changeAnyWorkItemPriority (scope check2)

User Access Control	System Actions Needed
WorkView SupervisedWorkItem Reoffer	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2) BRM.workItemAllocation (You must have this system action at the organization model level, as the one at the scoped level is not used.) Note - This function is available only from a supervised work view for a resource; it is not available from a supervised work view for an organizational entity.
WorkView SupervisedWorkItem AllocateToAnother	n/a See the following two rows in this table for the system actions required for the work item allocation functions.
WorkView SupervisedWorkItem AllocateToAnother CanAllocateFromOfferSet	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2) BRM.reallocateToOfferSet (scope check2) BRM.workItemAllocation (You must have this system action at the organization model level, as the one at the scoped level is not used.)
WorkView SupervisedWorkItem AllocateToAnother CanAllocateFromOrganization	DE.browseModel DE.resolveResource DE.resourceAdmin BRM.viewWorkList (scope check2) BRM.reallocateWorkItemToWorld (You must have this system action at the organization model level, as the one at the scoped level is not used.) BRM.workItemAllocation (You must have this system action at the organization model level, as the one at the scoped level is not used.) Note - This function is available only from a supervised work view for a resource; it is not available from a supervised work view for an organizational entity. Also note that the function this controls is actually named Allocate to World.

User Access Control	System Actions Needed
WorkView SupervisedWorkItem AllocateToAnother ShowResourcePreview	n/a *There are no system actions that control access to the* **Toggle Preview** *button / menu selection.*
WorkView SupervisedWorkItem EventViewer	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2)
WorkView SupervisedWorkItem Filter	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2)
WorkView SupervisedWorkItem Sort	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2)
WorkView SupervisedWorkItem Find	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2)
WorkView SupervisedWorkItem AutoRefresh	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2)
WorkView SupervisedWorkItem SelectColumns	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2)

User Access Control	System Actions Needed
WorkView SupervisedWorkItem DefaultColumns	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2)
WorkView SupervisedWorkItem Preview	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2)
WorkView SupervisedWorkItem PreviewData	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2)
WorkView SupervisedWorkItem Preview PreviewOn	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2)
WorkView SupervisedWorkItem Preview PreviewFloat	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2)
WorkView SupervisedWorkItem Preview PreviewOff	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2)
WorkView SupervisedWorkItem PageSize	DE.browseModel DE.resolveResource BRM.viewWorkList (scope check2)
WorkView WorkItem	DE.resolveResource
WorkView WorkItem SaveView	DE.resolveResource

User Access Control	System Actions Needed
WorkView WorkItem SaveViewAs	DE.resolveResource
WorkView WorkItem Open	DE.resolveResource
WorkView WorkItem OpenNext	DE.resolveResource BRM.autoOpenNextWorkItem
WorkView WorkItem OpenAuto	DE.resolveResource BRM.autoOpenNextWorkItem
WorkView WorkItem Cancel	DE.resolveResource
WorkView WorkItem Skip	DE.resolveResource BRM.skipWorkItem
WorkView WorkItem Pend	DE.resolveResource BRM.pendWorkItem

User Access Control	System Actions Needed
WorkView WorkItem PrioritizeAllocated	DE.resolveResource BRM.changeAllocatedWorkItemPriority
WorkView WorkItem PrioritizeAny	DE.resolveResource BRM.changeAnyWorkItemPriority
WorkView WorkItem AllocateToSelf	DE.resolveResource
WorkView WorkItem Reoffer	DE.resolveResource BRM.workItemAllocation
WorkView WorkItem AllocateToAnother	DE.resolveResource
WorkView WorkItem AllocateToAnother CanAllocateFromOfferSet	DE.browseModel DE.resolveResource BRM.workItemAllocation
WorkView WorkItem AllocateToAnother CanAllocateFromOrganization	DE.browseModel DE.resolveResource DE.resourceAdmin BRM.workItemAllocation BRM.reallocateWorkItemToWorld

User Access Control	System Actions Needed
WorkView WorkItem AllocateToAnother ShowResourcePreview	DE.resolveResource
WorkView WorkItem EventViewer	DE.resolveResource
WorkView WorkItem Filter	DE.resolveResource
WorkView WorkItem Sort	DE.resolveResource
WorkView WorkItem Find	DE.resolveResource
WorkView WorkItem AutoRefresh	DE.resolveResource
WorkView WorkItem SelectColumns	DE.resolveResource

User Access Control	System Actions Needed
WorkView WorkItem DefaultColumns	DE.resolveResource
WorkView WorkItem Preview	DE.resolveResource
WorkView WorkItem PreviewData	DE.resolveResource
WorkView WorkItem Preview PreviewOn	DE.resolveResource
WorkView WorkItem Preview PreviewFloat	DE.resolveResource
WorkView WorkItem Preview PreviewOff	DE.resolveResource
WorkView WorkItem PageSize	DE.resolveResource
ProcessTemplate	PE.queryProcessTemplate
ProcessTemplate Filter	PE.queryProcessTemplate
ProcessTemplate Sort	PE.queryProcessTemplate

User Access Control	System Actions Needed
ProcessTemplate Select Columns	PE.queryProcessTemplate
ShowPrivileges	None
ExportFilterXML	None
ApplicationLog	None
ConfigureOptions	DE.userAdmin (also see note on page 41)
ConfigureOptions Layout	DE.userAdmin (also see note on page 41)
ConfigureOptions WorkItemFloatOverride	*None*
ConfigureOptions BusinessServiceFloatOverride	*None*
ConfigureOptions LocaleSelector	*None*
ConfigureOptions LocaleSelector DisplayInAppHeader	*None*
Administration	None
Administration Configuration	DE.userAdmin WSB.applicationConfiguration
ShowErrorDetail	None
ShowErrorDetail ShowStackTrace	None
ShowMainOrganizationBrowser	None
OrganizationBrowser	DE.browseModel

User Access Control	System Actions Needed
OrganizationBrowser ShowContainersTree	DE.browseModel DE.LDAPAdmin
OrganizationBrowser ShowGroupsTree	DE.browseModel
OrganizationBrowser ShowOrganizationTree	DE.browseModel
OrganizationBrowser ListPotentialResources	DE.browseModel DE.LDAPAdmin
OrganizationBrowser EventViewerForOrganization	DE.browseModel
OrganizationBrowser EventViewerForResource	DE.browseModel And one of the following is needed to see a resource: • DE.LDAPAdmin (indirectly needed to list resources from an LDAP Container) • DE.resourceAdmin or DE.resolveResource (indirectly needed to list resources in a group or position)
OrganizationBrowser ShowResourceAttributesInResourceList	DE.browseModel DE.readParameters And one of these is required to see a resource: • DE.LDAPAdmin (indirectly needed to list resources from an LDAP Container) • DE.resourceAdmin or DE.resolveResource (indirectly needed to list resources in a group or position)
OrganizationBrowser ManageLDAPContainers	n/a This is never directly checked.

User Access Control	System Actions Needed
OrganizationBrowser ManageLDAPContainers NewContainer	DE.browseModel DE.LDAPAdmin
OrganizationBrowser ManageLDAPContainers EditContainer	DE.browseModel DE.LDAPAdmin
OrganizationBrowser ManageLDAPContainers DeleteContainer	DE.browseModel DE.LDAPAdmin DE.deleteLDAPAdmin DE.deleteResourceAdmin(3)
OrganizationBrowser ShowOrganizationPreview	DE.browseModel
OrganizationBrowser ShowOrganizationPreview PreviewPrivileges	DE.browseModel
OrganizationBrowser ShowOrganizationPreview PreviewCapabilities	DE.browseModel
OrganizationBrowser ShowOrganizationPreview PreviewPushDestinations	DE.browseModel DE.readPushDestinations

User Access Control	System Actions Needed
OrganizationBrowser ShowResourcePreview	DE.browseModel DE.resolveResource And if you are performing this function via the list of resources in an LDAP container, you need the following system action to view the resources in the container: - DE.LDAPAdmin
OrganizationBrowser ShowResourcePreview PreviewGroupMembership	DE.browseModel DE.resolveResource And if you are performing this function via the list of resources in an LDAP container, you need the following system action to view the resources in the container: - DE.LDAPAdmin
OrganizationBrowser ShowResourcePreview PreviewPositionsHeld	DE.browseModel DE.resolveResource And if you are performing this function via the list of resources in an LDAP container, you need the following system action to view the resources in the container: - DE.LDAPAdmin
OrganizationBrowser ShowResourcePreview PreviewResourceAttributes	DE.browseModel DE.resolveResource DE.readParameters And if you are performing this function via the list of resources in an LDAP container, you need the following system action to view the resources in the container: - DE.LDAPAdmin
OrganizationBrowser ShowResourcePreview PreviewPrivileges	DE.browseModel DE.resolveResource And if you are performing this function via the list of resources in an LDAP container, you need the following system action to view the resources in the container: - DE.LDAPAdmin

User Access Control	System Actions Needed
OrganizationBrowser ShowResourcePreview PreviewCapabilities	DE.browseModel DE.resolveResource And if you are performing this function via the list of resources in an LDAP container, you need the following system action to view the resources in the container: • DE.LDAPAdmin
OrganizationBrowser ShowResourcePreview PreviewPushDestinations	DE.browseModel DE.resolveResource DE.readPushDestinations And if you are performing this function via the list of resources in an LDAP container, you need the following system action to view the resources in the container: • DE.LDAPAdmin
OrganizationBrowser EditOrganization	n/a This is never directly checked.
OrganizationBrowser EditOrganization EditOrgPushDestinations	DE.browseModel DE.readPushDestinations DE.writePushDestinations
OrganizationBrowser EditOrganization Import	DE.browseModel DE.importLDAPAdmin
OrganizationBrowser EditOrganization Export	DE.browseModel DE.exportLDAPAdmin
OrganizationBrowser EditResources	n/a This is never directly checked.

User Access Control	System Actions Needed
OrganizationBrowser EditResources EditGroupMembership	DE.browseModel DE.resolveResource DE.resourceAdmin And if you are performing this function via the list of resources in an LDAP container, you need the following system action to view the resources in the container: - DE.LDAPAdmin
OrganizationBrowser EditResources EditPositionsHeld	DE.browseModel DE.resolveResource DE.resourceAdmin And if you are performing this function via the list of resources in an LDAP container, you need the following system action to view the resources in the container: - DE.LDAPAdmin
OrganizationBrowser EditResources EditResourceAttributes	DE.browseModel DE.readParameters DE.writeParameters And one of these is required to see a resource: - DE.LDAPAdmin (indirectly needed to list resources from an LDAP Container) - DE.resourceAdmin or DE.resolveResource (indirectly needed to list resources in a group or position)
OrganizationBrowser EditResources EditCapabilities	DE.browseModel DE.resourceAdmin And one of these is required to see a resource: - DE.LDAPAdmin (indirectly needed to list resources from an LDAP Container) - DE.resourceAdmin or DE.resolveResource (indirectly needed to list resources in a group or position)

User Access Control	System Actions Needed
OrganizationBrowser EditResources EditPushDestinations	DE.browseModel DE.readPushDestinations DE.writePushDestinations And one of these is required to see a resource: • DE.LDAPAdmin (indirectly needed to list resources from an LDAP Container) • DE.resourceAdmin or DE.resolveResource (indirectly needed to list resources in a group or position)
OrganizationBrowser EditResources CreateResources	DE.browseModel DE.createResourceAdmin DE.LDAPAdmin (indirectly needed—required to view lists of resources for an LDAP container, the only place where potential resources will exist)
OrganizationBrowser EditResources DeleteResources	DE.browseModel DE.deleteResourceAdmin One of these is required to see a resource: • DE.LDAPAdmin (indirectly needed to list resources from an LDAP Container) • DE.resourceAdmin or DE.resolveResource (indirectly needed to list resources in a group or position)
OrganizationBrowser EditResources RenameResource	DE.browseModel DE.resolveResource DE.resourceAdmin DE.LDAPAdmin
Help	None
Help Help	None
Help About	None

User Access Control	System Actions Needed
Help OrganizationBrowser	None
CustomMenuAccess	None
CustomInterfaces	None (Note, however, that you may need a system action for the location at which the custom interface launch control appears. For example, if a custom interface menu appears on the work item list, you need the appropriate system to access the work item list.)

(1) Although you need the **BIZSVC.listBusinessService** system action to list business services, and the **BIZSVC.executeBusinessService** system action to execute business services, you might need other system actions for your business services to execute correctly. For example, if your business service creates statefull instances of processes, you will also need the **PE.startprocess** system action. If that instance then creates a user task (work item) for the first step, you would also need the **BRM.scheduleWorkItem** system action.

(2) A scope check means that it checks to see if the system action is set on a specific group, organization unit, or position (for the purpose of providing access to supervised work views). If the system action is not set on a *scoped* level, it checks to see if it is set at the organization model level. For more information, see Scope of System Actions .

(3) This system action is needed to delete an LDAP container that contains resources.

The following three event-related system actions are also available from TIBCO Business Studio, although they are not currently used. Therefore, they are not shown in the table above:

- EC.openWorkItemAuditTrail
- EC.listTrocessTemplateAuditTrail
- EC.showProcessInstanceAuditTrail

There is also a **DE.organizationAdmin** system action available that does not have a corresponding user access control. This system action overrides container organization relationships as defined in the Organization Browser. Resources with this system action can see all organizations, regardless the organization relationships that have been defined.

Scope of System Actions

When system actions are configured in the organization model using TIBCO Business Studio, they are configured at the *organization model level* or they are *scoped* (that is, they are configured at the group, organization unit, or position level).

- **Organization model level** - Most system actions are configured at this level. They control access to most functions in the application, based on privileges held by the logged-in user.

  For more information, see System Actions at the Organization Model Level.

- **Group, organization unit, or position level** - There are also a number of system actions that are *scoped*, that is, they can be set on specific groups, organization units, and positions.

  The scoped system actions are specifically used to control access to, and functions from, supervised work views.

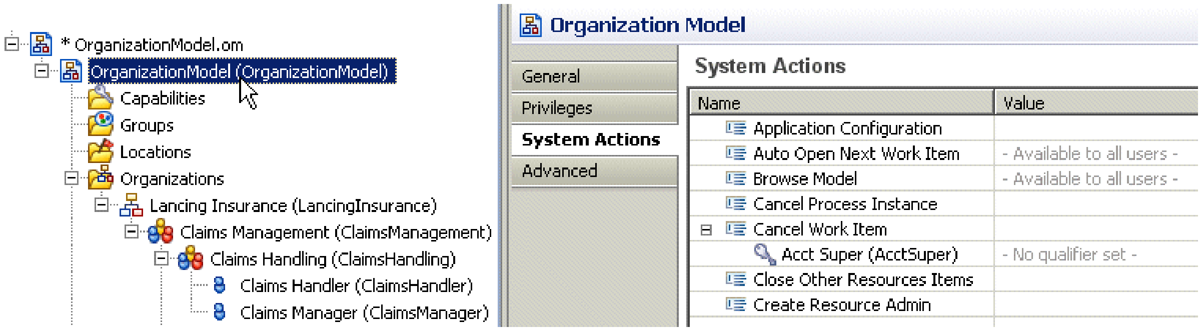
  For more information, see System Actions at the Group Organization Unit and Position Level.

System Actions at the Organization Model Level

System actions at the organization model level control access to most functions in the application, based on privileges held by the logged-in user.

System actions at the organization model level are defined in TIBCO Business Studio by selecting the organization model, then associating a privilege to the appropriate system action.

In the following example of system actions configured at the organization model level, users that have the "Acct Super" privilege can cancel work items (assuming they have the needed user access control in `userAccess.xml`).


* OrganizationModel.om
OrganizationModel (OrganizationModel)
Capabilities
Groups
Locations
Organizations
Lancing Insurance (LancingInsurance)
Claims Management (ClaimsManagement)
Claims Handling (ClaimsHandling)
Claims Handler (ClaimsHandler)
Claims Manager (ClaimsManager)
Organization Model
General
Privileges
System Actions
Advanced
System Actions

Name	Value
Application Configuration	
Auto Open Next Work Item	- Available to all users -
Browse Model	- Available to all users -
Cancel Process Instance	
Cancel Work Item	
Acct Super (AcctSuper)	- No qualifier set -
Close Other Resources Items	
Create Resource Admin	

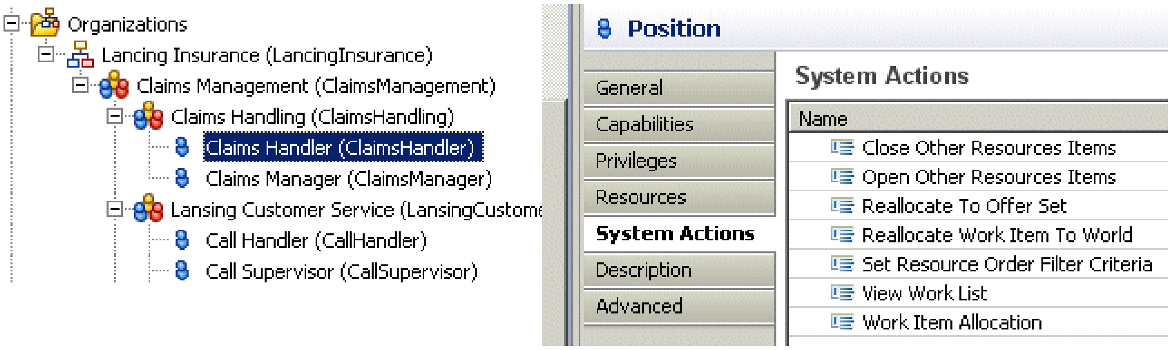
At the organization model level, some system actions are granted to all users by default (those that contain "Available to all users" in the **Value** column), and some are denied by default (those that have an empty **Value** column). If a privilege has been assigned to a system action (such as the Cancel Work Item system action), it overrides the default; that is, once a privilege is assigned to a system action, only users that have that privilege can perform the associated function (such as cancelling work items).

All of the system actions listed in the table in User Access Control to System Action Mapping can be configured at the organization model level.

System Actions at the Group Organization Unit and Position Level

To be able to create a supervised work view in the application, and to use some of the functionality available from the supervised work view, you must have the appropriate *scoped* system actions, that is, system actions configured on an individual group, organization unit, or position.

System actions at the group, organization unit, and position level are defined in TIBCO Business Studio by selecting the group, organization unit, or position, then associating a privilege with the appropriate scoped system action:


Organizations
Lancing Insurance (LancingInsurance)
Claims Management (ClaimsManagement)
Claims Handling (ClaimsHandling)
Claims Handler (ClaimsHandler)
Claims Manager (ClaimsManager)
Lansing Customer Service (LansingCustome
Call Handler (CallHandler)
Call Supervisor (CallSupervisor)
Position
General
Capabilities
Privileges
Resources
System Actions
Description
Advanced
System Actions

Name
Close Other Resources Items
Open Other Resources Items
Reallocate To Offer Set
Reallocate Work Item To World
Set Resource Order Filter Criteria
View Work List
Work Item Allocation

Note that although there are seven scoped system actions available in TIBCO Business Studio, there are only three of them that are used in Workspace at this time. Those are:

- **BRM.viewWorkList** - Controls access to supervised work views:

  - To create a supervised work view for an organizational entity, you must possess the privilege assigned to the system action, **BRM.viewWorkList**, on the group, organization unit, or position.
  - To create a supervised work view for an individual resource, you must possess the privilege assigned to the system action, **BRM.viewWorkList**, on a specific position to which the resource has been mapped

    This is further explained in Creating Supervised Work Views.

- BRM.closeOtherResourcesItems - Controls whether you can cancel a work item in a supervised work view for another resource.
- BRM.reallocateToOfferSet - Controls whether you can allocate a work item to a specific person in the original offer set.

The scoped system actions available in TIBCO Business Studio that are not currently used in the application are:

- **BRM.setResourceOrderFilterCriteria** (Eventually this may control whether someone viewing a supervised work view can save filter and sort criteria to the server so that it will become the default for that resource.)
- BRM.openOtherResourcesItems (You cannot open another resource's work items from a supervised work view.)
- **BRM.reallocateWorkItemToWorld** (This system action is available at the organization model level; use that one to also control access to the Allocate Work Items to World function on a supervised work view for a resource; this function is not available on a supervised work view for an organizational entity.)
- BRM.workItemAllocation (This system action is available at the organization model level; use that one to also control access to the Allocate functions on a supervised work view.)

Creating Supervised Work Views

The primary use of the scoped system actions are for controlling the ability to create supervised work views. A few of the other system actions available at the organizational entity level control access to functions *from* a supervised work view. But those, of course, would have no meaning unless you can first create the supervised work view.

The following describes what is necessary to set up system actions to be able to create a supervised work view.

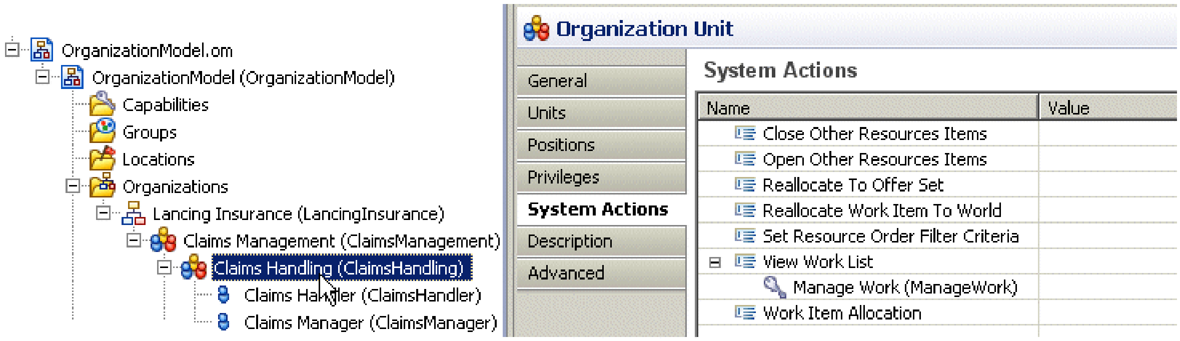
When a supervised work view is created in the application, it can be set up to contain either:

- **Work items sent to an organizational entity** - This type of supervised work view only contains work items that were sent directly to the specified organizational entity. When creating this type of supervised work view you can specify that the view contain either work items that were offered to the organizational entity and that still have a state of Offered, or work items that were offered to the organizational entity but that are now allocated to one or more users (i.e., their state is Allocated).

  To be able to create this type of supervised work view:

  - A privilege must be assigned to the **BRM.viewWorkList** system action for the group, organization unit, or position for which you want to supervise, and
  - you must possess the privilege (and possible qualifier) assigned to the scoped **BRM.viewWorkList** system action for the organizational entity you want to supervise (you gain the privilege by being mapped to a group or position that gives you that privilege).

    In the following example, the Claims Handling organization unit's scoped "View Work List" system action has a privilege assigned:


OrganizationModel.om
OrganizationModel (OrganizationModel)
Capabilities
Groups
Locations
Organizations
Lancing Insurance (LancingInsurance)
Claims Management (ClaimsManagement)
Claims Handling (ClaimsHandling)
Claims Handler (ClaimsHandler)
Claims Manager (ClaimsManager)
Organization Unit
General
Units
Positions
Privileges
System Actions
Description
Advanced

System Actions

Name	Value
Close Other Resources Items	
Open Other Resources Items	
Reallocate To Offer Set	
Reallocate Work Item To World	
Set Resource Order Filter Criteria	
View Work List	
Manage Work (ManageWork)	
Work Item Allocation	

In this example, a logged-in user who possesses the "Manage Work" privilege, can create a supervised work view for the Claims Handling organization unit.

Note - The checking for needed system actions is hierarchical—for more information, see System Action Hierarchy.

- **Work items for an individual resource** - This type of supervised work view contains all of the work items that are currently in the Inbox (unfiltered) of the resource for whom the supervised work view was set up.

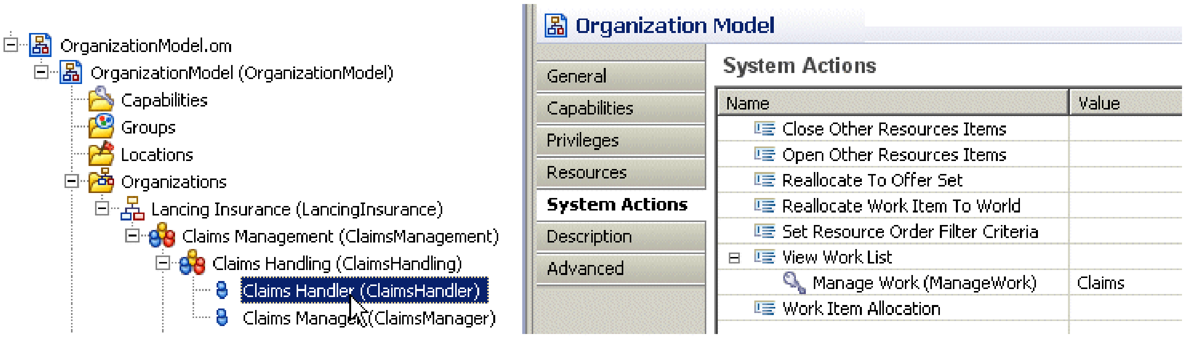
  To be able to create this type of supervised work view:

  – A privilege must be assigned to the **BRM.viewWorkList** system action for a specific position to which the desired resource is mapped,

  – you must possess the privilege (and possible qualifier) assigned to the scoped **BRM.viewWorkList** system action for the position (you gain the privilege by being mapped to a group or position that gives you that privilege), and

  – you must possess the privilege (and possible qualifier) assigned to the **DE.resourceAdmin** system action (which is a system action available at the organization model level)—this gives you permission to list the resources so that you can choose one to supervise.

In the following example, a privilege has been assigned to the "View Work List" (**BRM.viewWorkList**) scoped system action on the Claims Handler position:


OrganizationModel.om
OrganizationModel (OrganizationModel)
Capabilities
Groups
Locations
Organizations
Lancing Insurance (LancingInsurance)
Claims Management (ClaimsManagement)
Claims Handling (ClaimsHandling)
Claims Handler (ClaimsHandler)
Claims Manager (ClaimsManager)
Organization Model
General
Capabilities
Privileges
Resources
System Actions
Description
Advanced

System Actions

Name	Value
Close Other Resources Items	
Open Other Resources Items	
Reallocate To Offer Set	
Reallocate Work Item To World	
Set Resource Order Filter Criteria	
View Work List	
Manage Work (ManageWork)	Claims
Work Item Allocation	

This allows a user who has the "Manage Work" privilege, with a qualifier of "Claims", to create a supervised work view for a user who has been mapped to the Claims Handler position (assuming the user creating the supervised work view also possesses a privilege assigned to the **DE.resourceAdmin** system action).

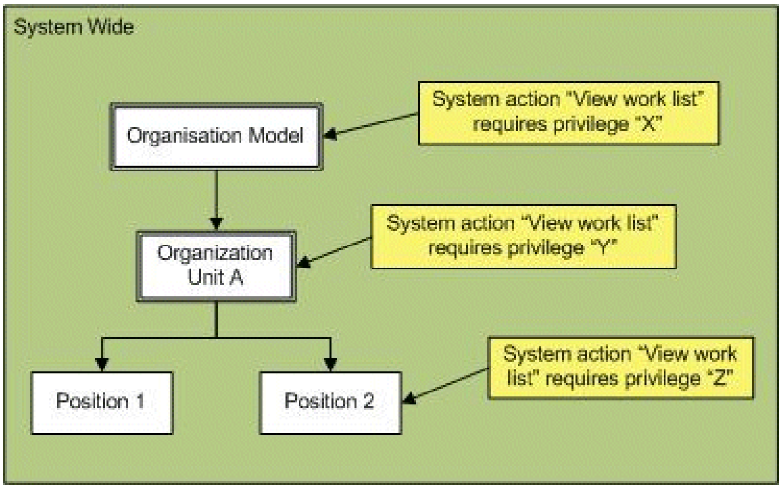
Note - The checking for needed system actions is hierarchical—for more information, see System Action Hierarchy.

System Action Hierarchy

The checking of system actions is hierarchical.

If different privileges are assigned to the same system action (e.g., **BRM.viewWorkList**) at different levels in the organization model, each level is checked when determining whether a user has the necessary privilege to be able to perform a particular system action. If the lowest level has no required privileges assigned, the parent entity is checked, and so on up the organization model hierarchy to the system-wide, organization model-level value.

For example, in the following diagram the "View work list" system action has been associated with three different privileges, "X", "Y" and "Z", at three different levels—the organization model, organization unit A, and position 2.


System Wide
Organisation Model
System action "View work list" requires privilege "X"
Organization Unit A
System action "View work list" requires privilege "Y"
Position 1
Position 2
System action "View work list" requires privilege "Z"

This means that a user who holds privilege "X", "Y" or "Z" can create a supervised work view:

- for either Position 2 or Organization Unit A, or
- for individual resources that have been mapped to Position 2.

If a user wants to view the work list of a user who holds Position 1, they must hold privilege "X" or "Y". This is because no privilege has been associated with Position 1, so any privileges associated with the parent entity are used instead. If privilege "Y" had not been associated with Organization Unit A, the user would instead need privilege "X", defined in the parent Organization Model.

As well as assigning different privileges at different levels, as shown above, qualifiers on the same privilege can be used to refine how access to a particular system action is controlled. (When comparing a required privilege to a held privilege, if either side is not qualified the comparison is positive. If both sides are qualified, the qualifications must match for the comparison to be positive.)

For more information about system actions, see the *BPM Concepts* guide.

Configuring Events

You can customize how event lists are created and displayed to the user.

For details about working with events from a user's standpoint, see the *TIBCO Workspace User's Guide*.

The following aspects of events can be customized:

- **Event descriptions** - This allows you to customize the text in event lists that describe the event.

  For more information, see Customizing Event Descriptions.
- **Links** - This allows you to customize the links that allow users to create a new event view containing events related to the currently selected entity or event.

  For more information, see Customizing Links.
- **View templates** - This allows you to provide custom templates for the user to use when creating an event view using the wizard.

  For more information, see Customizing View Templates.
- **Event attributes** - This allows you to customize the event attributes that are displayed when viewing events in an event list.

  For more information, see Customizing Event Attributes.

Events can be customized on a per-user basis. The user's privileges are used to determine how to customize the events displayed for that user. For information, see Mapping Privileges to Event Roles.

Location of the Event Configuration Files to Configure

The specific configuration files you need to modify to configure events depends on whether the application you are configuring is deployed or not.

- **A deployed application** - If configuring a deployed application, use the Configuration Administrator to modify event configuration files.

  For information about using the Configuration Administrator, see Using the Configuration Administrator.
- **A non-deployed WCC application** - If configuring a non-deployed WCC application (which might be the Workspace application), open the event configuration files in your local development environment.

  Events are configured in the following locations:

  – **eventRoles.xml** - This is the file in which privileges are mapped to event roles, which determines which of the `eventRoles` subdirectories (see below) are used to configure events for the logged-in user.

    This file is located as follows:

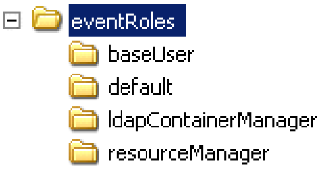
    ```
    StudioHome\wcc\version\JSXAPPS\WCCProjectName\
    ```

    where:

    - *StudioHome* is the directory in which TIBCO Business Studio was installed.
    - *StudioHome* is the directory in which TIBCO Business Studio was installed.
    - *version* is the version number of Workspace that was installed with TIBCO Business Studio.
    - *WCCProjectName* is the name of the General Interface Builder project that contains your custom application. If you are working with the Workspace application, this is "workspace".

      For more information about the `eventRoles.xml` file, see Mapping Privileges to Event Roles.

- **eventRoles directroy** - This directory contains subdirectories, each one corresponding to a role. For example:

  Each of the subdirectories contains the following configuration files:

  - eventAttributes.xml
  - eventComponents.xml
  - eventDescriptions.xml
  - eventLinks.xml
  - eventViewTemplates.xml

  These files are used to configure event attributes, components, descriptions, links, and view templates, respectively, for users that have been mapped to the role name that matches the name of the subdirectory.

  The configuration files in the `default` directory are used if a user does not have any of the privileges mapped to roles in the `eventRoles.xml` file (it is also used by the Configuration Administrator when a new role is added, or when setting a role's configurations to the default, for a deployed application).

  These directories are located as follows:

  ```
  StudioHome\wcc\version\JSXAPPS\base\locale\eventRoles\
  ```

  where:

  - *StudioHome* is the directory in which TIBCO Business Studio was installed.
  - *version* is the version number of Workspace that was installed with TIBCO Business Studio.

- **eventLocale.xml** - This file contains text strings that are referenced by all of the event configuration files (`eventAttributes.xml`, `eventComponents.xml`, `eventDescriptions.xml`, `eventLinks.xml`, and `eventViewTemplates.xml`).

  This file is located in the following directory:

  ```
  StudioHome\wcc\version\JSXAPPS\base\locale\
  ```

  where:

  - *StudioHome* is the directory in which TIBCO Business Studio was installed.
  - *version* is the version number of Workspace that was installed with TIBCO Business Studio.

  The `eventLocale.xml` file consolidates all of the event-related text into a single file, which helps facilitate localizing. For more information, see Event-Related Text.

Event-Related Text

All text that is displayed in the application event view list and Event Viewer is consolidated into a single "event locale" file.

This file is located as follows:

```
StudioHome\wcc\version\JSXAPPS\base\locale\eventLocale.xml
```

where:

- *StudioHome* is the directory in which TIBCO Business Studio was installed.
- *version* is the version number of Workspace that was installed with TIBCO Business Studio.

> If the application has been localized, additional event locale files with the country code in the file name (for example., `eventLocale.de.xml` for German) will also exist. For more information, see Configuring Events .

All of the event configuration files then point to the text in the locale file rather than having the text in the configuration files themselves. This allows for easier localization of the application.

The `eventLocale.xml` file contains a series of records, each with two attributes:

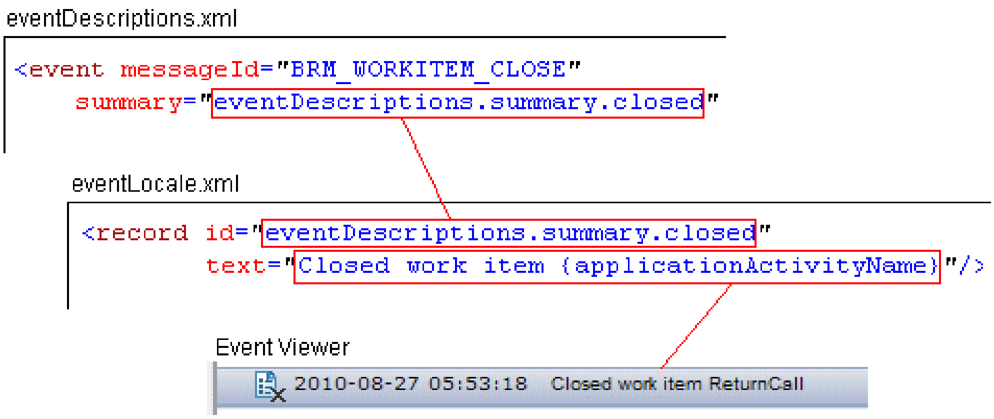
```
<record id="eventDescriptions.summary.closed"
        text="Closed work item {applicationActivityName}"/>
```

- **id** - This is a *key* that an event configuration files references to point to the string in the corresponding **text** attribute.

  The value in this attribute consists of three parts, as follows:

  - Part 1 identifies the event configuration file that references the text.
  - Part 2 identifies the attribute in the event configuration file that references the text.
  - Part 3 identifies the action, event, attribute, or link to which the text applies.
- **text** - The string that is displayed at runtime.

The following example shows how the text for a “summary” event description (which is defined in the `eventDescriptions.xml` file) is obtained from the `eventLocale.xml` file, and shown in the Event Viewer:



eventDescriptions.xml

```
<event messageId="BRM_WORKITEM_CLOSE"
    summary="eventDescriptions.summary.closed"
```

eventLocale.xml

```
<record id="eventDescriptions.summary.closed"
        text="Closed work item {applicationActivityName}"/>
```

Event Viewer

2010-08-27 05:53:18   Closed work item ReturnCall

To modify the text referenced by one of the attributes in the `eventDescriptions.xml`, `eventLinks.xml`, `eventViewTemplates.xml`, `eventAttributes.xml`, or `eventComponents.xml` file, locate the key in an **id** attribute in the `eventLocale.xml` file, then modify the string in the **text** attribute for that record.

Substitution tokens can be placed in the text strings by including an event attribute name enclosed in braces (such as {applicationActivityName} in the example above).

At runtime, the substitution tokens are replaced with values of the event attributes. For more information about substitution tokens, see Substitution Tokens.

Mapping Privileges to Event Roles

Users can have one or more privileges, which they inherit as a result of being assigned to groups or positions in the organization model (for information about assigning users to groups and positions, see the *Organization Browser User's Guide.* These privileges are used to determine how to display event information to the user.

This is accomplished using *event roles*, which specify the event customization files to use, based on the privileges held by the logged-in user.

Event roles are defined in the `eventRoles.xml` configuration file.

The following is an example `eventRoles.xml` file:

```
<data>
  <privilege name="ResourceManager" role="ResourceManager" precedence="1" />
  <privilege name="LdapContainerManager" role="LdapContainerManager"
precedence="2" />
  <privilege name="BaseUser" role="BaseUser" precedence="3" />
</data>
```

The attributes in the `eventRoles.xml` file have the following meanings:

- **name** - This is the name of a user access privilege. If the logged-in user has this privilege, the user is mapped to the event role specified in the **role** attribute, which in turn specifies the event configuration files to use for that user.

  If the logged-in user has more than one of the privileges listed in the **name** attributes, the **precedence** attribute is used to determine which role the user is mapped—see below.

  If the logged-in user does not have any of the privileges listed in the **name** attributes, the user is assigned a "default" role, which results in the configuration files in the `...\eventRoles\default` directory being used for that user (for more information about the event role directories, see the **role** attribute below).

- **role** - This identifies the subdirectory, in the `eventRoles` directory, in which the event configuration files will be obtained to determine how to display events for users who hold a privilege mapped to that role.

  Note that the role name does not *have* to be the same as the name of the privilege.

  You could also have multiple privilege names mapped to the same role. For instance, you could have "Reviewer" and "ReviewerAssistant" privileges both mapped to the "Reviewer" role. Users with either of those privileges would use the same event configuration files.

- **precedence** - This determines which role to use if the logged-in user has multiple privileges. If the user has multiple privileges, the role that has the highest precedence (1 being the highest) is used.

  If the user has multiple privileges, and the highest precedences are tied, the first of those in the list is used.

Note that all roles in the out-of-the-box `eventRoles.xml` file are commented out by default. This results in all users using the "default" role by default (i.e., it uses the event configuration files in the `...\eventRoles\default` directory).

You must update the out-of-the-box `eventRoles.xml` file to include privileges used in your system—see Adding Event Roles.

Adding Event Roles

You can add event roles (for instance, if a new privilege is added to the system) to the `eventRoles.xml` file.

The way in which you add event roles depends on whether your application is deployed or not.

Adding Event Roles to a Deployed Application

Procedure

1. Access the `eventRoles.xml` file using the Configuration Administrator.
2. Add a <Privilege/> element for the new role. For information about what to include in the element's attributes, see Mapping Privileges to Event Roles.
3. Click **Apply**.

   A directory of the same name is dynamically added under the `Event Roles` directory in the left pane of the Configuration Administrator. It adds the event configuration files for the newly added role to the database by copying the event configuration files from the `default` directory shown in the left pane (note that it actually copies the files from the `default` directory on the file system, *not* from the database if you had previously accessed the default configuration files, which would have caused them to be written to the database).

   This allows you to add roles via configuration (i.e., without requiring a re-deployment).
4. You can now configure each of the individual event configuration files for the new role by accessing those files in the newly created event role directory in the left pane of the Configuration Administrator.

   For more information about using the Configuration Administrator to configure events, see Configuring and Customizing a Deployed Application.

Adding Event Roles to a Non-Deployed Application

Procedure

1. Open the `eventRoles.xml` file, which is located as follows:

   ```
   StudioHome\wcc\version\JSXAPPS\WCCProjectName\
   ```

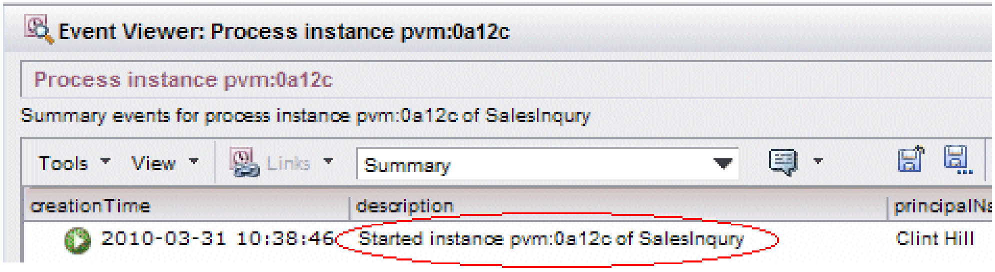
   where:

   - *StudioHome* is the directory in which TIBCO Business Studio was installed.
   - *version* is the version number of Workspace that was installed with TIBCO Business Studio.
   - *WCCProjectName* is the name of the General Interface Builder project that contains your custom application. If you are working with the Workspace application, this is “workspace”.
2. Add a <Privilege/> element for the new role. For information about what to include in the element's attributes, see Mapping Privileges to Event Roles.
3. Save and close the `eventRoles.xml` file.
4. Add a new subdirectory under the `...\eventRoles` directory that matches the name of the role.

   The easiest thing to do is copy the `default` directory and rename it to the new role name.
5. Copy the event configuration files to the new directory (they will already be there if you copied the `Default` directory in step 4).
6. You can now configure each of the individual event configuration files for the new role by directly accessing those files and making the desired changes.

Customizing Event Descriptions

You can customize the descriptions of events that are displayed in the **description** column of event lists.



Event Viewer: Process instance pvm:0a12c

Process instance pvm:0a12c

Summary events for process instance pvm:0a12c of SalesInqury

creationTime	description	principalNa
2010-03-31 10:38:46	Started instance pvm:0a12c of SalesInqury	Clint Hill

Note that the user can choose to display either detailed or summarized descriptions—both of these descriptions are customizable.

Event descriptions are specified in the `eventDescriptions.xml` file. For information about the location of this file, see Location of the Event Configuration Files to Configure.

The `eventDescriptions.xml` file contains **event** elements, one for each possible event. For example, the following is for the “close work item” event:

```
<event messageId="BRM_WORKITEM_CLOSE"
       summary="eventDescriptions.summary.closed"
       detail="eventDescriptions.detail.closed"/>
```

The **Event** element has the following attributes:

- **messageId** - Identifies the event. Note that the messageId is prefixed with an acronym for the component that generates the event, as follows:
  - BRM - Business Resource Management
  - DE - Directory Engine
  - WPCORE - Work Presentation
  - BX - Process Engine
  - WS - Workspace
- **summary** - The string in this attribute is a key that points to the description that is displayed at runtime for the event when the user is displaying a summarized description: **View** > **Event Descriptions** > **Show Summarized Descriptions**.

  The key in this attribute is mapped to the summary description in the `eventLocale.xml` file—for more information, see Event-Related Text.
- **detail** - The string in this attribute is a key that points to the description that is displayed at runtime for the event when the user is displaying a detailed description: **View** > **Event Descriptions** > **Show Detailed Descriptions**.

  The key in this attribute is mapped to the detail description in the `eventLocale.xml` file—for more information, see Event-Related Text.

Procedure

1. Open the `eventDescriptions.xml` file for the desired role.
2. Locate the **messageId** for the event whose description you want to customize.

   Notice that events in `eventDescriptions.xml` are categorized by type, e.g., work item events, resource events, deployment events, etc.

3. Note the key in the **summary** and/or **detail** attribute, depending on whether you want to change the summary description, detail description, or both.
4. Open the `eventLocale.xml` file—for information about its location, see Event-Related Text.
5. Locate the key(s) you noted in step 3.
6. Modify the description in the **text** attribute as desired.

   Substitution tokens can be placed in the descriptions by including an event attribute name enclosed in braces. For example:

   ```
   text="Closed work item {applicationActivityName}"
   ```

   At runtime, the substitution tokens are replaced with values of the event attributes. For more information about substitution tokens, see Substitution Tokens.
7. Save and close the `eventLocale.xml` file.
8. Close the `eventDescriptions.xml` file.

Result

At runtime, if the messageId for an event is not found in the `eventDescriptions.xml` file, a default description for the event from the server is displayed in the **description** column for that event.

Also note that you can specify whether the summary or detail description is displayed by default when an event list is displayed. For information, see defaultEventDescription Attribute.

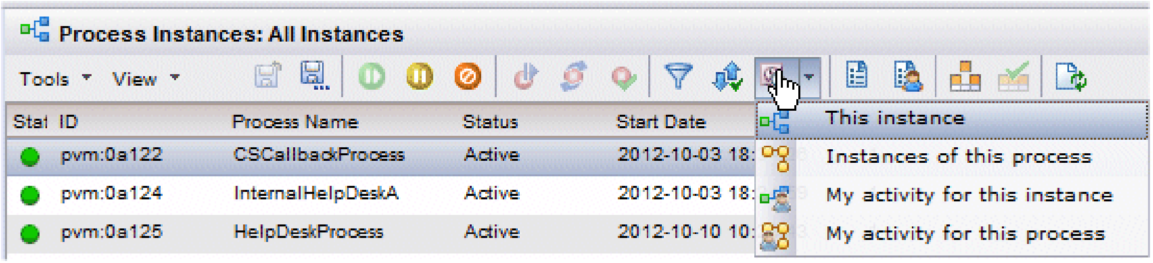
Customizing Links

You can define *links* that allow the user to create a new event view containing events related to the currently selected entity or event.

There are three types of links:

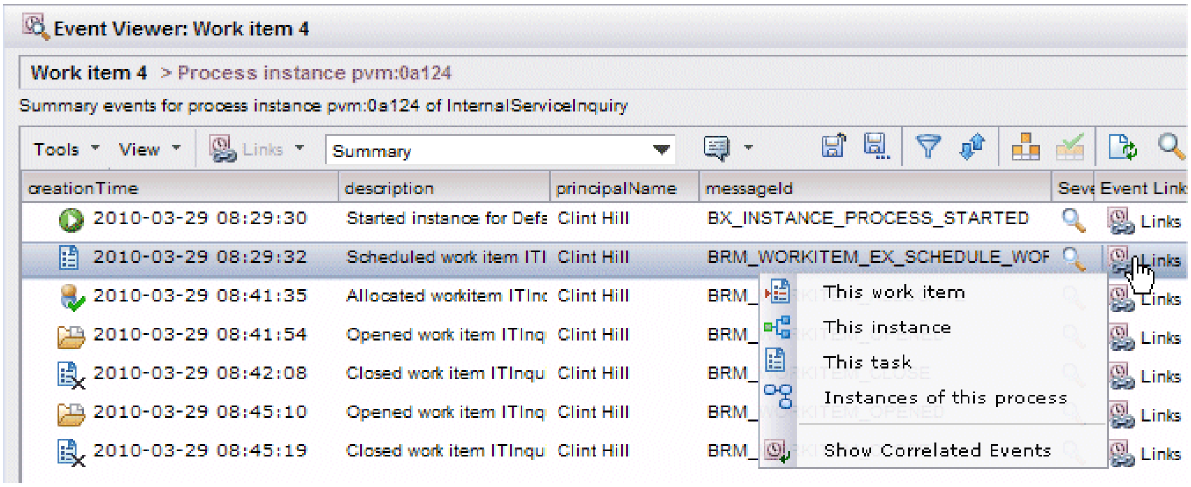
- **Contextual Links** - These links are accessed via the toolbar on lists of work items, process instances, organizational entities, and resources. These links generate an event view containing a list of events in the context of the selected item in the list, using attributes of that item to construct a filter to retrieve the pertinent events.

  The following is an example of contextual links available from the process instance list:

  
  Process Instances: All Instances
  Tools
  View
  This instance
  Instances of this process
  My activity for this instance
  My activity for this process

Stat	ID	Process Name	Status	Start Date
●	pvm:0a122	CSCallbackProcess	Active	2012-10-03 18:
●	pvm:0a124	InternalHelpDeskA	Active	2012-10-03 18:
●	pvm:0a125	HelpDeskProcess	Active	2012-10-10 10:

  The links shown in the drop-down menu above is an example provided in the Workspace application. You can customize the drop-down menu that is displayed by specifying the selections available from that menu for the type of entity or event that is selected. This allows you to control which additional event views/lists the user will be able to create.
- **Event Links** - These are links that are accessed via a **Links** button drop-down list from the Event Viewer toolbar, or in the **Event Links** column for the currently selected event. These links allow the user to explore events that are related to, or relevant to, a specific event type. Selecting one of these links causes a new event list to be displayed in the Event Viewer, plus it generates a path (also known as a "bread crumb trail") of event links in the upper-left part of the Event Viewer. The following is an example:



Event Viewer: Work item 4

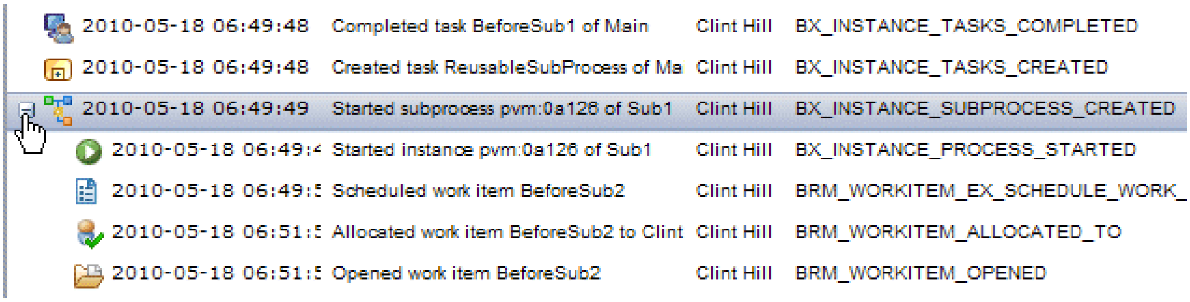
Work item 4 > Process instance pvm:0a124

Summary events for process instance pvm:0a124 of InternalServiceInquiry

creationTime	description	principalName	messageId	Seve	Event Link
2010-03-29 08:29:30	Started instance for Defa	Clint Hill	BX_INSTANCE_PROCESS_STARTED		Links
2010-03-29 08:29:32	Scheduled work item ITI	Clint Hill	BRM_WORKITEM_EX_SCHEDULE_WOF		Links
2010-03-29 08:41:35	Allocated workitem ITInc	Clint Hill	BRM_		Links
2010-03-29 08:41:54	Opened work item ITInq	Clint Hill	BRM_		Links
2010-03-29 08:42:08	Closed work item ITInqu	Clint Hill	BRM_		Links
2010-03-29 08:45:10	Opened work item ITInq	Clint Hill	BRM_		Links
2010-03-29 08:45:19	Closed work item ITInqu	Clint Hill	BRM_		Links

The links shown in the drop-down menu above is an example provided in the Workspace application. You can customize the drop-down menu that is displayed by specifying the selections available from that menu for the type of event that is selected. This allows you to control which additional event views/lists the user will be able to create.

- **Sub-Process Links** - These links are accessed by expanding a sub-process start event in the Event Viewer:



2010-05-18 06:49:48	Completed task BeforeSub1 of Main	Clint Hill	BX_INSTANCE_TASKS_COMPLETED
2010-05-18 06:49:48	Created task ReusableSubProcess of Ma	Clint Hill	BX_INSTANCE_TASKS_CREATED
2010-05-18 06:49:49	Started subprocess pvm:0a126 of Sub1	Clint Hill	BX_INSTANCE_SUBPROCESS_CREATED
2010-05-18 06:49:4	Started instance pvm:0a126 of Sub1	Clint Hill	BX_INSTANCE_PROCESS_STARTED
2010-05-18 06:49:5	Scheduled work item BeforeSub2	Clint Hill	BRM_WORKITEM_EX_SCHEDULE_WORK_
2010-05-18 06:51:5	Allocated work item BeforeSub2 to Clint	Clint Hill	BRM_WORKITEM_ALLOCATED_TO
2010-05-18 06:51:5	Opened work item BeforeSub2	Clint Hill	BRM_WORKITEM_OPENED

These links define the filter query executed when the sub-process event is expanded—for more information, see Sub-Process Links.

All links are specified in the `eventLinks.xml` file. For information about the location of this file, see Location of the Event Configuration Files to Configure.

Defining the Link Type

Both contextual and event links are defined by <links/> elements in the `eventLinks.xml` file, with a **messageId** attribute that identifies the context (for contextual links) or event type (for event links) to which the links are associated.

For information about defining a sub-process link type, see Sub-Process Links.

Contextual Links

The **messageId** attribute for contextual links is suffixed with "_WorkspaceContext":

```
<links messageId="Context_WorkspaceContext">
```

where *Context* identifies the entity that is the context for the event list. The available values for *Context* are:

- **WorkItem** - defines the contextual links when a work item is currently selected in the work item list.

- **ProcessInstance** - defines the contextual links when a process instance is currently selected in the process instance list.
- **LdapContainer** - defines the contextual links when an individual LDAP container is currently selected in the Organization Browser.
- **LdapContainers** - defines the contextual links when “LDAP Containers” is currently selected in the Organization Browser.
- **Group** - defines the contextual links when a group is currently selected in the Organization Browser.
- **Position** - defines the contextual links when a position is currently selected in the Organization Browser.
- **Organization** - defines the contextual links when an organization is currently selected in the Organization Browser.
- **OrgUnit** - defines the contextual links when an organization unit is currently selected in the Organization Browser.
- **OrgModelVersion** - defines the contextual links when the organization model is currently selected in the Organization Browser.
- **ResourceList** - defines the contextual links when a resource is currently selected in the Organization Browser resource list.

For example:

```
<links messageId="WorkItem_WorkspaceContext">
          .
          .
          .
</links>
```

This example specifies the links available when the user selects a work item and clicks the **Links** button on the work item list.

Event Links

The **messageId** attribute for event links identifies the event type:

```
<links messageId="EVENT_TYPE">
```

- where *EVENT_TYPE* identifies the type of event for which events will be returned in the event list. See the `eventLinks.xml` file for example <links/> elements for each of the available event types.

For example:

```
<links messageId="BRM_WORKITEM_OPENED">
          .
          .
          .
</links>
```

This example specifies the links available when the user selects an event of type BRM_WORKITEM_OPENED in the Event Viewer, then clicks the **Links** button.

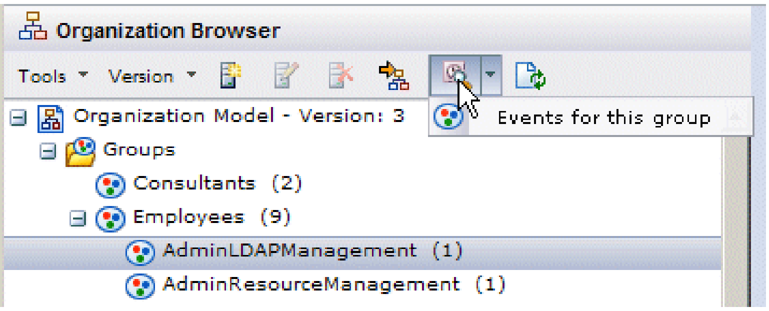
Defining the Individual Links

Each <links/> element used for contextual and event links can contain one or more <link/> elements, each one defining a selection from a **Link** button drop-down list, as well as the resulting event list when that selection is made from the **Link** button drop-down list.

The following example shows a <links/> element that contains a single <link/> element.

```
<links messageId="Group_WorkspaceContext">
  <link name="eventLinks.name.group"
        menuText="eventLinks.menuText.thisGroup"
        image="JSXAPPS/base/application/images/mnmGroup.gif"
        defaultFilterId="eventLinks.id.all"
        defaultEventDescription="detail">
        <filters>
          <filter id="eventLinks.id.all"
                  query="managedObjectId='{guid}' AND
                  messageCategory='ORGANIZATIONAL_ENTITY' AND
                  entityType='GROUP'"
                  description="eventLinks.description.eventsGroup"/>
         </filters>
  </link>
</links>
```

As this example has a **messageId** of “Group_WorkspaceContext”, it defines the contextual links available from the **Links** button drop-down list when a group is selected in the Organization Browser, as follows:

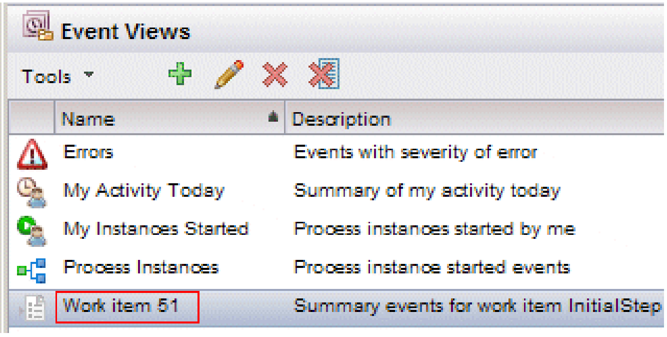


Each <link/> element contains:

- Five attributes that define the characteristics of the link. These attributes are described below.
- A <filters/> element with one or more subordinate <filter/> elements that specify the pre-defined filters that will be available on the event list that results from the link being selected by the user. For information about specifying these pre-defined filters, see Specifying Pre-defined Filters.

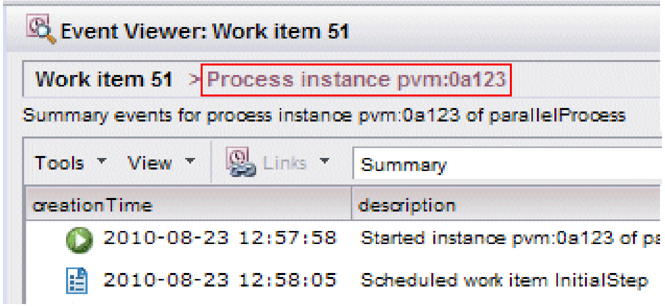
name Attribute

For contextual links, the **name** attribute specifies the text to display in the **Name** column of the newly created event view:



Name	Description
Errors	Events with severity of error
My Activity Today	Summary of my activity today
My Instances Started	Process instances started by me
Process Instances	Process instance started events
Work item 51	Summary events for work item InitialStep

For event links, the **name** attribute specifies the text that is displayed in the "bread crumb trail" in the Event Viewer as the user displays additional event lists using links:



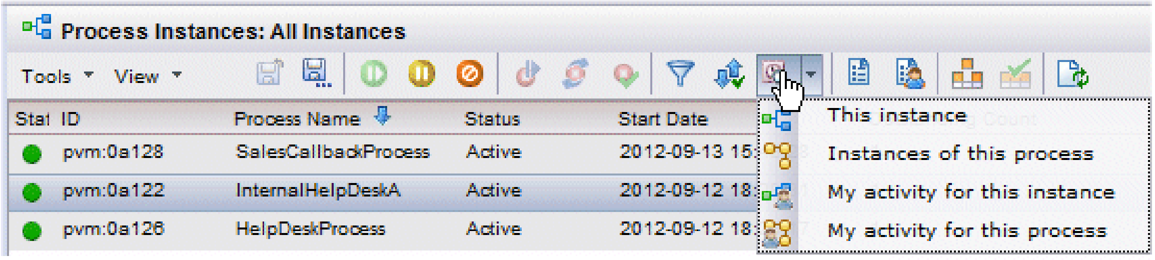
Summary events for process instance pvm:0a123 of parallelProcess

creationTime	description
2010-08-23 12:57:58	Started instance pvm:0a123 of pa
2010-08-23 12:58:05	Scheduled work item InitialStep

The **name** attribute contains a key that points to the text string in the `eventLocale.xml` file—for more information, see Event-Related Text.

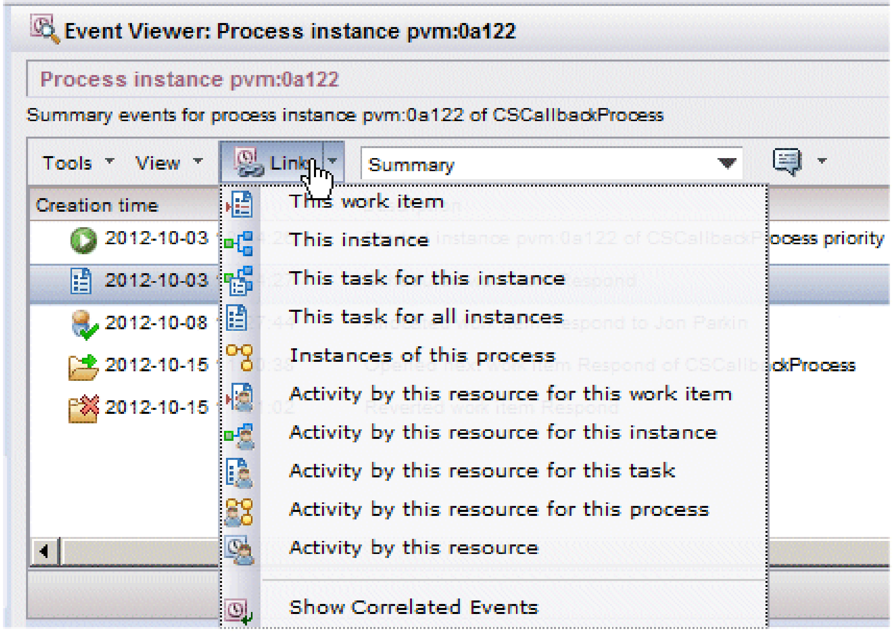
menuText Attribute

For contextual links, the **menuText** attribute specifies the text that appears in the **Links** button drop-down list for the currently selected entity:


Process Instances: All Instances
Tools
View
This instance
Instances of this process
My activity for this instance
My activity for this process

Stat	ID	Process Name	Status	Start Date
●	pvm:0a128	SalesCallbackProcess	Active	2012-09-13 15:
●	pvm:0a122	InternalHelpDeskA	Active	2012-09-12 18:
●	pvm:0a126	HelpDeskProcess	Active	2012-09-12 18:

For event links, the **menuText** attribute specifies the text that appears in the **Links** button drop-down list for the currently selected event type in the Event Viewer:

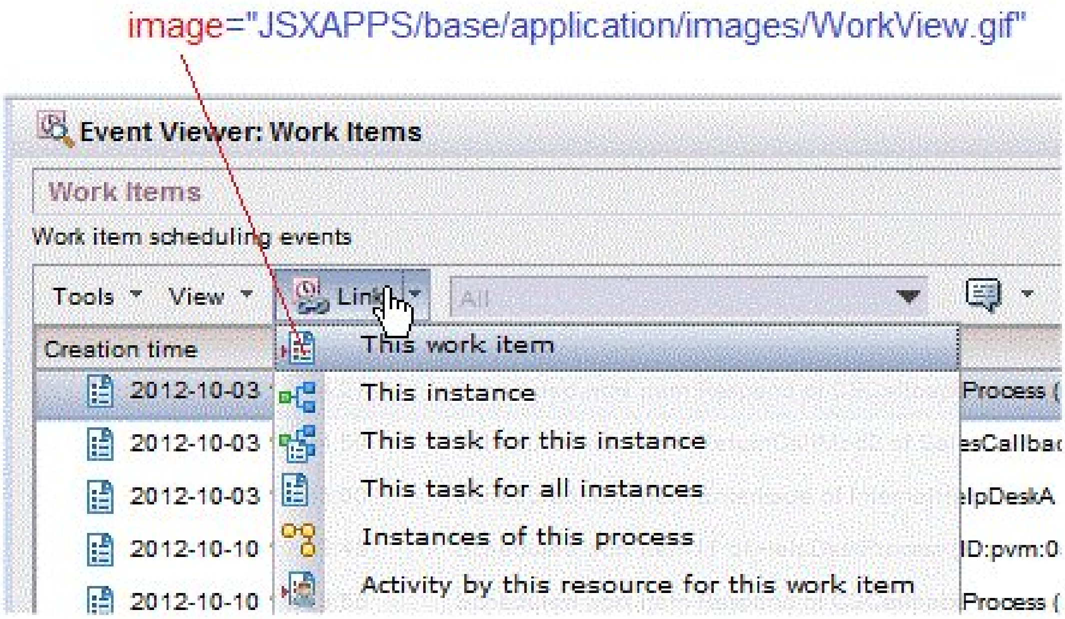

Event Viewer: Process instance pvm:0a122
Process instance pvm:0a122
Tools
View
Links
Summary
Creation time
2012-10-03
2012-10-03
2012-10-08
2012-10-15
2012-10-15
ocess priority
ckProcess
This work item
This instance
This task for this instance
This task for all instances
Instances of this process
Activity by this resource for this work item
Activity by this resource for this instance
Activity by this resource for this task
Activity by this resource for this process
Activity by this resource
Show Correlated Events

Summary events for process instance pvm:0a122 of CSCallbackProcess

The **menuText** attribute contains a key that points to the text string in the `eventLocale.xml` file—for more information, see Event-Related Text.

image Attribute

This attribute specifies the image to display to the left of the applicable **Links** drop-down menu selection. For example:



image="JSXAPPS/base/application/images/WorkView.gif"

If an image is not specified, the default image is displayed.

For contextual links, you should also have a corresponding grey-scale image, with a file name that ends with "Temp.gif" (e.g., `WorkViewTemp.gif`). The "Temp" version of the image is displayed in the Event Views list when the event link is executed and a temporary view is created.

defaultFilterId Attribute

This attribute specifies which of the pre-defined filters specified by the <filter/> element to display by default.

For more information, see Specifying Pre-defined Filters.

defaultEventDescription Attribute

This attribute specifies which of the descriptions (summary or detail) to display in the **description** column by default.

For more information, see Customizing Event Descriptions.

Defining Link Sets

Each <links/> element used with contextual and event links can contain one or more *link sets*, which are defined using <linkSet/> elements. A <linkSet/> element allows you to define a group of <link/> elements, then reuse that group multiple times in various contextual and event links.

For example, the following defines a link set:

```
<linkSet name="WS_SESSION_SET">
  <link name="eventLinks.name.resourceActivity"
        menuText="eventLinks.menuText.activityByResource"
        image="JSXAPPS/base/application/images/mnmAllUserActivity.gif"
        defaultFilterId="eventLinks.id.summary"
        defaultEventDescription="detail">
        <filters>
```

```
           <!-- All activity by this resource -->
           <filter id="eventLinks.id.all"
                   query="principalName='{principalName}'"
                   description="eventLinks.description.allActivityByResource"/>
              .
              .
              .
</linkSet>
```

This link set can then be used by including the <linkSet/> element in the <links/> element, as follows:

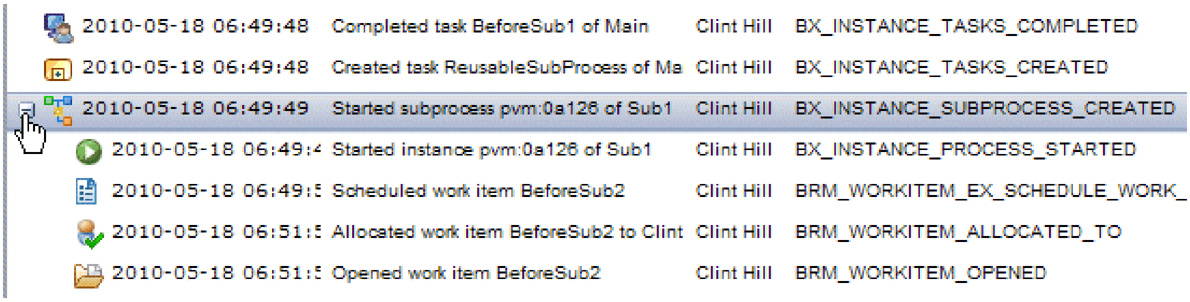
```
<links messageId="WS_SECURITY_LOGIN">
   <linkSet name="WS_SESSION_SET"/>
</links>
```

There are many other examples of link sets in the `eventLinks.xml` file.

Sub-Process Links

The sub-process link defines the filter query that is executed when an event with `messageId`='**BX_INSTANCE_SUBPROCESS_CREATED**' is expanded to show events of the sub-process instance.

For example:



2010-05-18 06:49:48	Completed task BeforeSub1 of Main	Clint Hill	BX_INSTANCE_TASKS_COMPLETED
2010-05-18 06:49:48	Created task ReusableSubProcess of Ma	Clint Hill	BX_INSTANCE_TASKS_CREATED
2010-05-18 06:49:49	Started subprocess pvm:0a126 of Sub1	Clint Hill	BX_INSTANCE_SUBPROCESS_CREATED
2010-05-18 06:49:4	Started instance pvm:0a126 of Sub1	Clint Hill	BX_INSTANCE_PROCESS_STARTED
2010-05-18 06:49:5	Scheduled work item BeforeSub2	Clint Hill	BRM_WORKITEM_EX_SCHEDULE_WORK_
2010-05-18 06:51:5	Allocated work item BeforeSub2 to Clint	Clint Hill	BRM_WORKITEM_ALLOCATED_TO
2010-05-18 06:51:5	Opened work item BeforeSub2	Clint Hill	BRM_WORKITEM_OPENED

There is a single <subProcessLink> element in the `eventLinks.xml` file that controls the filtering for all sub-process events:

```
<subProcessLink>
   <filter query="(managedObjectId='{subprocessInstanceId}' AND
                 (messageId='BX_INSTANCE_PROCESS*' OR
                  messageId='BX_INSTANCE_SUBPROCESS*')) OR
                 (managedObjectId='{subprocessInstanceId}' AND
                  managedObjectType &lt;&gt; '*Event' AND
                 managedObjectType &lt;&gt; '*Gateway' AND
                   managedObjectType &lt;&gt; 'userTask') OR
                 (messageCategory='WORK_ITEM' AND
                  parentObjectId='{subprocessInstanceId}')"
           maxEvents="100"/>
  <sorts>
        <sort attribute="creationTime" order="ASCENDING"/>
  </sorts>
</subProcessLink>
```

filter Element

The <subProcessLink> element contains a <filter> element that has the following attributes:

- **query** - This attribute is a string value used to query the event collector database for events related to the sub-process. All events that satisfy the filter string specified in this attribute are returned from the event collector database and displayed in the Event Viewer when the sub-process start event is expanded.

  Note that the string value in this attribute follows the same rules as the query string used for pre-defined event filters—for more information, see Event Filter String.

You can modify the default query string in this attribute to cause the list of sub-process events to contain whatever you desire.

- **maxEvents** - This attribute contains an integer that specifies the maximum number of events to display for the sub-process when it is expanded. The default is 100.

  If the user expands a sub-process start event that contains more events than the maximum number allowed, an alternative method is to select the “Started instance” event, then select “This instance” from the **Links** menu to create a new event list containing *all* of the events related to the sub-process instance that was started.

sorts Element

The <subProcessLink> element contains a <sorts> element that defines how the sub-process are sorted. It has the following attributes:

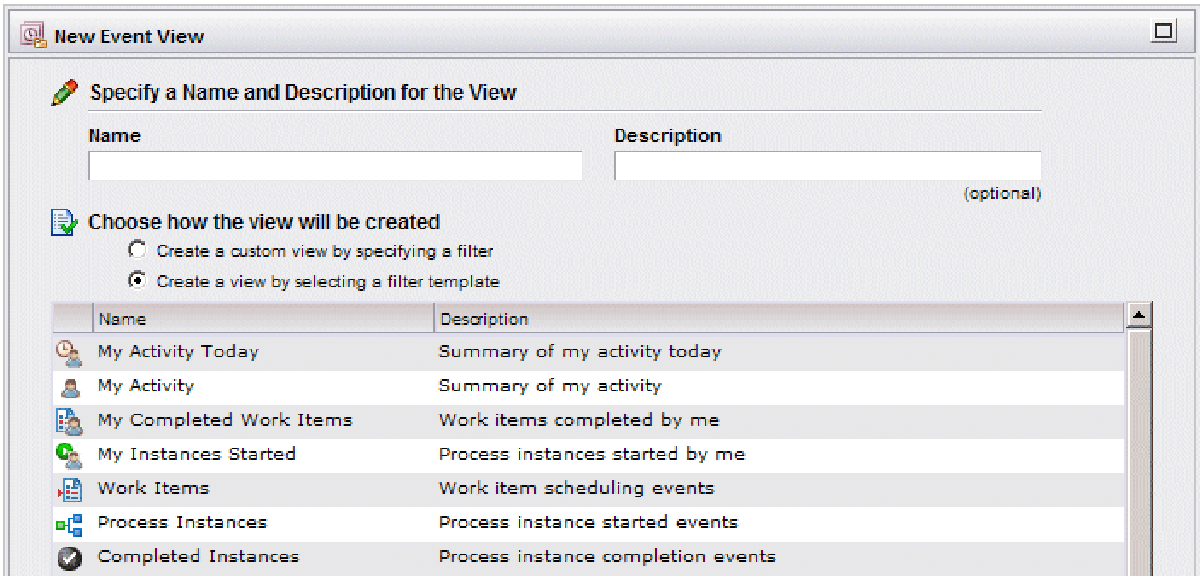
- **attribute** - The name of the event attribute to sort on.
- **order** - The sort order: either ASCENDING or DESCENDING

Customizing View Templates

Event view templates can be used when creating an event view using the wizard. The templates are pre-defined filters that allow the user to quickly create an event view.

Event view templates are specified in the `eventViewTemplates.xml` file. For information about the location of this file, see Location of the Event Configuration Files to Configure.

The `eventViewTemplates.xml` file is composed of a number of <template/> elements, each of which provides a selection on the New Event View dialog when the **Create a view by selecting a filter template** option is selected (see the illustration below).



New Event View

Specify a Name and Description for the View

Name

Description

(optional)

Choose how the view will be created

○ Create a custom view by specifying a filter

● Create a view by selecting a filter template

Name	Description
My Activity Today	Summary of my activity today
My Activity	Summary of my activity
My Completed Work Items	Work items completed by me
My Instances Started	Process instances started by me
Work Items	Work item scheduling events
Process Instances	Process instance started events
Completed Instances	Process instance completion events

You can modify or delete the existing example templates provided in Workspace, or create new templates, using the information provided below.

Each <template/> element in the `eventViewTemplates.xml` file has:

- Six attributes that define the characteristics of the template. These attributes are described below.
- A <filters/> element with one or more subordinate <filter/> elements that specify the pre-defined filters that will be available on the event list that results from an event view being created using a

template. For information about specifying these pre-defined filters, see Specifying Pre-defined Filters.

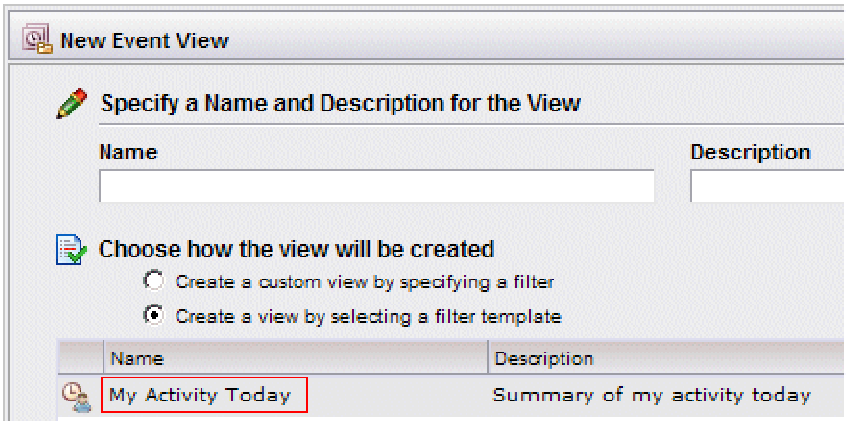
```
<template id="0"
   name="eventViewTemplates.name.myActivityToday"
   image="JSXAPPS/base/application/images/UserToday.gif"
   defaultEventView="true"
   defaultFilterId="eventViewTemplates.id.summary"
   defaultEventDescription="detail">
   <filters>
      <!-- All activity by this resource -->
      <filter id="eventViewTemplates.id.all"
          query="principalName='{sessionResourceName}' AND
                 creationTime = {~TODAY~}"
          description="eventViewTemplates.description.allMyActivityToday"/>
              .
              .
              .
</template>
```

Id Attribute

This attribute is an identifier for the event template record. This must be unique within the `eventViewTemplates.xml` file.

name Attribute

This attribute specifies the text to display in the **Name** column on the New Event View dialog for the template represented by the <template/> element:



New Event View

Specify a Name and Description for the View

Name

Description

Choose how the view will be created

○ Create a custom view by specifying a filter

● Create a view by selecting a filter template

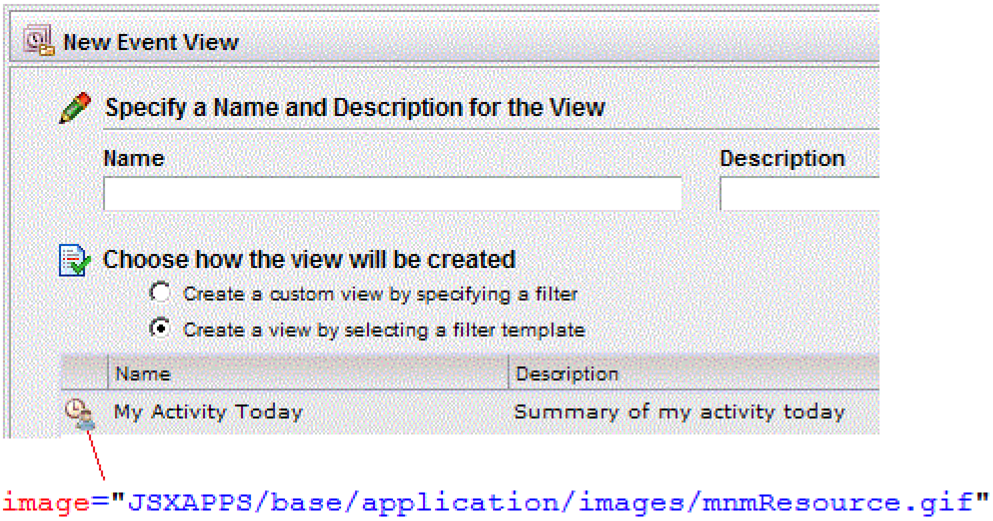
Name	Description
My Activity Today	Summary of my activity today

Note that this text also becomes the default name of the newly created event view, although the user can change it in the **Name** field on the New Event View dialog.

The **name** attribute contains a key that points to the text string in the `eventLocale.xml` file—for more information, see Event-Related Text.

image Attribute

This attribute specifies the image to display to the left of the template name on the New Event View dialog when the user is choosing a template, as well as on the **Event Views** list after creating the event view with the template. For example:



New Event View

Specify a Name and Description for the View

Name

Description

Choose how the view will be created

○ Create a custom view by specifying a filter

● Create a view by selecting a filter template

Name	Description
My Activity Today	Summary of my activity today

image="JSXAPPS/base/application/images/mnmResource.gif"

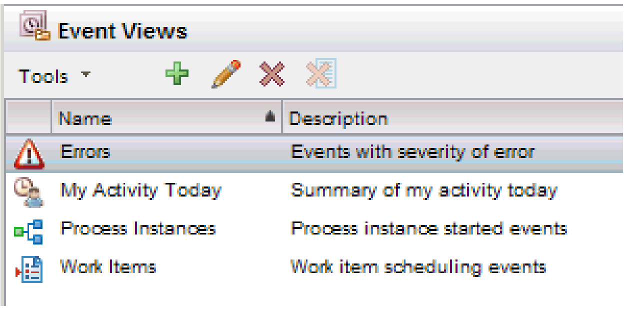
defaultEventView

This attribute specifies whether or not the event view template is used to automatically create an event view that is shown in the user's event view list the first time the user logs in.

By default, four of the pre-defined event view templates have this attribute set to "true". They are:

- Errors
- My Activity Today
- Process Instances
- Work Items

This results in the user's event view list appearing as follows on the first login:



Event Views

Name	Description
Errors	Events with severity of error
My Activity Today	Summary of my activity today
Process Instances	Process instance started events
Work Items	Work item scheduling events

After the initial login, the event view list will contain the event views that were present the last time the user logged out (except for temporary views, which are removed upon logout).

defaultFilterId Attribute

This attribute specifies which of the pre-defined filters specified by the <filter/> element to display by default.

For more information, see Specifying Pre-defined Filters.

defaultEventDescription Attribute

This attribute specifies which of the descriptions (summary or detail) to display in the **description** column by default.

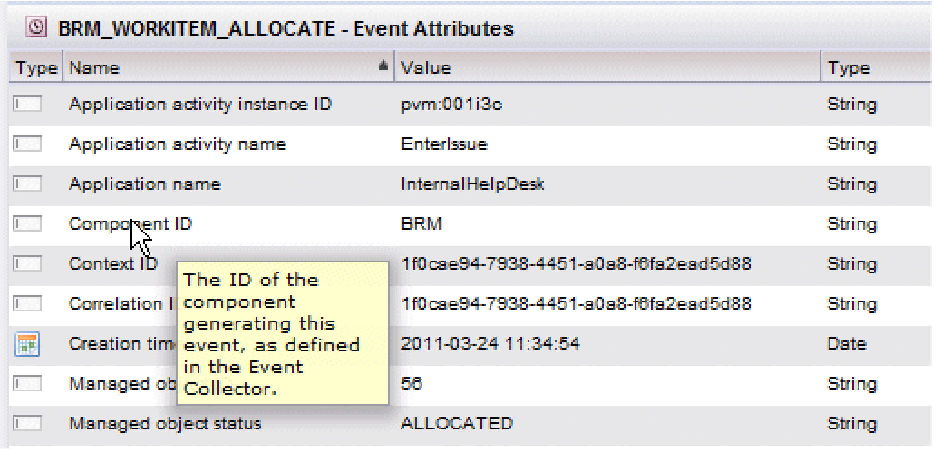
For more information, see Customizing Event Descriptions.

Customizing Event Attributes

An event attribute configuration file is available to customize various aspects of event attributes.

You can customize the following:

- The attribute description that is displayed when the user hovers the mouse pointer over an attribute in the event attribute list:



BRM_WORKITEM_ALLOCATE - Event Attributes

Type	Name	Value	Type
	Application activity instance ID	pvm:001i3c	String
	Application activity name	EnterIssue	String
	Application name	InternalHelpDesk	String
	Component ID	BRM	String
	Context ID	1f0cae94-7938-4451-a0a8-f6fa2ead5d88	String
	Correlation I	1f0cae94-7938-4451-a0a8-f6fa2ead5d88	String
	Creation tim	2011-03-24 11:34:54	Date
	Managed ob	56	String
	Managed object status	ALLOCATED	String

The ID of the component generating this event, as defined in the Event Collector.

- Whether or not an attribute is available in the **Attribute Selector**.
- Whether or not the attribute is shown in the event attribute list by default.

Event attributes are customized using the `eventAttributes.xml` file. For information about the location of this file, see Location of the Event Configuration Files to Configure.

The `eventAttributes.xml` file is composed of a number of <EventAttribute/> elements, one for each possible event attribute in the system. Each <EventAttribute/> element has the following attributes that you can use to customize the event attributes as described above:

- **Name** - The name of the event attribute. Do not change these. Each one identifies a specific event attribute.
- **DisplayName** - This identifies the string that is displayed in the **Name** column for the attribute in the event attribute list. This can be customized to the desired name, or localized if needed.

  The **DisplayName** attribute contains a key that points to the text string in the `eventLocale.xml` file—for more information, see Event-Related Text.
- **Description** - This identifies the string that is displayed when the user hovers the mouse pointer over an attribute in the event attribute list. This can be customized to the desired description, or localized if needed.

  The **Description** attribute contains a key that points to the text string in the `eventLocale.xml` file—for more information, see Event-Related Text.
- **Available** - Set this to “true” if you want the event attribute to appear on the **Attribute Selector**. If you make the attribute available on the Attribute Selector, the user can include it in the event attribute list (if the event attribute applies to the selected event type).

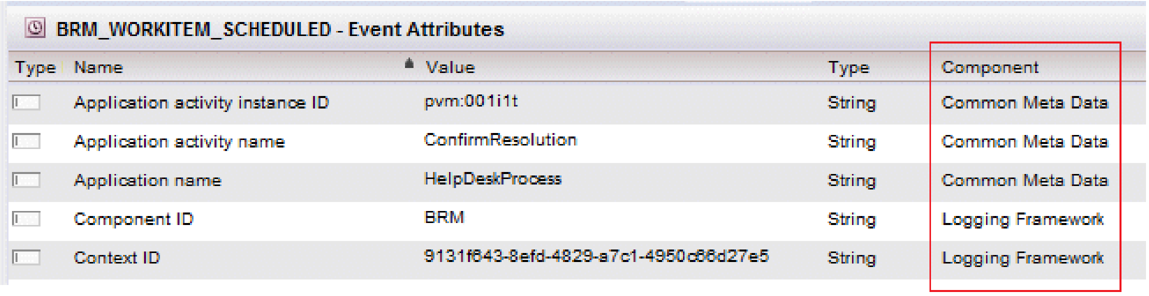
  Set this to “false” if you do not want the event attribute available to the user to include on the event attribute list.
- **Default** - Set this to “true” if you want the event attribute to be displayed in the event attribute list by default (that is, if it applies to the selected event type). Setting this to “true” essentially causes the event attribute to appear on the right side of the Attribute Selector (in the **Selected Attributes** list)

by default—although, the user can move it from the right side to the left side of the Attribute Selector so that the event attribute is not displayed in the event attribute list.

Customizing Event Components

By default, the Event Viewer displays events from specific ActiveMatrix BPM components.

For example:



BRM_WORKITEM_SCHEDULED - Event Attributes

Type	Name	Value	Type	Component
	Application activity instance ID	pvm:001i1t	String	Common Meta Data
	Application activity name	ConfirmResolution	String	Common Meta Data
	Application name	HelpDeskProcess	String	Common Meta Data
	Component ID	BRM	String	Logging Framework
	Context ID	9131f643-8efd-4829-a7c1-4950c66d27e5	String	Logging Framework

The components whose attributes are displayed in the Event Viewer are configured in the `eventComponents.xml`:

```
    <EventComponent Name="N2LF"
                Description="eventComponents.Description.N2LF"/>
    <EventComponent Name="Common"
                Description="eventComponents.Description.Common"/>
    <EventComponent Name="EC"
                Description="eventComponents.Description.EC"/>
    <EventComponent Name="BX"
                Description="eventComponents.Description.BX"/>
    <EventComponent Name="PFE"
                Description="eventComponents.Description.PFE"/>
    <EventComponent Name="BRM"
                Description="eventComponents.Description.BRM"/>
                   .
                   .
                   .
```

Each <EventComponent> element contains the following attributes:

- **Name** - The name of the event component.
- **Description** - A key that corresponds to a record with a matching "id" attribute value in the `JSXAPPS/base/locale/eventLocale.xml` resource file. At runtime, the value of the "text" attribute, for the corresponding record in `eventLocale.xml`, is substituted for this attribute value and becomes the component description that is displayed in the list of attributes.

If a new component has been added to the system, it can be added to Workspace so it appears in the Event Viewer, as described in the following procedure.

Procedure

1. Open the `eventComponents.xml` file for the desired role.
2. Add a new <EventComponentelement> for the new component. For example:

```
<EventComponent Name="WFE"
           Description="eventComponents.Description.WFE"/>
```

3. Save and close the `eventComponents.xml` file.
4. Open the `eventLocale.xml` file—for information about its location, see Event-Related Text.

5. Scroll down to the bottom of the `eventLocale.xml` file and locate the “eventComponents” records.
6. Add a new “eventComponents” record for the new component:
   a) Set the **id** attribute to the value you provided in the **Description** attribute in step 2.
   b) Set the **text** attribute value to the name you want displayed for the component in the Event Viewer.
7. Save and close the `eventLocale.xml` file.
8. Open the `eventAttributes.xml` file and add entries for the component’s attributes.

   For information about adding attributes to the `eventAttributes.xml` file, see Customizing Event Attributes.
9. Save and close the `eventAttributes.xml` file.

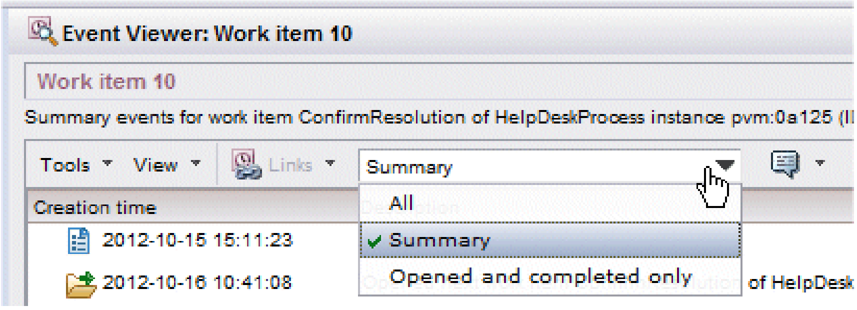
Result

The new component and its attributes will now appear in the Event Viewer.

Specifying Pre-defined Filters

Pre-defined filters are those available to the user from a drop-down list in the Event Viewer.

For example:



Event Viewer: Work item 10

Work item 10

Summary events for work item ConfirmResolution of HelpDeskProcess instance pvm:0a125 (I

Creation time

2012-10-15 15:11:23

2012-10-16 10:41:08 of HelpDesk

Pre-defined filters can be specified in two different areas of events:

- **Links** - When links are defined with <link/> elements in the `eventLinks.xml` file, a <filters/> element is included with one or more subordinate <filter/> elements, each one representing a selection from the pre-defined filter drop-down list for the event list created by that <link/> element.
- **Templates** - When templates are defined with <template/> elements in the `eventViewTemplates.xml` file, a <filters/> element is included with one or more subordinate <filter/> elements, each one representing a selection from the pre-defined filter drop-down list for the event list created by that <template/> element.

Each <filter/> element contains a query string that is used to include or exclude certain event types from the event list when that pre-defined filter is selected.

Both the `eventLinks.xml` and the `eventViewTemplates.xml` files contain default pre-defined filters for each link and template, respectively, which you are free to modify, remove, or add to.

For example, the following specifies three pre-defined filters that are available from an event list created via a work item contextual link (messageId = WorkItem_WorkspaceContext”) when the “This work item” link is selected:

```
<links messageId="WorkItem_WorkspaceContext">
  <link name="eventLinks.name.workItem"
        menuText="eventLinks.menuText.thisWorkItem"
        image="JSXAPPS/base/application/images/WorkView.gif"
        defaultFilterId="eventLinks.id.summary"
        defaultEventDescription="summary">
        <filters>
          <!-- Show all events, including BX user task related events -->
          <filter id="eventLinks.id.all"
             query="applicationActivityInstanceId='{ActivityID}'"
             description="eventLinks.description.allEventsForItemOfInstanceApp"/>
          <!-- Don't show BX related user task events -->
          <filter id="eventLinks.id.summary"
              query="applicationActivityInstanceId='{ActivityID}' AND
                      messageId &lt;&gt; 'BX_INSTANCE_TASKS*'"
              description="eventLinks.description.summaryEventsForItemApp"/>
          <!-- Only show BRM workitem open and complete events -->
          <filter id="eventLinks.id.openedCompletedOnly"
               query="applicationActivityInstanceId='{ActivityID}' AND
                     (messageId='BRM_WORKITEM_OPENED' OR
                      messageId='BRM_WORKITEM_EX_COMPLETE_WORK_ITEM')"
               description="eventLinks.description.openedCompletedForItemApp"/>
         </filters>
</link>
```

The three <filter/> elements cause the “All”, “Summary”, and “Open and completed only” selections to be displayed in the pre-defined filter drop-down list when the event list defined by the <link/> element is displayed.

Note that you can specify which of these filters to use by default by configuring the **defaultFilterId** attribute under the <link/> element (see defaultFilterId Attribute) or the <template/> element (see defaultFilterId Attribute).

The <filter/> element provides the following attributes to define each of the filters:

id Attribute

This attribute identifies the text that is displayed in the pre-defined filter drop-down list to identify the filter.

The **id** attribute contains a key that points to the text string in the `eventLocale.xml` file—for more information, see Event-Related Text.

query Attribute

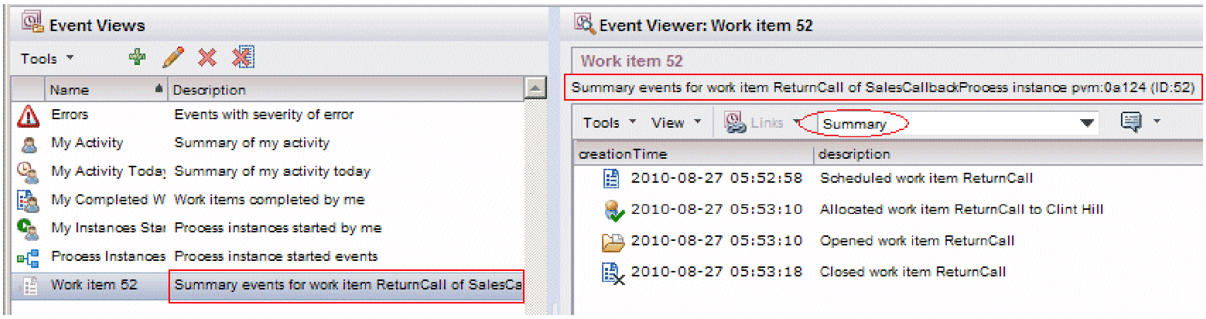
This attribute is a string value used to query the event collector database for events. All events that satisfy the filter string specified in this attribute are returned from the event collector database and displayed in the Event Viewer when the pre-defined filter is selected.

For more information about the filter string in this attribute, see Event Filter String.

description Attribute

This attribute specifies the description to display when the pre-defined filter is selected.

The description is displayed in two locations, as shown in the example below. In this example, when the “Summary” pre-defined filter is selected, it results in the description being displayed in the **Description** column in the event view list, as well as above the toolbar in the Event Viewer:



Name	Description
Errors	Events with severity of error
My Activity	Summary of my activity
My Activity Toda	Summary of my activity today
My Completed W	Work items completed by me
My Instances Sta	Process instances started by me
Process Instances	Process instance started events
Work item 52	Summary events for work item ReturnCall of SalesCa

Summary events for work item ReturnCall of SalesCallbackProcess instance pvm:0a124 (ID:52)

creationTime	description
2010-08-27 05:52:58	Scheduled work item ReturnCall
2010-08-27 05:53:10	Allocated work item ReturnCall to Clint Hill
2010-08-27 05:53:10	Opened work item ReturnCall
2010-08-27 05:53:18	Closed work item ReturnCall

The **description** attribute contains a key that points to the text string in the `eventLocale.xml` file—for more information, see Event-Related Text.

Event Filter String

The <link/> element in the `eventLinks.xml` file, and the <template/> element in the `eventViewsTemplates.xml` file, have at least one subordinate <filter/> element, that in turn contains a **query** attribute, which specifies a string value that is used to query the event collector database for events.

All events that satisfy the filter string specified in the **query** attribute are returned from the event collector database and displayed in the Event Viewer when the user displays an event list represented by the <link/> or <template/> element.

For more information about the <filter/> element and **query** attribute, see Specifying Pre-defined Filters.

This describes how to design the filter string in the **query** attribute.

In the following **query** attribute example, all events are returned that are associated with the application from which the currently selected work item is associated, *and* that are associated with the currently logged-in user:

```
query="applicationName='{AppName}' AND
       principalName='{sessionResourceName}'"
```

The filter string can contain:

- **Event attributes** - These represent values in the events that you are filtering on. In the example above, **applicationName** and **principalName** are event attributes that contain the name of the application associated with the currently selected work item, and the name of the resource (user) associated with the currently selected work item, respectively.

  All available event attributes are listed in the `eventAttributes.xml` file. Note, however, that not all event attributes are applicable to all event types, and not all event attributes are filterable and sortable.

- **Substitution tokens** - These are replaced with attribute values when the link is executed. They provide the values to which you are comparing the values in the event attributes to determine which to return from the event collector database.

  Substitution tokens in the filter string must be enclosed in { } braces, as well as single quotes (see **{AppName}** and **{sessionResourceName}** in the example above).

  For more information about substitution tokens, see Substitution Tokens.

- **Event Message IDs** - These identify the specific event. The ID must be enclosed in single quotes. For example:

```
query="messageId='BRM_WORKITEM_EX_SCHEDULE*'"
```

For a complete list of the available event message IDs, see the **messageId** attributes in the `eventDescriptions.xml` file.

- **Logical and comparative operators** - You can include the following operators in the filter string:

Operator	Description
AND	Logical AND
OR	Logical OR
=	Equal to

Operator	Description
&lt;&gt;	Not equal to
&lt;	Less than
&lt;=	Less than or equal to
&gt;	Greater than
&gt;=	Greater than or equal to

When using the comparative operators, the values being compared must be of the same type.

- **Parentheses** - You can group, in parentheses, attributes/tokens being compared to obtain the desired hierarchy in the filter string.
- **Wildcards** - The following wildcard characters can be used in the filter string:

Wildcard	Description
?	Matches any single character.
*	Matches any number of characters.

Wildcard characters can also be escaped using the \ character, for example:

```
   ‘str\?ng’
```

And the escape character can be escaped, for example:

```
   ‘str\\ng’
```

- Dynamic Time-Period / Point-in-Time Variables - These are special variables that define a date/time range or point-in-time relative to the current date/time *at the point in time when the expression is evaluated*.

For instance, the **~TODAY~** dynamic time-period variable allows you to define a filter that always represents the *current day*.

Dynamic time-period variables in the filter string must be enclosed in { } braces. The following is an example usage of the **~TODAY~** variable:

```
query="principalName='{sessionResourceName}' AND
       creationTime = {~TODAY~}"
```

This variable evaluates to the current date when the filter string is evaluated.

The available dynamic time-period variables are:

- **~TODAY~** - The current day, from 12:00 A.M. to 12:00 P.M.
- **~THISWEEK~** - The current week, from 12:00 A.M. Sunday morning to 12:00 P.M. Saturday evening.
- **~THISMONTH~** - The current month, from the 1st through the last day of the month.
- **~THISQUARTER~** - The current 3-month quarterly period. The specific three months to which this refers depends on how the definition of a quarter is configured. As companies use different months on which to begin a quarter, this may be referring to:
- This month, and the next two months,
- last month, this month, and next month, or

- the previous two months, and this month.

  Which of these periods to use depends on the setting of the **FilterMaskQuarterStart** parameter in the application's `config.xml` file. For more information, see First Quarter Month.

- **~THISYEAR~** - This current year, from January 1 through December 31.

The variables above that represent fixed time periods (~TODAY~, ~THISWEEK~, ~THISMONTH~, ~THISQUATER~, and ~THISYEAR~) can also be extended to increase their range over multiple periods. To do this, use the form:

```
~variable[+-]#~
```

where *variable* is one of the fixed dynamic time-period variables listed above, [+-] indicates whether to include additional periods in the future (+) or in the past (-), and # is a number greater than zero.

This causes the variable to represent the current time period specified by *variable*, plus the additional number of future or previous number periods. Two examples are shown below:

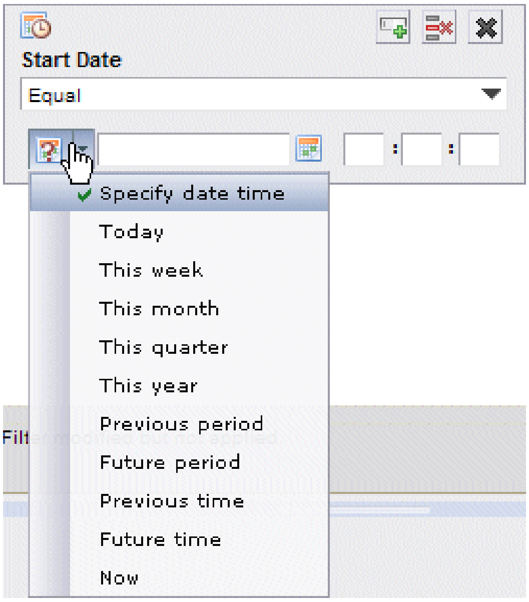
- ~TODAY+2~ - Represents the current day, plus 2 future days.
- ~THISWEEK-1~ - Represents the current week, plus the previous week.

- **~PERIOD-***dd*:*hh*:*mm***~** - A time period that includes the last *dd* days, *hh* hours and *mm* minutes, from the point in time the filter is invoked.
- **~PERIOD+***dd*:*hh*:*mm*~ - A time period that includes the coming *dd* days, *hh* hours, and *mm* minutes, from the point in time the filter is invoked.
- **~POINT**-*dd*:*hh*:*mm***~** - A point in time *dd* days, *hh* hours, and *mm* minutes in the past.
- **~POINT**+*dd*:*hh*:*mm***~** - A point in time *dd* days, *hh* hours, and *mm* minutes in the future.

  The place holder fields *dd*, *hh* and *mm* in the above variables can be assigned values greater than or equal to 0. The syntax must include all place holder fields.

- **~NOW~** - The current date and time.

For a usage example of the dynamic time-period / point-in-time variables, see the filter mask for date fields:



Substitution Tokens

Substitution tokens can be used in a number of attribute string values when customizing events.

They can be used in the following attributes:

- **summary** attribute when customizing the summary description for an event—see Customizing Event Descriptions.
- **detail** attribute when customizing the detail description for an event—see Customizing Event Descriptions.
- **name** attribute when customizing event links—see name Attribute.
- **query** attribute when customizing pre-defined filters for event links and templates (see query Attribute), as well as query strings for defining sub-process links (see Sub-Process Links).
- **description** attribute when customizing pre-defined filters —see description Attribute.

  Substitution tokens that you include in these attributes are replaced with attribute values at runtime.

  When included in an attribute, substitution tokens must be enclosed in { } braces (as well as single quotes in the **query** attribute). For example:

  ```
  query="applicationActivityInstanceId='{ActivityID}' AND
        principalName='{sessionResourceName}'"
  ```

Types of Substitution Tokens

There are three types of substitution tokens—event attributes, application attributes, and context-specific attributes.

- **Event attributes** - These include all of the event attributes that are defined in the `eventAttributes.xml` file. Note, however, only a subset of these attributes are pertinent to any given event type.

  The event attributes can be included as substitution tokens in any of the attributes (**summary**, **detail**, **query**, etc.) in which substitution tokens are allowed.

- **Application attributes** - There are two of these, and they can also be included as substitution tokens in any of the attributes (**summary**, **detail**, **query**, etc.) in which substitution tokens are allowed. The available Application attributes are:

  – **{sessionResourceName}** - Contains the name of the resource currently logged into the application.

  – **{sessionResourceGuid}** - Contains the Guid of the resource currently logged into the application.

- **Context-specific attributes** - These are attributes that contain data that is specific to the type of entity selected when creating contextual events. For example, if creating an event view/list in the context of a work item, the available context-specific attributes contain work item-related information: work item id, start date, state, etc.

  The context-specific attributes can be used as substitution tokens only in the **name**, **query**, and **description** attributes when defining contextual links in the `eventLinks.xml` file.

  The following is a list of the context-specific attributes that can be used as substitution tokens for the various types of contextual links (categorized by **messageId**). Note that the substitution tokens available from work item and process instance-related links relate to the columns shown on the work item and process instances lists, respectively.

  – WorkItem_WorkspaceContext

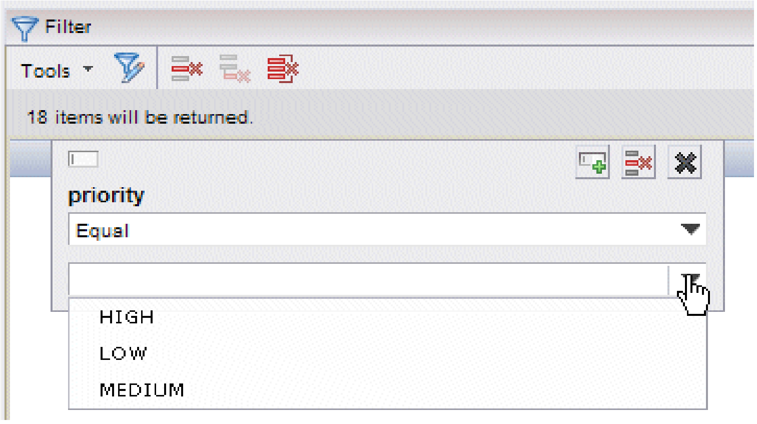
    – {ID} - work item Id

  - {Version} - work item version
  - {Name}
  - {Description}
  - {StartDate}
  - {EndDate}
  - {DistributionStrategy}
  - {Priority}
  - {GroupID}
  - {ActivityID}
  - {AppID}
  - {AppInstance}
  - {AppName}
  - {ScheduleStatus}
  - {State}
  - {WorkTypeID}
  - {WorkTypeUID}
  - {WorkTypeDescription}
- ProcessInstance_WorkspaceContext
  - {INSTANCE.VERSION}
  - {INSTANCE.ID}
  - {INSTANCE.NAME}
  - {MODULE.NAME}
  - {INSTANCE.STATUS}
  - {INSTANCE.START_DATE}
  - {INSTANCE.COMPLETION_DATE}
  - {INSTANCE.WAITING_WORK_COUNT}
- LdapContainer_WorkspaceContext
- LdapContainers_WorkspaceContext
- Group_WorkspaceContext
- Position_WorkspaceContext
- Organization_WorkspaceContext
- OrgUnit_WorkspaceContext
- OrgModelVersion_WorkspaceContext
- ResourceChanges_WorkspaceContext
  - {name} - name of selected resource
  - {guid} - Guid of selected resource

Event Filter Masks

When filtering event lists, the user can select, from a drop-down list, among the possible event attribute values for the selected attribute.

For example, when adding the "Priority" event attribute to the filter expression, the drop-down list on the Filter dialog allows the user to select LOW, MEDIUM, or HIGH:



The masks that are used to provide the drop-down list for each event attribute are defined in the **FilterMaskEventAttributes** record in the application's `config.xml` file. For example, the following mask in `config.xml` provides the drop-down list shown above:

```
<record jsxid="FilterMaskEventAttributes">
   <EventAttribute name="priority">
      <AttributeValue name="LOW"/>
      <AttributeValue name="MEDIUM"/>
      <AttributeValue name="HIGH"/>
   </EventAttribute>
          .
          .
    .
```

To define/extend the set of values in a drop-down list, add **AttributeValue** elements as children of the **EventAttribute** element, where the **name** attribute of the **AttributeValue** element defines the value displayed to the user.

For information about modifying the `config.xml` file to update the event filter masks, see Introduction.

Customizations

You can perform some customization tasks on either the Workspace application or a custom WCC application.

Font Color and Image Settings

There are two configuration files available that allow you to customize the appearance of your application by modifying fonts, colors, and image settings.

Both of these configuration files are located in the following directory:

`StudioHome\wcc\version\JSXAPPS\base\jss\`

where:

- *StudioHome* is the directory in which TIBCO Business Studio was installed.
- *version* is the version number of Workspace that was installed with TIBCO Business Studio.

The two configuration files are:

- `workspaceStyles.css` - This file allows you to control the fonts in menu items, as well as the fonts in the hierarchical matrix for work views, business services, event viewer, and Organization Browser.
- `workspaceCSS.xml` - This file allows you to control many other areas of the application, such as:
  - icon/button images
  - font type, size, color (for fonts in the application not controlled by the `workspaceStyles.css` file)
  - background colors

    In this file, change only the values in the **jsxtext** attributes.

These files are well-commented to indicate the area of the application that will be affected by the configuration.

Adding Custom Menus and Toolbar Buttons

Custom menus and/or toolbar buttons can be added to the Workspace application or your custom WCC application using configuration settings in the application’s configuration file, `config.xml`.

Note that although custom menus and toolbar buttons are configured in the `config.xml` file, adding them to your application involves more than that. You must also modify files in your local WCC development environment, then after modifying and testing it, you can deploy the application to a node (see Deploying an Application After Customizing).

The record element that specifies a custom menu or toolbar button has a **jsxid** attribute value of **customMenus**:

```
<record jsxid="customMenus">
    <!-- custom menu or toolbar elements added here -->
</record>
```

The <record jsxid="customMenus"> element can have either **toolbar** or **menu** child elements, that display custom toolbar buttons and menus, respectively.

Both the **toolbar** and **menu** elements have three attributes:

- **parent** - The name of the list or toolbar on which the custom menu or button is to be displayed. The following values are valid(1):

  - “AuditEventList”
  - "BusinessServiceList"
  - "EventViewList"
  - “MainAppToolbar” (this only applies to the Workspace application, not to custom WCC applications)
  - “ProcessInstanceList”
  - “ProcessTemplateList’
  - "ProcessViewList"
  - "WorkItemList"
  - "WorkViewList"
- **width** - The display width in pixels. Note, however, that this is used only with a parent of "MainAppToolbar". For all other parents, it is ignored.
- **prototype** - The path to the default prototype XML file. This is the General Interface prototype that defines the GUI components for the menu or toolbar.

Both the toolbar and menu elements may also have optional **locale** child elements that define prototype XML files that are localized for specific languages. If a **locale** child element exists, and it corresponds to the language and locale currently selected by the user, the language-specific prototype is loaded, otherwise the default prototype is loaded.

Each **locale** child element has three attributes:

- **localeKey** - Locale identifier. This value must correspond to one of the **key** attribute values defined in the **locale** element of the `JSXAPPS\base\locale\locale.xml` configuration file. The **key** attribute values are of the format: ll or ll_CC, where ll is a lowercase, two- letter ISO 639 language code, and CC is the optional, uppercase, two-letter ISO 3166 country code. For a list of codes, visit these web sites:
  - International Organization for Standardization (ISO):

    http://www.iso.ch/iso/en/ISOOnline.frontpage
  - Language codes:

    http://www.loc.gov/standards/iso639-2/langhome.html
  - Country codes:

    http://www.iso.ch/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/index.html
- **width** - The display width in pixels. Note, however, that this is used only with a parent of "MainAppToolbar". For all other parents, it is ignored.
- **prototype** - The path to the prototype XML file containing language-specific data. This is the General Interface prototype that defines the GUI components for the menu or toolbar.

The following is an example <record jsxid="customMenus"> element that would cause a custom menu and toolbar button to appear on the work item list (parent = "WorkItemList"):

```
<record jsxid="customMenus">
   <menu parent="WorkItemList"
         prototype="customMenusSample/prototypes/menus/MenuSample.xml"></menu>
   <toolbar parent="WorkItemList"
         prototype="customMenusSample/prototypes/toolbars/ToolbarSample.xml"></
```

[3] A custom WCC application can define new **parent** values in **customMenus** and provide its own custom handling for adding the menu or toolbar specified by the new parent name. See the **loadWorkspaceCustomMenu** method in the mixin interface (`JSXAPPS\wcc\application\js\AppMain.js`) for an example of how this is handled for the **MainAppToolbar** value.

```
toolbar>
</record>
```

The **prototype** attribute references the prototype for the custom menu or toolbar button—in the example above, they reference the prototypes provided in the custom menus sample: `MenuSample.xml` and `ToolbarSample.xml` (for more information about the custom menus sample, see Provided Custom Menus Sample).

The prototypes themselves reference the event handlers for the custom buttons and/or menu selections. For example, the `ToolbarSample.xml` prototype executes the `sampleToolbarHandler`, which is located in the `AppMain.js` file in the following directory:

StudioHome`\wcc\`*version*`\JSXAPPS\`*WCCProjectName*`\application\js`

where:

- *StudioHome* is the directory in which TIBCO Business Studio was installed.
- *version* is the version number of Workspace that was installed with TIBCO Business Studio.
- *WCCProjectName* is the name of the General Interface Builder project that contains your custom application. If you are working with the Workspace application, this is "workspace".

Add your custom application code to the appropriate handler in `AppMain.js`.

Provided Custom Menus Sample

When Workspace is installed, a custom menus sample program is installed that contains sample files for adding custom menus and toolbar buttons.

It is located as follows:

StudioHome`\wcc\`*version*`\samples\CustomMenus`

where:

- *StudioHome* is the directory in which TIBCO Business Studio was installed.
- *version* is the version number of Workspace that was installed with TIBCO Business Studio.

The custom menu samples provide prototypes and images that cause two custom menus and two custom toolbar buttons to be displayed on whatever parent you specify in the <**record jsxid="customMenus"**> element of the application's `config.xml` file.

You can replace the prototypes and images in the sample with your own to display the desired menus and/or toolbar buttons. (See the TIBCO General Interface Builder documentation for details on creating custom menu or toolbar prototype files.) You can then add your custom code to the appropriate event handler in `AppMain.js`.

Procedure to Run the Custom Menus Sample

Follow the procedure in this topic to use the custom menus sample to display custom menus and toolbar buttons in a custom WCC application.

Procedure

1. Copy the custom menus samples directory:

   ```
   StudioHome\wcc\version\samples\CustomMenus\customMenusSample
   ```

   ... to the application root directory:

   ```
   StudioHome\wcc\version\JSXAPPS\ProjectName\customMenusSample
   ```

   where:

   - *StudioHome* is the directory in which TIBCO Business Studio was installed.

- *version* is the version number of Workspace that was installed with TIBCO Business Studio.
- *WCCProjectName* is the name of the General Interface Builder project that contains your custom application. If you are working with the Workspace application, this is "workspace".

2. In the application’s `config.xml` file, add **menu** or **toolbar** elements under the <record jsxid=”customMenus”> element.

   For example, to cause the sample custom menus and toolbar buttons to appear on the work item list, add the following:

```
<record jsxid="customMenus">
   <menu parent="WorkItemList"
      prototype="customMenusSample/prototypes/menus/MenuSample.xml"></menu>
   <toolbar parent="WorkItemList"
     prototype="customMenusSample/prototypes/toolbars/ToolbarSample.xml"></
toolbar>
</record>
```

   Note that there are example **menu** and **toolbar** elements in the application’s `config.xml` file that you can use or modify to fit your needs. There are additional examples using all of the available **parent** values in the `\samples\CustomMenus\customMenusSample\config.sample.xml` file.

3. Save and close the application’s `config.xml` file.

4. Add the desired custom application code to the appropriate event handler.

   See **sampleToolbarHandler** and **sampleMenuHandler**, which are located in the `AppMain.js` file in the following directory:

```
StudioHome\wcc\version\JSXAPPS\WCCProjectName\application\js
```

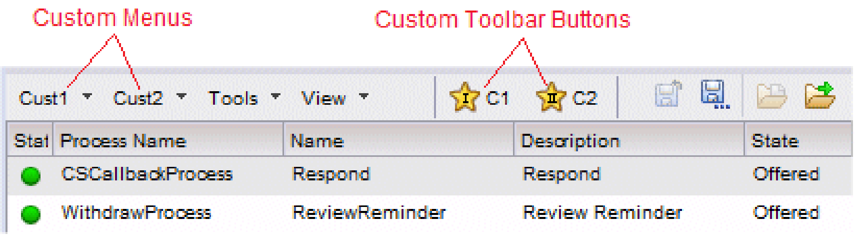
   where:

- *StudioHome* is the directory in which TIBCO Business Studio was installed.
- *version* is the version number of Workspace that was installed with TIBCO Business Studio.
- *WCCProjectName* is the name of the General Interface Builder project that contains your custom application. If you are working with the Workspace application, this is "workspace".

5. Deploy and run your application.

   For information about deploying the application, see Deploying an Application After Customizing.

   The provided custom menus sample causes the following menus and buttons to appear on the work item list:



Stat	Process Name	Name	Description	State
●	CSCallbackProcess	Respond	Respond	Offered
●	WithdrawProcess	ReviewReminder	Review Reminder	Offered

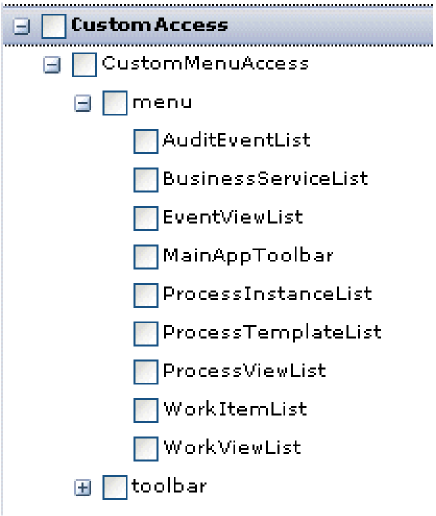
   The custom menu selection and toolbar button event will be handled by the event handlers you modified in step 4.

> To display custom menus or toolbar buttons, each of the major WCC components that display a list (e.g., work item list, process instance list, etc.) call the loadCustomMenu method. This method displays the custom menu or toolbar button if one is defined in the **customMenus** parameter in the `config.xml` file. For more information about the loadCustomMenu method, see the *TIBCO Workspace Components Developer’s Guide.*

Controlling User Access to Custom Menus and Toolbar Buttons

Access to custom menus and toolbar buttons is controlled using user access sets, just like all of the other functions in the application.

In the Configuration Administrator graphical editor, **CustomAccess** selections are provided that allows you to grant access to each possible custom menu or toolbar button:



And in the `userAccess.xml` file (available either through the Configuration Administrator or directly in the file—see Configuring Access Sets by Editing XML), **CustomMenuAccess access** elements are provided to control access to custom menus and toolbar buttons:

```
<access name="CustomAccess">
   <access name="CustomMenuAccess">
      <access name="menu">
         <access name="AuditEventList"></access>
         <access name="BusinessServiceList"></access>
         <access name="EventViewList"></access>
         <access name="MainAppToolbar"></access>
         <access name="ProcessInstanceList"></access>
         <access name="ProcessTemplateList"></access>
         <access name="ProcessViewList"></access>
         <access name="WorkItemList"></access>
         <access name="WorkViewList"></access>
      </access>
      <access name="toolbar">
         <access name="AuditEventList"></access>
         <access name="BusinessServiceList"></access>
         <access name="EventViewList"></access>
         <access name="MainAppToolbar"></access>
         <access name="ProcessInstanceList"></access>
         <access name="ProcessTemplateList"></access>
         <access name="ProcessViewList"></access>
         <access name="WorkItemList"></access>
         <access name="WorkViewList"></access>
      </access>
   </access>
</access>
```

Each graphical editor **CustomAccess** check box, and `userAccess.xml` file **CustomMenuAccess access** element, controls access to a custom menu and/or toolbar button on a particular list or toolbar.

Either check the box in the graphical editor, or include the **access** element to grant access to custom menus / toolbar buttons on the list or toolbar.

For more information about specifying user access, see Configuring User Access.

Controlling Access to Subordinate Items

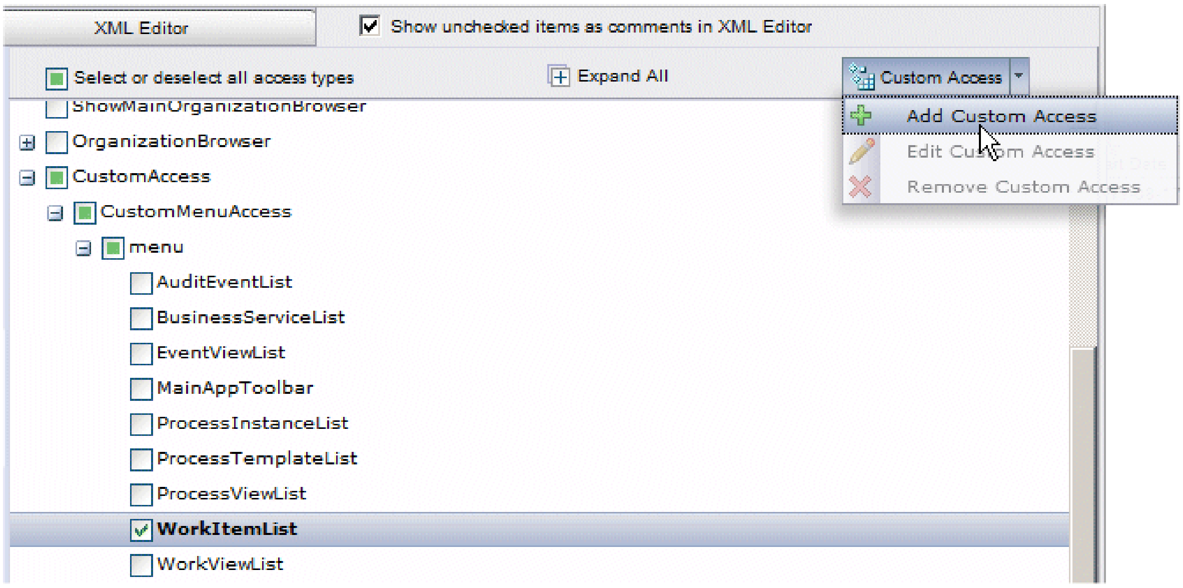
You can control access to subordinate items on custom menus and toolbar buttons, either by using the graphical editor or by adding **access** elements to the XML.

Using the Graphical Editor

The graphical editor can be used to control access to subordinate items on custom menus and toolbar buttons.

Procedure

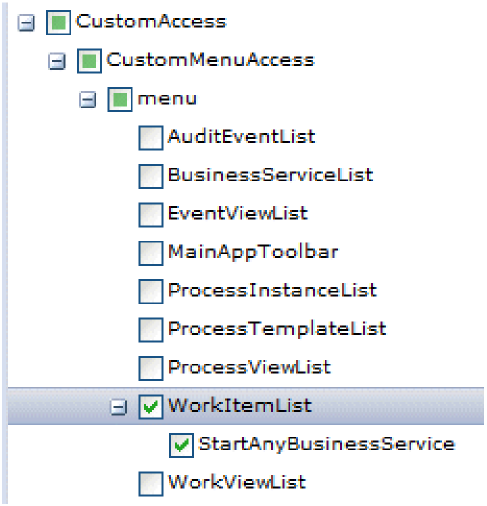
1. Open the graphical editor in the Configuration Administrator and select the desired user access set (see Configuring User Access Sets Using the Graphical Editor).
2. In the **Custom Access** check boxes, check the box for the list that contains the custom menu or toolbar button.
3. From the **Custom Access** button drop-down list, select **Add Custom Access**:



This causes the Add Custom Access dialog to display.

4. In the **Add Custom Access** dialog, enter the name of the subordinate custom menu/toolbar item.

This causes the subordinate item to be added under the access control. For example:



- ⊟ 🟩 CustomAccess
  - ⊟ 🟩 CustomMenuAccess
    - ⊟ 🟩 menu
      - ☐ AuditEventList
      - ☐ BusinessServiceList
      - ☐ EventViewList
      - ☐ MainAppToolbar
      - ☐ ProcessInstanceList
      - ☐ ProcessTemplateList
      - ☐ ProcessViewList
      - ⊟ ☑ WorkItemList
        - ☑ StartAnyBusinessService
      - ☐ WorkViewList

Result

Once you've added a subordinate item, you can also edit or remove the item using selections on the **Custom Access** button drop-down list.

Note that a sample application is provided that illustrates creating subordinate items for custom menus and toolbar buttons, as well as controlling access to those items; see User Access Sample Application.

Using Access Elements in the XML

Access elements in the XML can be used to control access to subordinate items on custom menus and toolbar buttons.

Procedure

1. Open the `userAccess.xml` file, either in the Configuration Administrator or in the file system, depending on whether the application has been deployed or not (for more information, see Configuring Access Sets by Editing XML).
2. In the desired user access set, add an **access** element, for the subordinate item, under the access control for the custom menu or toolbar button. For example:

Result

```
<access name="CustomAccess">
   <access name="CustomMenuAccess">
      <access name="menu">
         <access name="WorkItemList">
             <access name="StartAnyBusinessService"></access>
         </access>
      </access>
```

Note that a sample application is provided that illustrates creating subordinate items for custom menus and toolbar buttons, as well as controlling access to those items; see User Access Sample Application.

loadCustomMenu Override Method

A **loadCustomMenu** override method is provided in the Application class that calls the **isAuth** method to determine whether or not to display custom menus and/or toolbar buttons in the application.

The **loadCustomMenu** method is located in the `Application.js` file, which is located in the following directory

`StudioHome\wcc\version\JSXAPPS\WCCProjectName\application\js`

where:

- *StudioHome* is the directory in which TIBCO Business Studio was installed.
- *version* is the version number of Workspace that was installed with TIBCO Business Studio.
- *WCCProjectName* is the name of the General Interface Builder project that contains your custom application. If you are working with the Workspace application, this is "workspace".

Customizing Columns in Lists

You can customize the columns in lists that are displayed in WCC applications (including the Workspace application).

Columns can be customized in the following ways:

- Specify the columns that are displayed by default, as well as the order in which they are displayed, and the default width of each column—see Specifying Default Columns.
- Specify which columns appear in the Column Selector for each list type, making them available or unavailable for users to add or remove from their lists—see Specifying Columns Available in the Column Selector.
- Specify whether or not the user can reorder (move) or resize columns—see Restricting Reordering and Resizing of Columns.

There are also user access controls that allow you to control a user’s ability to manipulate (create, edit, remove, and save) views—see the **NewView**, **EditView**, **RemoveView**, **SaveView**, and **SaveViewAs** controls for the various types of views in Configuring User Access.

You can control a user's ability to add, modify or remove columns in a list using the **SelectColumns** user access control on the appropriate list. This is also described in Configuring User Access.

Specifying Default Columns

A number of lists in Workspace contain default columns.

They are:

- Work view list
- Work item list
- Business service list
- Process view list
- Process instance list
- Event view list
- Event Viewer (event list)
- Process template list(1)
- Process templateEx list

You can modify the columns that are displayed by default in these lists by modifying the following corresponding files in the `...\JSXAPPS\base\components\xml\` directory:

- `WorkViewDefaultColumns.xml`
- `WorkItemDefaultColumns.xml`

[4] The process template list is displayed when using the Start Process Instance function, whereas the process template Ex list is displayed when creating a process view with the wizard.

- BusinessServiceDefaultColumns.xml
- ProcessViewDefaultColumns.xml
- ProcessInstanceDefaultColumns.xml
- EventViewDefaultColumns.xml
- AuditEventDefaultColumns.xml
- ProcessTemplatesDefaultColumns.xml
- ProcessTemplateExDefaultColumns.xml

As an example, the following shows the default contents of the WorkItemDefaultColumns.xml file:

```
<columns>
    <column width="24" id="StateImage" />
    <column width="120" id="AppName" />
    <column width="120" id="Name" />
    <column width="120" id="Description" />
    <column width="56" id="State" />
    <column width="120" id="AppInstanceDescription" />
    <column width="110" id="AppInstance" />
    <column width="40" id="Priority" />
    <column width="130" id="StartDate" />
    <column width="24" id="DeadlineImage" />
    <column width="130" id="EndDate" />
    <column width="40" id="ID" />
</columns>
```

This file defines which columns on the work item list to display by default (and their associated default column width in pixels). The order they are defined in this file determines the order they are shown in the list by default. The value of the **id** attribute must correspond to the name of the field as it is known on the server.

The values in ...\JSXAPPS\base\components\xml\listTypeDisplayFields.xml provide a reference of the available field names (in the **jsxid** attributes), where *listType* indicates the type of list (for example "WorkItem").

The location of the files that are modified to implement this customization depends on whether you are customizing a deployed Workspace application or a custom WCC application, as follows:

- If you are customizing a deployed Workspace application, see Location of Files on a BPM Runtime Machine for information about the location of the files and directories mentioned above.
- If you are using a deployed custom WCC application, you must make the appropriate changes to the files in your WCC development environment, as the files on the BPM runtime machine are not accessible. Therefore, you must make the changes, then redeploy the application. (Or if the application had not been deployed yet, make the changes before deploying the application.)

Bear in mind that changing *listType*DefaultColumns.xml will only change the default set of columns. If a user has made changes to the columns on a list prior to the changes made in *listType*DefaultColumns.xml, they won't see the effect of those changes (since they are not using the default columns, but rather their own persisted set of columns). Users that have not made any column changes of their own will see the new default. The Column Selector tool, available from each list, has a **Use Default** button to revert the list of columns to the defaults defined in the *listType*DefaultColumns.xml file.

Specifying Columns Available in the Column Selector

You can specify which columns are available to users to add to, or remove from, lists via the Column Selector.

The records in ...\JSXAPPS\base\components\xml\listTypeDisplayFields.xml define the fields that are available in the Column Selector and may be selected for display in a list (where *listType* indicates the type of list, for example "WorkItem"). Removing records from this file makes the corresponding fields unavailable for users to display as columns in the list.

The following are the attributes available in records in the `listType`DisplayFields.xml file:

- **`index`** - A unique identifier for the record.
- **`jsxid`** - The name of the field.
- **`jsximg`** - The path to an image that corresponds to the data type of the field.
- **`jsxtext`** - The text that identifies the field in the Column Selector.
- **`header`** - The text that displays in the column header of the list.
- **`defaultwidth`** - The initial width of the column, in pixels.
- **`find`** - If set to "no", the column is not available as a searchable field in the Find tool of the list. If "yes", or not specified, the column is searchable.
- **`jsxdatatype`** - The type of sort to perform when the list is sorted by this column, either "number" or "text" (default if not specified).
- **`tooltip`** - Popup help text that displays when the cursor hovers over the column header. If not specified, the value of the header attribute is used.

The **`jsxtext`**, **`header`**, and **`tooltip`** attributes may be localized by enclosing a key (**`jsxid`** attribute), from `...\JSXAPPS\base\locale\locale.xml`, in brackets (for example, **`header="{txtName}"`**).

Note that if a field is available to users in the Column Selector, and they select the field for display as a column in a list, subsequent removal of the field from *`listType`*`DisplayFields.xml` does not remove the column from the list, it only removes it from the list of available columns in the Column Selector. If this occurs, users are no longer able to remove the column from their list because it is not available in the Column Selector. For them to be able to remove it from their list, the record for the column would need to be added back into the *`listType`*`DisplayFields.xml` file so that it is available in the Column Selector.

The location of the files that are modified to implement this customization depends on whether you are customizing a deployed Workspace application or a custom WCC application, as follows:

- If you are customizing a deployed Workspace application, see Location of Files on a BPM Runtime Machine for information about the location of the files and directories mentioned above.
- If you are using a deployed custom WCC application, you must make the appropriate changes to the files in your WCC development environment, as the files on the BPM runtime machine are not accessible. Therefore, you must make the changes, then redeploy the application. (Or if the application had not been deployed yet, make the changes before deploying the application.)

Restricting Reordering and Resizing of Columns

You can prevent users from reordering (moving) and resizing columns on a list.

This can be configured for the following lists:

- Work view list
- Work item list
- Business service list
- Process view list
- Process instance list
- Event view list
- Event Viewer (event list)
- Process template list(2)
- Process templateEx list

To configure reordering and resizing of columns, you must add and set a property in the appropriate XML prototype (`listDefault.xml`) for the desired list. You can find the `listDefault.xml` file for the desired list in an appropriately named subdirectory of the following directory:

```
...\JSXAPSS\base\components\ListContainer\prototypes\
```

For example, to configure reordering and resizing of columns in the work item list, modify the following file:

```
...\JSXAPSS\base\components\ListContainer\prototypes\WorkItem\listDefault.xml
```

The specific location of the files that are modified to implement this customization depends on whether you are customizing a deployed Workspace application or a custom WCC application, as follows:

- If you are customizing a deployed Workspace application, see Location of Files on a BPM Runtime Machine for information about the location of the files and directories mentioned above.
- If you are using a deployed custom WCC application, you must make the appropriate changes to the files in your WCC development environment, as the files on the BPM runtime machine are not accessible. Therefore, you must make the changes, then redeploy the application. (Or if the application had not been deployed yet, make the changes before deploying the application.)

After making any of the changes described here, you may need to clear your browser cache for the changes to take effect.

The following customizations can be performed. Note that these customizations apply to the entire list for all users who log into the application you modify.

Turn reordering of columns on or off

Procedure

1. Open the appropriate `listDefault.xml` file (as described above).
2. Locate the <object type="jsx3.gui.Matrix"> element.
3. In the <variants> sub-element, add a **jsxreorder** attribute and set its value to "0" to turn off column reordering or "1" to turn on column reordering:

   ```
   <object type="jsx3.gui.Matrix">
   <variants jsxpagingmodel="3" jsxselectionmodel="2" jsxscalewidth="0"
   jsxheaderheight="18" jsxrowheight="20" jsxreorder="0"></variants>
   ```
4. Save and colse the `listDefault.xml` file.

Fix some columns and allow reordering of others

Procedure

1. Open the appropriate `listDefault.xml` file (as described above).
2. Locate the <object type="jsx3.gui.Matrix"> element.
3. In the <variants> sub-element, add a **jsxfixedcolumnindex** attribute and set its value to the number of columns you would like fixed—all higher columns are reorderable. For example, if you set **jsxfixedcolumnindex="3"**, columns 1, 2, and 3 will be fixed, and columns 4 and higher will be reorderable:

[5] The process template list is displayed when using the Start Process Instance function, whereas the process templateEx list is displayed when creating a process view with the wizard.

If neither **jsxreorder** nor **jsxfixedcolumnindex** are present in the `listDefault.xml` file, reordering is turned on by default.

```
<object type="jsx3.gui.Matrix">
<variants jsxpagingmodel="3" jsxselectionmodel="2" jsxscalewidth="0"
jsxheaderheight="18" jsxrowheight="20" jsxfixedcolumnindex="3"></
variants>
```

4. Save and colse the `listDefault.xml` file.

Turn resizing of columns on or off

Procedure

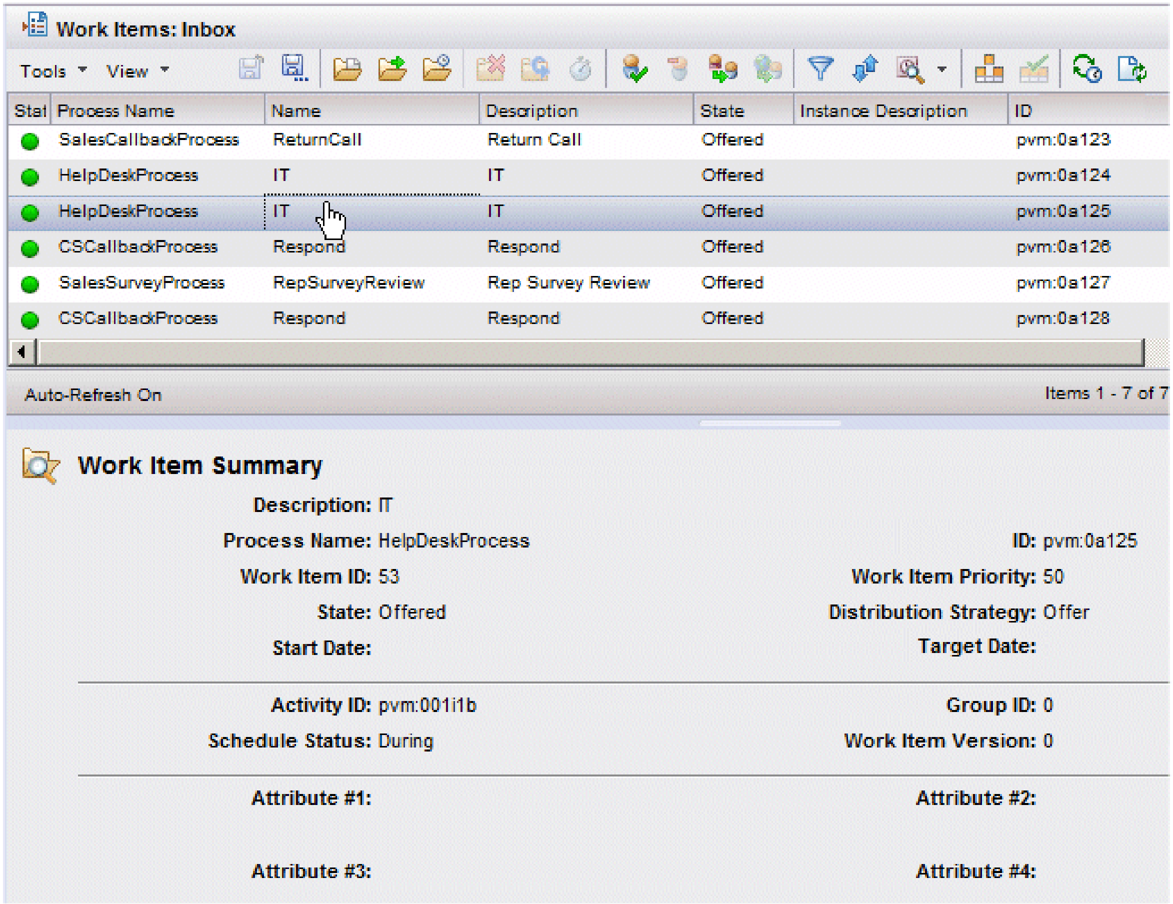
1. Open the appropriate `listDefault.xml` file (as described above).
2. Locate the <object type="jsx3.gui.Matrix"> element.
3. In the <variants> sub-element, add a **jsxresize** attribute and set its value to "0" to turn off column resizing or "1" to turn on column resizing (column reordering is on by default if the **jsxresize** attribute is absent):

```
<object type="jsx3.gui.Matrix">
<variants jsxpagingmodel="3" jsxselectionmodel="2" jsxscalewidth="0"
jsxheaderheight="18" jsxrowheight="20" jsxreorder="0" jsxresize="0"></variants>
```

4. Save and close the `listDefault.xml` file.

Customizing the Work Item and Process Instance Summaries

The **Work Item Summary** is displayed in the preview pane (which must be turned on) when a work item is selected (single click) in the work item list.



Stat	Process Name	Name	Description	State	Instance Description	ID
●	SalesCallbackProcess	ReturnCall	Return Call	Offered		pvm:0a123
●	HelpDeskProcess	IT	IT	Offered		pvm:0a124
●	HelpDeskProcess	IT	IT	Offered		pvm:0a125
●	CSCallbackProcess	Respond	Respond	Offered		pvm:0a126
●	SalesSurveyProcess	RepSurveyReview	Rep Survey Review	Offered		pvm:0a127
●	CSCallbackProcess	Respond	Respond	Offered		pvm:0a128

Auto-Refresh On

Items 1 - 7 of 7

Work Item Summary

Description: IT

Process Name: HelpDeskProcess

ID: pvm:0a125

Work Item ID: 53

Work Item Priority: 50

State: Offered

Distribution Strategy: Offer

Start Date:

Target Date:

Activity ID: pvm:001i1b

Group ID: 0

Schedule Status: During

Work Item Version: 0

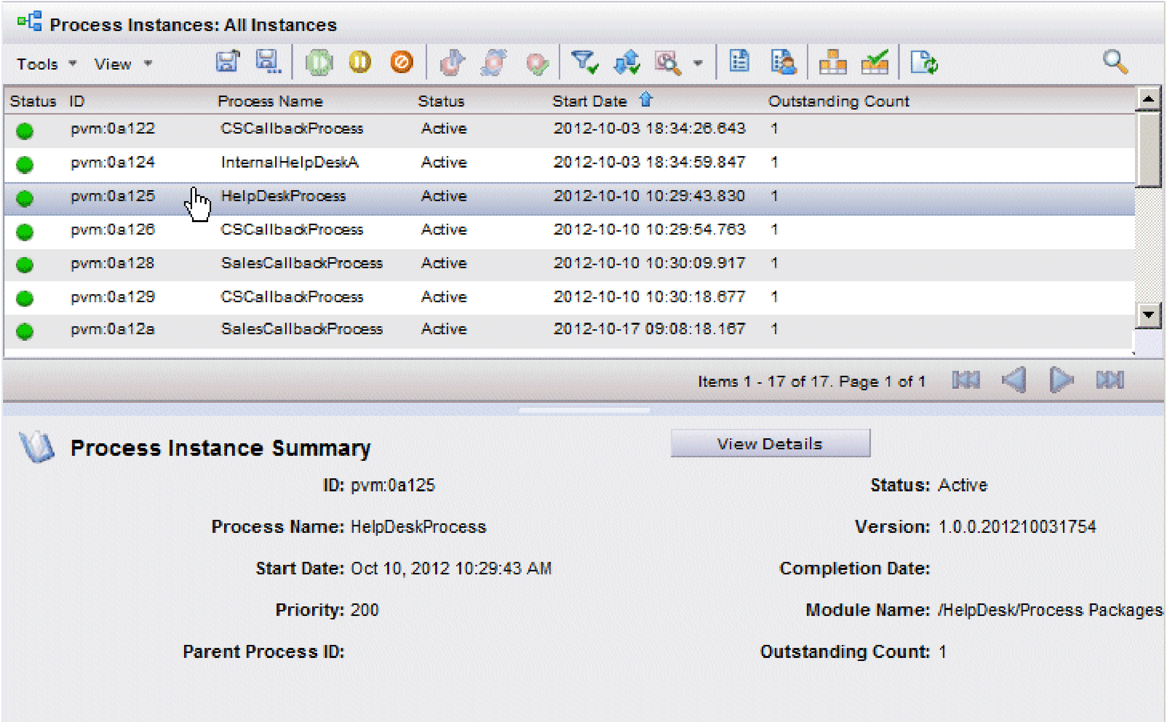
Attribute #1:

Attribute #2:

Attribute #3:

Attribute #4:

The **Process Instance Summary** is displayed in the preview pane (which must be turned on) when a process instance is selected (single click) in the process instance list:



Status	ID	Process Name	Status	Start Date	Outstanding Count
●	pvm:0a122	CSCallbackProcess	Active	2012-10-03 18:34:26.643	1
●	pvm:0a124	InternalHelpDeskA	Active	2012-10-03 18:34:59.847	1
●	pvm:0a125	HelpDeskProcess	Active	2012-10-10 10:29:43.830	1
●	pvm:0a126	CSCallbackProcess	Active	2012-10-10 10:29:54.763	1
●	pvm:0a128	SalesCallbackProcess	Active	2012-10-10 10:30:09.917	1
●	pvm:0a129	CSCallbackProcess	Active	2012-10-10 10:30:18.677	1
●	pvm:0a12a	SalesCallbackProcess	Active	2012-10-17 09:08:18.167	1

Items 1 - 17 of 17. Page 1 of 1

Process Instance Summary

ID: pvm:0a125

Status: Active

Process Name: HelpDeskProcess

Version: 1.0.0.201210031754

Start Date: Oct 10, 2012 10:29:43 AM

Completion Date:

Priority: 200

Module Name: /HelpDesk/Process Packages

Parent Process ID:

Outstanding Count: 1

You can do the following:

- Remove any or all information that is displayed in the summaries by default.
- Reorganize the information.
- Modify the appearance of the information (font type, size, etc.).
- Add anything to the summaries that can be accomplished through HTML.

Modifying the Work Item Summary

You can modify the Work Item Summary that is displayed in the preview pane.

This is done by changing the following file:

```
/JSXAPPS/base/components/ListContainer/prototypes/WorkItem/WorkItemSummary.xml
```

The specific location of this file depends on whether you are modifying a deployed or an undeployed application. For information, see Introduction.

The `WorkItemSummary.xml` file contains a series of <**span**> element pairs, each pair representing an item of information in the Work Item Summary:

```
<div style="position:relative;width:100%;height:
100%;overflow:auto;display:           block;background-color:#ececee;">
    <span style="position:absolute;left:8px;top:7px;">#IMAGE#</span>
    <span style="position:absolute;width:692px;height:18px;left:40px;top:
12px;         font-size:14px;font-family:Arial;font-weight:bold;">#TITLE#</span>
    <span style="position:absolute;width:196px;height:16px;left:8px;top:
35px;         font-size:11px;font-family:Arial;font-weight:bold;text-align:right;">
        #LDESC#</span>
    <span style="position:absolute;width:524px;height:16px;left:208px;top:
35px;         font-size:11px;font-family:Arial;font-weight:normal;">#DESC#</span>
    <span style="position:absolute;width:196px;height:16px;left:
8px;top:         75px;font-size:11px;font-family:Arial;font-weight:bold;text-
align:right;         ">#LITEMID#</span>
    <span style="position:absolute;width:184px;height:16px;left:208px;top:
75px;         font-size:11px;font-family:Arial;font-weight:normal;">#ITEMID#</span>
```

```
        .
        .
        .
</div>
```

Each pair of <**span**> elements contains tokens that represent a label and a work item property (note that the tokens in the first two <**span**> elements, IMAGE and TITLE, provide the header information in the summary).

The label tokens (those that begin with an "L", for example, LDESC) are substituted at runtime with text from the appropriate localization file, depending on the language chosen. The table below provides the name of the variable in the localization file (`locale.xml` for English) for each label. For information about the names and location of the localization files, see Localization.

The work item property tokens in the <**span**> elements (for example, DESC) are substituted at runtime with the appropriate property values for the selected work item.

You can remove or comment out the desired <**span**> elements to remove information from the summary, change the look and feel by modifying the values in the **style** attributes, or add any valid HTML (the `WorkItemSummary.xml` file is simply an HTML file, even though its extension is xml).

Note, however, that you *must not* remove nor comment out the <**div**> elements in the `WorkItemSummary.xml` file. To remove all information from the summary, remove all <**span**> elements, but retain the top-level <**div**> elements.

The table below lists the labels shown in the Work Item Summary, the variable from the localization file used to display the label text in the summary, and the substitution tokens representing the information:

Label in Summary	Text Variable in Localization File	Substitution Tokens
Description	txtDescription	LDESC/DESC
Process Name	txtInstanceName	LAPPNAME/ APPNAME
ID	txtInstanceID	LAPPINSTANCE/ APPINSTANCE
Work Item ID	txtWorkItemID	LITEMID/ITEMID
Work Item Priority	txtWorkItemPriority	LPRIORITY/ PRIORITY
Distribution Strategy	txtDistributionStrategy	LDISTSTRATEGY/ DISTSTRATEGY
Start Date	txtStartDate	LSTART/START
Target Date	txtEndDate	LEND/END
Activity ID	txtActivityID	LACTIVITY/ ACTIVITY
Group ID	txtGroupID	LGROUPID/ GROUPID
Schedule Status	txtScheduleStatus	LSCHEDULE/ SCHEDULE

Label in Summary	Text Variable in Localization File	Substitution Tokens
Work Item Version	txtWorkItemVersion	LVERSION/ VERSION
Attribute #1	txtAttribute1	LATTR1/ATTR1
.	.	.
.	.	.
.	.	.
Attribute #14	txtAttribute14	LATTR14/ATTR14

Modifying the Process Instance Summary

You can modify the Process Instance Summary that is displayed in the preview pane.

This is done by changing the following file:

```
/JSXAPPS/base/components/ListContainer/prototypes/ProcessInstance/
InstanceSummary.xml
```

The specific location of this file depends on whether you are modifying a deployed or an undeployed application. For information, see Introduction).

The `InstanceSummary.xml` file contains a series of <**span**> element pairs, each pair representing an item of information in the Process Instance Summary:

```
<div style="position:relative;width:100%;height:
100%;overflow:auto;display:            block;background-color:#ececee;">
    <span style="position:absolute;left:8px;top:7px;">#IMAGE#</span>
    <span style="position:absolute;width:692px;height:18px;left:40px;top:
12px;         font-size:14px;font-family:Arial;font-weight:bold;">#TITLE#</span>
     <span style="position:absolute;width:196px;height:16px;left:8px;top:
35px;         font-size:11px;font-family:Arial;font-weight:bold;text-
align:         right;">#LID#</span>
     <span style="position:absolute;width:140px;height:16px;left:208px;top:
35px;         font-size:11px;font-family:Arial;font-weight:normal;">#ID#</span>
     <span style="position:absolute;width:208px;height:16px;left:348px;top:
35px;         font-size:11px;font-family:Arial;font-weight:bold;text-
align:         right;">#LSTATUS#</span>
     <span style="position:absolute;width:152px;height:16px;left:560px;top:
35px;         font-size:11px;font-family:Arial;font-weight:normal;">#STATUS#</span>
       .
       .
       .
</div>
```

Each pair of <**span**> elements contains tokens that represent a label and a process instance property (note that the tokens in the first two <**span**> elements, IMAGE and TITLE, provide the header information in the summary).

The label tokens (those that begin with an "L", for example, LSTATUS) are substituted at runtime with text from the appropriate localization file, depending on the language chosen. The table below provides the name of the variable in the localization file (`locale.xml` for English) for each label. For information about the names and location of the localization files, see Localization.

The process instance property tokens in the <**span**> elements (for example, STATUS) are substituted at runtime with the appropriate property values for the selected process instance.

You can remove or comment out the desired <**span**> elements to remove information from the summary, change the look and feel by modifying the values in the **style** attributes, or add any valid HTML (the `InstanceSummary.xml` file is simply an HTML file, even though its extension is xml).

Note, however, that you *must not* remove nor comment out the <**div**> elements in the `InstanceSummary.xml` file. To remove all information from the summary, remove all <**span**> elements, but retain the top-level <**div**> elements.

The table below lists the labels shown in the Process Instance Summary, the variable from the localization file used to display the label text in the summary, and the substitution tokens representing the information:

Label in Summary	Text Variable in Localization File	Substitution Variable
ID	txtInstanceID	LID/ID
Status	txtInstanceStatus	LSTATUS/STATUS
Process Name	txtInstanceName	LINSTANCENAME/ INSTANCENAME
Version	txtInstanceVersion	LVERSION/ VERSION
Start Date	txtInstanceStartDate	LSTART/START
Completion Date	txtInstanceCompletionDate	LEND/END
Priority	txtInstancePriority	LPRIORITY/ PRIORITY
Module Name	txtModuleName	LMODNAME/ MODNAME
Parent Process ID	txtInstanceParentProcessID	LPARENTID/ PARENTID
Outstanding Count	txtInstanceWaitingWorkCount	LOUTCOUNT/ OUTCOUNT

Custom Interfaces

Custom interfaces allow custom menus and toolbar buttons to be displayed in a WCC application.

Custom menus and toolbar buttons can display:

- web pages in iFrames,
- web pages in TIBCO PageBus Managed Hub iFrames, or
- TIBCO General Interface prototypes.

Custom interfaces can be used to embed external web applications into Workspace, or into a custom WCC application. The embedding can be a static configuration, such as configuring an iFrame in Workspace to host the TIBCO tibbr interface, or a customer's web portal application. It can also be dynamic, where an action taken in Workspace, such as selecting a work item, causes the external web application to take some action based on that specific work item, or vice versa. This dynamic interaction is facilitated either by custom script in Workspace that uses the data of the selected item to manipulate the URL of the external web application, or by exchanging PageBus events whose payload contains the information about the action taken in Workspace, or the action to be taken by Workspace. The custom interface mechanism also allows for the embedding of custom TIBCO General Interface prototypes into the Workspace application.

The custom interface mechanism hides, as much as possible, the details of the management of these embedded UIs within the Workspace/WCC application. Any custom script that needs to be developed only needs to focus on manipulating the external web application or taking the action triggered by the external application and does not need to manage how the embedded display gets handled in Workspace. The custom interface infrastructure largely shields the implementer from these details. Consequently, very little (or no) knowledge of TIBCO General Interface is required to integrate custom interfaces into Workspace.

User access to custom interfaces is managed in the Configuration Administrator. This allows you to integrate custom user interfaces into your application via configuration (rather than writing code) as much as practically possible.

Custom interface samples are also included in the WCC distribution; these samples demonstrate the various custom interface types and launch mechanisms.

Custom Interface Use Cases

This topic presents some use cases that can be achieved with the custom interface mechanism.

This is a representative list of possible use cases, but is not meant to be an exhaustive list:

- Embed the TIBCO tibbr web interface into the master selection area of Workspace for general usage by the user. If a user selects a work item or process instance, automatically navigate the embedded tibbr interface to a filtered set of topics relating to the process instance or process template of the selected item, then display the tibbr interface.
- Select a work item where one of the work item attributes contains the customer ID. Provide a toolbar button on the work item list that, when selected, shows a report from an external customer website that displays the order history for that customer. The display of the external website is embedded in the Workspace application and temporarily overlays the work item list, or is displayed in a separate dialog in Workspace.
- Provide a toolbar button on the work item list that, when clicked, uses the value of one of the work item attributes of the selected work item to open a mapping website (like Bing Maps) to display a map of the address contained in that work item attribute.
- Provide menus/toolbars that display TIBCO Spotfire reports for the selected work item or process instance. Or, from a selected event in the event viewer, display a TIBCO Spotfire report showing the work done by the Principle Name of the selected event. The report can be embedded in Workspace, or displayed in a separate browser window.
- Provide a toolbar button on the process instance list that, when clicked, opens an external website into an iFrame embedded in Workspace that displays the regulatory documentation for that process.
- Provide a toolbar button at the top level of the application that opens a TIBCO PageBus Managed Hub iFrame in a dialog embedded in Workspace that contains a customer's order processing web interface. The customer's order processing web interface could be modified to publish custom PageBus events that are subscribed to in the Workspace application. Upon receiving these PageBus events, Workspace could use the payload of the event to do things like start a specified business services, or display a list of process instances filtered by some attribute in the event payload (like customer ID).
- Embed a custom TIBCO General Interface prototype into Workspace that provides an automated work delivery system that pushes the next available work item to the user (based on some filter) without directly exposing any work list to the user.

Workspace Application vs a Custom WCC Application

Custom interfaces can be configured for any WCC application. However, there are some aspects of custom interfaces that are only applicable to the "Workspace" application.

The "Workspace" application refers to the application that is automatically deployed to the runtime node when TIBCO ActiveMatrix BPM is installed on the runtime node, as well as the Workspace application that is installed in your development environment when TIBCO Business Studio is installed. The Workspace application that is installed in your development environment with TIBCO Business Studio can be customized in some way, then deployed to the runtime node, replacing the Workspace application that was automatically installed on the runtime node. In the context of custom interfaces, references to the Workspace application means either of these.

A "custom WCC application" is one that you create using WCC components.

For more information about the distinction between these applications, see Introduction.

Custom Interface Difference in each Application Type

Buttons and menus that launch a custom interface can be placed in the following locations only in a Workspace application, as these locations do not apply to custom WCC applications:

- Master Selection Area
- Main Application Toolbar

  Also, custom interfaces can be specified to open in the following locations only in a Workspace application, as these locations do not apply to custom WCC applications (for more information about these locations, see Custom Interface Location):

- Area A
- Area B
- Area C
- Area D

  Custom interfaces can be opened in a dialog or browser window in a custom WCC application, however. Also, the `loadCustomInterface` helper method allows custom WCC applications to load 'simpleIframe', 'simpleHubIframe', or 'simplePrototype' custom interfaces into a specific parent container.

Launching Custom Interfaces

Custom interfaces are launched via custom menus and toolbar buttons that can be placed in various locations throughout the Workspace application. They may be configured to launch automatically when a menu item or button is selected. Or they may be launched from JavaScript methods, which provides the ability to apply logic and parameterization when invoking a custom interface. "Helper" methods are also provided to facilitate loading and placement of interfaces launched from JavaScript.

When launched, custom interfaces may be displayed in various areas of the Workspace layout, as well as in dialogs or separate browser windows. The location at which a custom interface can be displayed is dependent on the type of custom interface, and the location of the menu or toolbar button from which it is launched.

The launch mechanism, location, text strings and source of custom interfaces are defined in an XML file (`customInterfaces.xml`) that can be managed using the Configuration Administrator of the Workspace application. Each custom interface has associated user access elements, which provide the ability to grant or prohibit access to the interface for specific users, groups, or positions in the organization.

Supported Custom Interfaces

This topic describes the types of custom interfaces that are supported.

They are:

- **Simple iFrame** - This type of custom interface is simply an iFrame embedded in Workspace, or a browser window, and displays a source URL specified in `customInterfaces.xml`. The URL is automatically loaded and the interface displays at a specified location when the launch component is executed.

- **Simple Managed Hub iFrame** - This custom interface is a TIBCO PageBus Managed Hub iFrame embedded in Workspace, or a browser window, and opens a web application at the URL specified in `customInterfaces.xml`. A managed Hub iFrame has the capability to communicate with WCC components and other web applications via secure PageBus events. The URL, of the web application, is automatically loaded and the interface displays at a specified location when the launch component is executed.
- **Simple Prototype** - This custom interface consists of a TIBCO General Interface prototype containing user interface components. Custom JavaScript may be added to the Workspace application to handle events generated by the prototype components. The prototype is automatically loaded into a specified location when the launch component is executed.
- **Scripted Interface** - This custom interface is determined by a custom JavaScript method that is automatically invoked when the launch component is executed. The type, functionality, and location of the custom interface is managed by custom JavaScript. Helper methods are provided to use with this custom interface type; see Custom Interface Helper Methods.

During Workspace application initialization, the contents of `customInterfaces.xml` are extracted and the launch component, for each custom interface, is placed at the specified location within the Workspace layout along with the associated text and image definition. The execute event, of the launch component, is set to either automatically load the source of the interface (for the "simple" interfaces), or to call a JavaScript method that performs the function of launching the custom interface (for the "scripted" interface).

If the Workspace Configuration Administrator is used to define custom interfaces in `customInterfaces.xml`, associated user access elements are automatically added to `userAccess.xml` to define which users in the organization have access to the launch component for each interface. For more information, see User Access Elements for Custom Interfaces.

Custom Interface Configuration

The custom interface definition file, `customInterfaces.xml`, contains elements that define the location, type, launch component, and attributes of each custom interface in the Workspace application.

The way in which you update the `customInterfaces.xml` file depends on whether you are configuring custom interfaces in the Workspace application that is already deployed to a runtime node, or to the Workspace application that was installed in your development environment with TIBCO Business Studio, as follows:

- **Deployed Workspace Application** - Configuring custom interfaces in a Workspace application that is deployed on a runtime node can be done in one of two ways:
  - **Using the Configuration Administrator** - This provides a graphical UI to modify the `customInterfaces.xml` file. For more information, see Configuration Administrator Interface.

    For a more general discussion of the Configuration Administrator, see Using the Configuration Administrator.
  - **Directly Modifying Configuration File** - You can also directly modify the `customInterfaces.xml` file on the runtime node to configure custom interfaces. Note, however, that this method of configuration can only be used for the Workspace application. It cannot be used for a deployed custom WCC application (the specific location in the file system in which the WCC application is deployed is indeterminable—it contains GUIDs determined at deployment time).

    For information about the elements in the `customInterfaces.xml` file used to configure custom interfaces, see Configuration Elements.

    For information about locating the `customInterfaces.xml` file on the runtime node, see Directly Modifying Files on the BPM Runtime Machine.

    If you are configuring custom interfaces for a Workspace application that is deployed on a runtime node, you need to understand that once you modify a configuration file with the

Configuration Administrator, its contents are written to the database; after that point, that configuration is read from the database, and not from the file in the file system. For more information about this, see Configuration Administrator Writes Files to the Database.

- **Workspace Application in the Development Environment** - You can also configure custom interfaces in either the Workspace application that is installed in your development environment with TIBCO Business Studio, or in a custom WCC application you are developing.

  The custom interface definition file, `customInterfaces.xml`, is located as follows in your development environment:

  ```
     StudioHome\wcc\version\JSXAPPS\workspace\WCCProjectName
  ```

  where:

  – *`StudioHome`* is the directory in which TIBCO Business Studio was installed.

  – *version* is the version number of Workspace that was installed with TIBCO Business Studio.

  – *WCCProjectName* is the name of the General Interface Builder project that contains your custom application. If you are working with the Workspace application, this is “workspace”.

  For information about the elements in the `customInterfaces.xml` file used to configure custom interfaces, see Configuration Elements.

  After customizing/configuring the application, it must be deployed to the runtime node. For information about deploying an application after customizing it in your development environment, see Deploying an Application After Customizing.

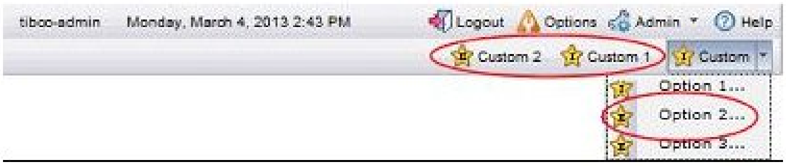
Menu and Toolbar Button Location

Custom interfaces can be launched from various locations in the Workspace application.

The following shows the locations at which you can place menus and toolbar buttons to launch custom interfaces, as well as the elements in the `customInterfaces.xml` file used to configure custom interfaces launched from those locations (for more detail about these elements, see Configuration Elements):

Main Application Toolbar

This menu and toolbar button location is applicable only for the Workspace application. It does not apply for custom WCC applications.

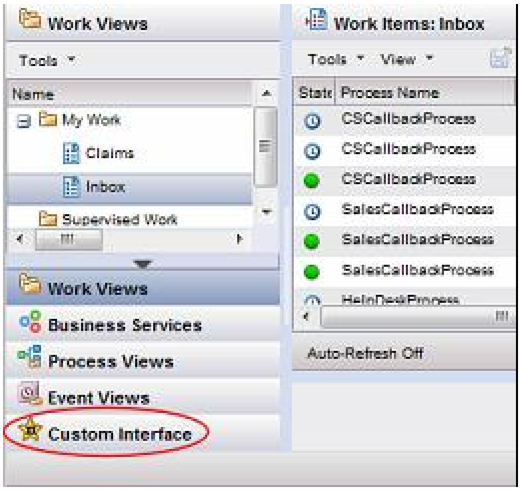


Use the following elements in `customInterfaces.xml` to configure custom interfaces launched from this location:

```
<WorkspaceToolbar>
   <menus>
   </menus>
   <toolbarButtons>
   </toolbarButtons>
</WorkspaceToolbar>
```

Master Selection Area

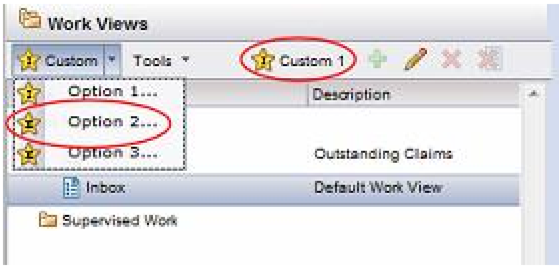
This toolbar button location is applicable only for the Workspace application. It does not apply for custom WCC applications.



Use the following elements in `customInterfaces.xml` to configure custom interfaces launched from this location:

```
<WorkspaceMasterSelection>
   <toolbarButtons>
   </toolbarButtons>
</WorkspaceMasterSelection>
```

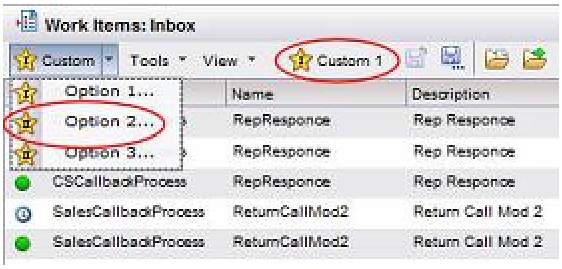
Work Views List



Use the following elements in `customInterfaces.xml` to configure custom interfaces launched from this location:

```
<WorkViewsList>
   <menus>
   </menus>
   <toolbarButtons>
   </toolbarButtons>
</WorkViewsList>
```

Work Items List

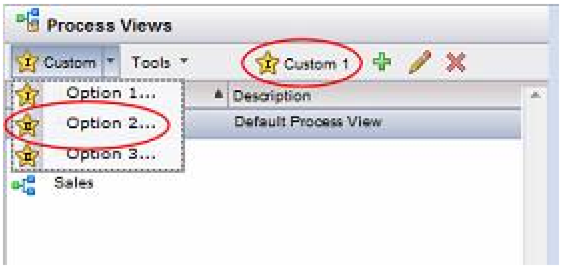


	Name	Description
	RepResponce	Rep Responce
	RepResponce	Rep Responce
CSCallbackProcess	RepResponce	Rep Responce
SalesCallbackProcess	ReturnCallMod2	Return Call Mod 2
SalesCallbackProcess	ReturnCallMod2	Return Call Mod 2

Use the following elements in `customInterfaces.xml` to configure custom interfaces launched from this location:

```
<WorkItemsList>
   <menus>
   </menus>
   <toolbarButtons>
   </toolbarButtons>
</WorkItemsList>
```

Process Views List

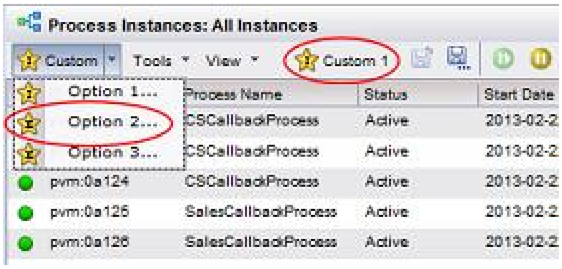


Description
Default Process View

Use the following elements in `customInterfaces.xml` to configure custom interfaces launched from this location:

```
<ProcessViewsList>
   <menus>
   </menus>
   <toolbarButtons>
   </toolbarButtons>
</ProcessViewsList>
```

Process Instances List

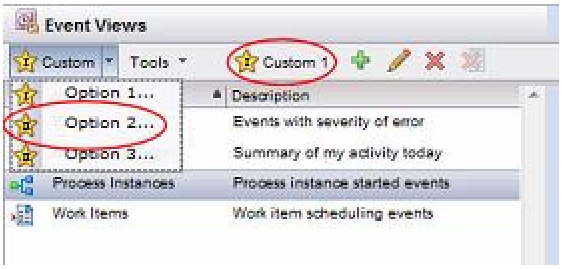


	Process Name	Status	Start Date
	CSCallbackProcess	Active	2013-02-2
	CSCallbackProcess	Active	2013-02-2
pvm:0a124	CSCallbackProcess	Active	2013-02-2
pvm:0a125	SalesCallbackProcess	Active	2013-02-2
pvm:0a126	SalesCallbackProcess	Active	2013-02-2

Use the following elements in `customInterfaces.xml` to configure custom interfaces launched from this location:

```
<ProcessInstancesList>
   <menus>
   </menus>
   <toolbarButtons>
   </toolbarButtons>
</ProcessInstancesList>
```

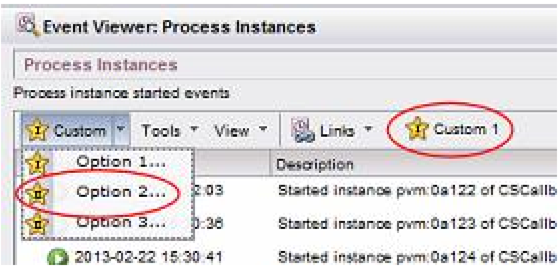
Event Views List


Event Views
Custom
Tools
Custom 1
Option 1...
Option 2...
Option 3...
Description
Events with severity of error
Summary of my activity today
Process Instances
Process instance started events
Work Items
Work item scheduling events

Use the following elements in `customInterfaces.xml` to configure custom interfaces launched from this location:

```
<EventViewsList>
   <menus>
   </menus>
   <toolbarButtons>
   </toolbarButtons>
</EventViewsList>
```

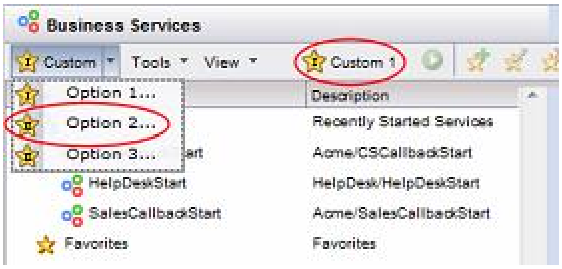
Events List


Event Viewer: Process Instances
Process Instances
Process instance started events
Custom
Tools
View
Links
Custom 1
Option 1...
Option 2...
Option 3...
Description
2:03
Started instance pvm:0a122 of CSCallb
0:36
Started instance pvm:0a123 of CSCallb
2013-02-22 15:30:41
Started instance pvm:0a124 of CSCallb

Use the following elements in `customInterfaces.xml` to configure custom interfaces launched from this location:

```
<EventsList>
   <menus>
   </menus>
   <toolbarButtons>
   </toolbarButtons>
</EventsList>
```

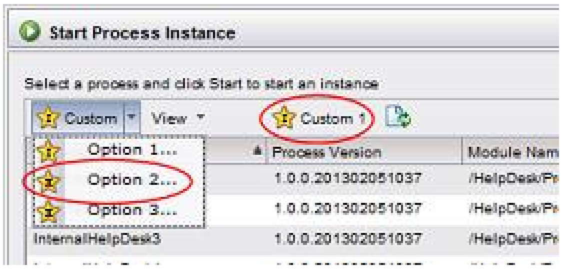
Business Services List



Use the following elements in `customInterfaces.xml` to configure custom interfaces launched from this location:

```
<BusinessServicesList>
   <menus>
   </menus>
   <toolbarButtons>
   </toolbarButtons>
</BusinessServicesList>
```

Process Templates List



Select a process and click Start to start an instance

	Process Version	Module Nam
	1.0.0.201302051037	/HelpDesk/Pr
	1.0.0.201302051037	/HelpDesk/Pr
InternalHelpDesk3	1.0.0.201302051037	/HelpDesk/Pr

Use the following elements in `customInterfaces.xml` to configure custom interfaces launched from this location:

```
<ProcessTemplatesList>
   <menus>
   </menus>
   <toolbarButtons>
   </toolbarButtons>
</ProcessTemplatesList>
```

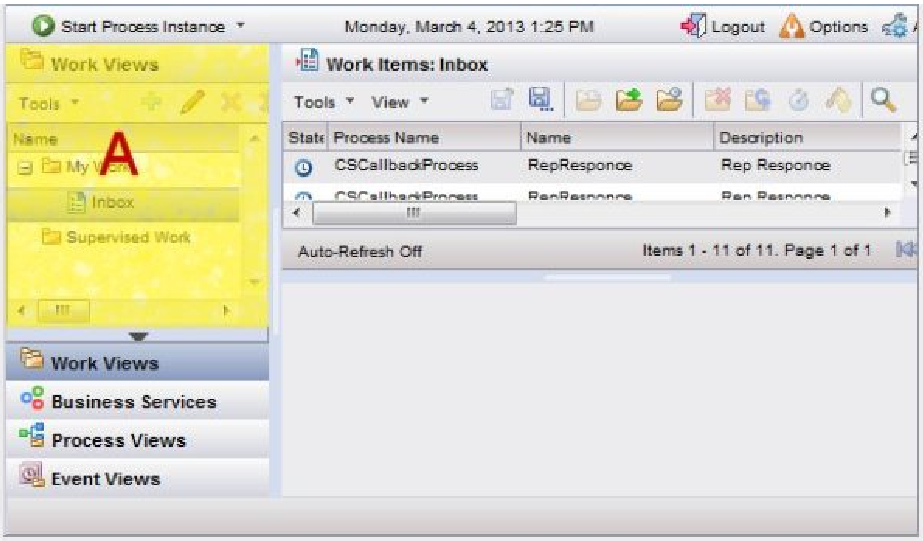
Custom Interface Location

When defining menu items or toolbar buttons that launch a custom interface, a location must be specified that determines where in the Workspace application the custom interface will appear.

These locations, which are identified as Area A, Area B, Area C, Area D, dialog, and browser, are illustrated below:

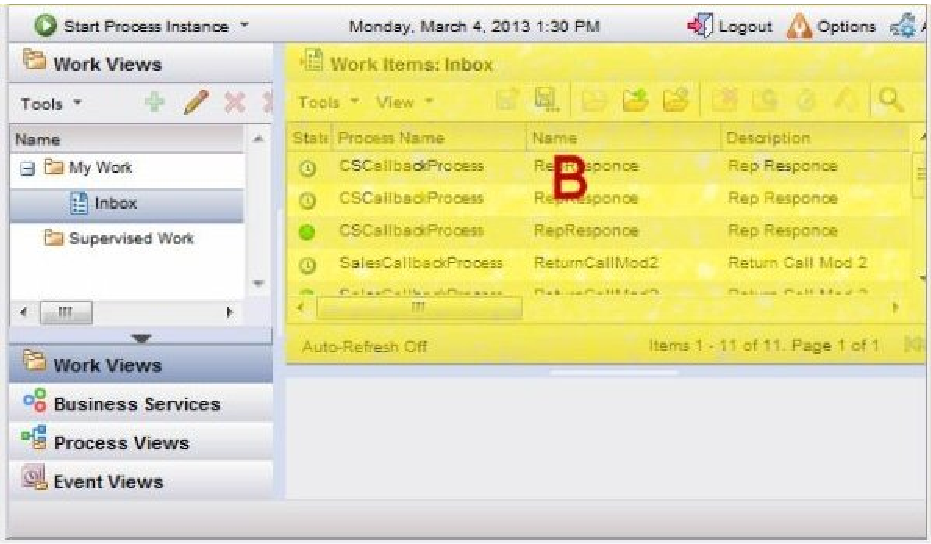
Custom interface locations Area A, Area B, Area C, and Area D are applicable only for the Workspace application. They do not apply for custom WCC applications.

Area A



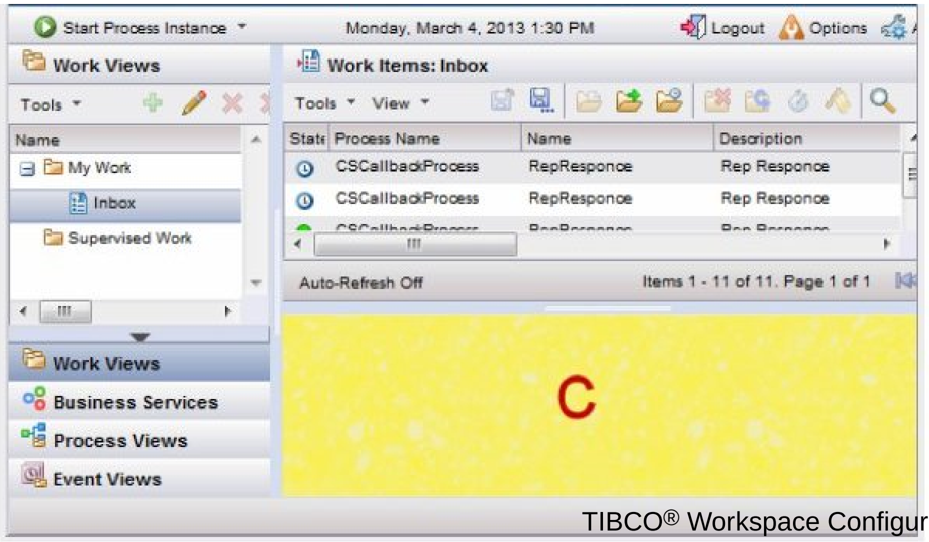
State	Process Name	Name	Description
	CSCallbackProcess	RepResponce	Rep Responce

Area B



State	Process Name	Name	Description
	CSCallbackProcess	RepResponce	Rep Responce
	CSCallbackProcess	RepResponce	Rep Responce
	CSCallbackProcess	RepResponce	Rep Responce
	SalesCallbackProcess	ReturnCallMod2	Return Call Mod 2

Area C



State	Process Name	Name	Description
	CSCallbackProcess	RepResponce	Rep Responce
	CSCallbackProcess	RepResponce	Rep Responce

Area D

button that launches the interface, only a subset of locations may be valid. Invalid locations are not displayed.

The following table details which locations are available for custom interfaces launched from each of the menu/toolbar button locations:

Launch Location	simpleIframe	simpleHubIframe	simplePrototype	scriptedInterface
WorkspaceToolbar	dialog, browser	dialog, browser	dialog, browser	dialog, browser
WorkspaceMasterSelection	Area D	Area D	Area A, B, C, D	Area A, B, C, D (Areas A, B, C only valid for simplePrototpe)
WorkViewsList	Area D, dialog, browser	Area D, dialog, browser	Area A, B, C, D, dialog, browser	Area A, B, C, D, dialog, browser (Areas A, B, C only valid for simplePrototpe)
WorkItemsList	Area D, dialog, browser	Area D, dialog, browser	Area A, B, C, D, dialog, browser	Area A, B, C, D, dialog, browser (Areas A, B, C only valid for simplePrototpe)
ProcessViewsList	Area D, dialog, browser	Area D, dialog, browser	Area A, B, C, D, dialog, browser	Area A, B, C, D, dialog, browser (Areas A, B, C only valid for simplePrototpe)
ProcessInstancesList	Area D, dialog, browser	Area D, dialog, browser	Area A, B, C, D, dialog, browser	Area A, B, C, D, dialog, browser (Areas A, B, C only valid for simplePrototpe)
EventViewsList	Area D, dialog, browser	Area D, dialog, browser	Area A, B, C, D, dialog, browser	Area A, B, C, D, dialog, browser (Areas A, B, C only valid for simplePrototpe)

Launch Location	simpleIframe	simpleHubIframe	simplePrototype	scriptedInterface
EventsList	Area D, dialog, browser	Area D, dialog, browser	Area A, B, C, D, dialog, browser	Area A, B, C, D, dialog, browser (Areas A, B, C only valid for simplePrototpe)
BusinessServicesList	Area D, dialog, browser	Area D, dialog, browser	Area A, D, dialog, browser	Area A, D, dialog, browser (Areas A only valid for simplePrototpe)
ProcessTemplatesList	dialog, browser	dialog, browser	dialog, browser	dialog, browser

Configuration Elements

The `customInterfaces.xml` file contains a series of elements that provide the framework to configure custom interfaces.

For example:

```
<data>
   <customInterfaces>
      <WorkspaceToolbar>
         <menus>
         </menus>
         <toolbarButtons>
         </toolbarButtons>
      </WorkspaceToolbar>
      <WorkspaceMasterSelection>
         <toolbarButtons>
         </toolbarButtons>
      </WorkspaceMasterSelection>
               .
               .
               .
      <ProcessTemplatesList>
         <menus>
         </menus>
         <toolbarButtons>
         </toolbarButtons>
      </ProcessTemplatesList>
   </customInterfaces>
</data>
```

This document provides details about configuring custom interfaces via the XML in the `customInterfaces.xml` file. If desired, you can configure custom interfaces via the Configuration Administrator UI rather than the XML. For information, see Configuration Administrator Interface .

The `customInterfaces.xml` file contains a root `<data>` element and a `<customInterfaces>` sub-element.

Below the `<customInterfaces>` element are a number of sub-elements that specify the location from which you want the custom interface to be launched; the example above shows three of the location elements: `<WorkspaceToolbar>` (the main application toolbar), `<WorkspaceMasterSelection>` (the master selection area of Workspace), and `<ProcessTemplatesList>` (the process template list). For a complete list, see Configuring <menu> and <menuItem> Elements.

Menus and Toolbar Elements

Subordinate to each of the elements that specify where the custom interface is launched, are <menus> and <toolbarButtons> elements (with the exception of the <WorkspaceMasterSelection> location element, which has only a subordinate <toolbarButtons> element).

The <menus> and <toolbarButtons> elements are empty by default. For example:

```
<WorkspaceToolbar>
   <menus>
   </menus>
   <toolbarButtons>
   </toolbarButtons>
</WorkspaceToolbar>
```

To configure custom interfaces to launch from a menu, you must add <menu> and <menuItem> elements below <menus>, where <menu> represents a drop-down menu, and <menuItem> configures an individual item on the menu. For example:

```
<menus>
    <menu>
      <menuItem>
      </menuItem>
      <menuItem
      </menuItem>
    </menu>
  </menus>
```

To configure custom interfaces to launch from a toolbar button, you must add <toolbarButton> elements below <toolbarButtons>. For example:

```
<toolbarButtons>
    <toolbarButton
    </toolbarButton>
    <toolbarButton
    </toolbarButton>
  </toolbarButtons>
</WorkItemsList>
```

Then inside of the <menu> and <toolbarButton> elements, you must add the appropriate attributes that define the desired custom interface. These attributes are described in Configuring <menu> and <menuItem> Elements and Configuring <toolbarButton> Elements.

Configuring <menu> and <menuItem> Elements

The <menu> and <menuItem> elements must contain the appropriate attributes to configure the desired custom interface that is to be launched from a menu selection.

The following is an example of a custom interface that can be launched from a menu on the work item list:

```
<WorkItemsList>
  <menus>
    <menu name="WIMenu" displayText="Accounts"
      description="accounts payable and receivable access"
      toolTip="open accounts payable or receivable"
      image="JSXAPPS/base/application/images/custom2.gif"
      enable="always">
      <menuItem name="WIitem1" displayText="Payable"
         description="accounts payable access"
         image="JSXAPPS/base/application/images/custom1.gif"
         enable="always">
            <simpleHubIframe location="dialog"
            features="resizable,minimize,maximize,close,center"
            source="http://ozquadling:8090/iFrameInterface/Payable.html">
            </simpleHubIframe>
      </menuItem>
      <menuItem name="WIitem2" displayText="Receivable"
        description="accounts receivable access"
        image="JSXAPPS/base/application/images/custom1.gif"
        enable="always">
        <scriptedInterface location="browser"
        features="resizable,scrollbars,status,center,width=500,height=400"
        source="loadInterface_WIitem2">
        </scriptedInterface>
      </menuItem>
    </menu>
  </menus>
</WorkItemsList>
```

The `<menu>` element contains the following attributes:

- **name** - (required) A unique identifier for the custom interface menu. A user access element is created with this name (for more information, see User Access Elements for Custom Interfaces).
- **displayText** - (optional) If you are not localizing your application, this can be the text you want to display on the menu title. If you are localizing, this must be the 'id' record, in the `customInterfaceLocale.xml` file, that specifies the text to display on the menu title. You may also choose to not provide display text, for instance, if you want only an image to appear.
- **description** - (optional) This is the description that displays in the Configuration Administrator when the user access control for the menu is selected. If you are not localizing your application, this can be the text you want to display for the user access control description. If you are localizing, this must be the 'id' record, in the `customInterfaceLocale.xml` file, that specifies the user access control description.
- **toolTip** - (optional) This is the text that displays when the cursor hovers over the menu. If you are not localizing your application, this can be the tooltip text you want to display. If you are localizing, this must be the 'id' record, in the `customInterfaceLocale.xml` file, that specifies the tooltip text.
- **image** - (optional) The path to the image associated with the menu.
- **enable** - (optional) Specifies when the menu should be enabled, as follows
  - **always** - (default) The menu is always enabled. Note that this is the only possible value for menus in the main application toolbar (<WorkspaceToolbar> element).
  - **any** - The menu is enabled when any items in the associated list are selected; it is disabled if no items are selected.
  - **single** - The menu is enabled when a single item in the associated list is selected; it is disabled if no items or multiple items are selected.
  - **never** - The menu is initially disabled; enabling/disabling of the menu is managed in custom JavaScript.

The <menu> element must contain one or more <menuItem> sub-elements that represent items in the drop-down area of the menu. Custom interfaces are launched from the <menuItem> elements, which have the following attributes:

- **name** - (required) A unique identifier for the custom interface menu item. A user access element is created with this name (for more information, see User Access Elements for Custom Interfaces).
- **displayText** - (optional) If you are not localizing your application, this can be the text you want to display on the menu item. If you are localizing, this must be the 'id' record, in the `customInterfaceLocale.xml` file, that specifies the text to display on the menu item. You may also choose to not provide display text, for instance, if you want only an image to appear.
- **description** - (optional) This is the description that displays in the Configuration Administrator when the user access control for the menu item is selected. If you are not localizing your application, this can be the text you want to display for the user access control description. If you are localizing, this must be the 'id' record, in the `customInterfaceLocale.xml` file, that specifies the user access control description.
- **image** - (optional) The path to the image associated with the menu item.

- **enable** - (optional) Specifies when the menu item should be enabled, as follows:
  - **always** - (default) The menu item is always enabled.
  - **any** - The menu item is enabled when any items in the associated list are selected; it is disabled if no items are selected.
  - **single** - The menu item is enabled when a single item in the associated list is selected; it is disabled if no items or multiple items are selected.
  - **never** - The menu item is initially disabled; enabling/disabling of the menu item is managed in custom JavaScript.

The <menuItem> element must also have a sub-element that specifies the type of custom interface that is launched from the menu item. For information about the available custom interface type elements, including their attributes, see Custom Interface Type Specification.

Configuring <toolbarButton> Elements

The <toolbarButton> element must contain the appropriate attributes to configure the desired custom interface that is to be launched from a toolbar button.

The following is an example of a custom interface that can be launched from a toolbar button on the work item list:

```
<WorkItemsList>
   <toolbarButtons>
     <toolbarButton name="WIbutton1" displayText="Orders"
        description="orders access" toolTip="outstanding orders"
        image="JSXAPPS/base/application/images/custom1.gif"
        enable="any">
        <simplePrototype location="areaB"
          source="JSXAPPS/CustomInterface/prototypes/interface2.xml">
        </simplePrototype>
    </toolbarButton>
    <toolbarButton name="WIbutton2" displayText="Deliverables"
       description="deliverables access"
       toolTip="outstanding deliverables"
       image="JSXAPPS/base/application/images/custom1.gif"
       enable="single">
       <simpleIframe location="areaC"
         source="http://www.acme.com/deliverables/">
       </simpleIframe>
    </toolbarButton>
  </toolbarButtons>
</WorkItemsList>
```

The <toolbarButton> element contains the following attributes:

- **name** - (required) A unique identifier for the custom interface toolbar button. A user access element is created with this name (for more information, see User Access Elements for Custom Interfaces).
- **displayText** - (optional) If you are not localizing your application, this can be the text you want to display on the toolbar button. If you are localizing, this must be the 'id' record, in the `customInterfaceLocale.xml` file, that specifies the text to display on the toolbar button. You may also choose to not provide display text, for instance, if you want only an image to appear.
- **description** - (optional) This is the description that displays in the Configuration Administrator when the user access control for the toolbar button is selected. If you are not localizing your application, this can be the text you want to display for the user access control description. If you are localizing, this must be the 'id' record, in the `customInterfaceLocale.xml` file, that specifies the user access control description.
- **toolTip** - (optional) This is the text that displays when the cursor hovers over the toolbar button. If you are not localizing your application, this can be the tooltip text you want to display. If you are localizing, this must be the 'id' record, in the `customInterfaceLocale.xml` file, that specifies the tooltip text.

- **image** - (optional) The path to the image associated with the toolbar button.
- **enable** - (optional) Specifies when the toolbar button should be enabled, as follows:
  - **always** - (default) The toolbar button is always enabled. Note that this is the only possible value for toolbar buttons in the main application toolbar (<WorkspaceToolbar> element), as well as in the master selection area (<WorkspaceMasterSelection> element).
  - **any** - The toolbar button is enabled when any items in the associated list are selected; it is disabled if no items are selected.
  - **single** - The toolbar button is enabled when a single item in the associated list is selected; it is disabled if no items or multiple items are selected.
  - **never** - The toolbar button is initially disabled; enabling/disabling of the button is managed in custom JavaScript.

The <toolbarButton> element must also have a sub-element that specifies the type of custom interface that is launched from the toolbar button. For information about the available custom interface type elements, including their attributes, see Custom Interface Type Specification.

Custom Interface Type Specification

Both the <menuItem> and <toolbarButton> elements must have a sub-element that specifies the type of custom interface. This sub-element has attributes that specify the source and location of the custom interface.

The following are valid sub-elements of the <menuItem> and <toolbarButton> elements:

Custom Interface Type	Description
`<simpleIframe>`	A simple iFrame interface.
`<simpleHubIframe>`	A simple Managed Hub iFrame interface. This must contain <subscribe> or <publish> sub-elements, which contains <topic> sub-elements specifying the event topics to which the interface subscribes and publishes. For more information, see Simple Managed Hub iFrames Subscriptions and Publications .
`<simplePrototype>`	A TIBCO General Interface prototype.
`<scriptedInterface>`	The custom interface is determined in JavaScript. All custom interfaces of this type must have a handler method in the ...`JSXAPPS\workspace\application\js\AppMain.js` file. The name of the method must match the value of the **source** attribute in the scripted interface definition. The method is automatically invoked when the menu item or toolbar button of the custom interface is selected. Helper methods are provided to use with this interface; see Custom Interface Helper Methods .

For more information about these custom interface types, see Supported Custom Interfaces.

Each of the custom interface type elements includes the following attributes:

- **source** - (required) The source of the custom interface.
  - For `<simpleIframe>`, this is a URL to a web page.
  - For `<simpleHubIframe>`, this is a URL to a web application.

  - For `<simplePrototype>`, this is a path to a TIBCO General Interface prototype.
  - For `<scriptedInterface>`, this is the name of a method in `JSXAPPS\workspace\application\js\AppMain.js` that contains custom JavaScript for creating the custom interface.

- **location** - (required) The location in the Workspace application where the custom interface will display. Depending on the type of custom interface, and the location of the menu that launches the interface in the Workspace application, only a subset of the following locations may be valid. For specifics regarding valid locations for each type of custom interface, see Valid Custom Interface Locations. Possible values include:

  - **areaA** - The area where the work views, process views, event views, and business services lists display. See Area A.
  - **areaB** - The area where the work items, process instances, and events lists display. See Area B on page 186.
  - **areaC** - The area where the preview pane for work items, and process instances display, and where the event attributes display in the event viewer. See Area C.
  - **areaD** - The combination of areaB and areaC. Note that when areaD is specified, the vertical splitter that separates the left and right (in side-by-side layout), or upper and lower (in stacked layout) areas of the application layout is disabled for the simpleIframe and simpleHubIframe custom interface types. See Area D.
  - **dialog** - The custom interface displays in a floating dialog. See Dialog.
  - **browser** - The custom interface displays in separate browser window. See Browser.
  - **script** - The location is determined in JavaScript (scriptedInterface type only).

- **features** - If the location is "dialog" or "browser", an additional optional **features** attribute may be included to specify positioning and display characteristics of the dialog or browser window. The value of the **features** attribute is a comma-separated string that may contain any of the following:

  For location = "dialog":

  - **resizable** - Permits resizing of the dialog.
  - **modal** - The dialog is modal and no operation may be performed until the dialog is closed.
  - **minimize** - A button is displayed for minimizing the dialog .
  - **maximize** - A button is displayed for maximizing the dialog.
  - **close** - A button is displayed for closing the dialog.
  - **fullscreen** - The dialog is displayed maximized to fill the screen.
  - **center** - The dialog is displayed in the center of the screen (not applicable if "fullscreen" is specified).
  - **left=***nnn* - The dialog is displayed *nnn* number of pixels from the left of the screen (not applicable if "center" or "fullscreen" is specified).
  - **top=***nnn* - The dialog is displayed *nnn* number of pixels from the top of the screen (not applicable if "center" or "fullscreen" is specified).
  - **width=***nnn* - The dialog is displayed *nnn* number of pixels wide (not applicable if "fullscreen" is specified).
  - **height=***nnn* - The dialog is displayed *nnn* number of pixels high (not applicable if "fullscreen" is specified).

    Example for location = "dialog":

    ```
    features="resizable,minimize,maximize,close,center"
    ```

    For location = "browser":

- – **channelmode** - The browser window is displayed in "theater mode", that is, a maximized window (Internet Explorer only).
- – **dialog** - The browser window is displayed as a dialog.
- – **directories** - The "Bookmarks Toolbar" is displayed (Firefox only).
- – **location** - The "Navigation Toolbar" is displayed in Internet Explorer, the "Location Bar" is displayed in Firefox, or the "Toolbar" and "BookMarks Bar" are displayed in Safari.
- – **menubar** - The "Menu Bar" is displayed.
- – **resizable** - The user may resize the browser window (Internet Explorer only).
- – **scrollbars** - Scrollbars are displayed if the contents overflow the size of the browser window (Internet Explorer and Firefox only).
- – **status** - A status bar is included in the browser window (Internet Explorer only).
- – **toolbar** - The "Toolbar" is displayed.
- – **fullscreen** - The browser window is displayed maximized to fill the screen.
- – **center** - The browser window is displayed in the center of the screen (not applicable if "fullscreen" is specified).
- – **left=***nnn* - The browser window is displayed *nnn* number of pixels from the left of the screen (not applicable if "center" or "fullscreen" is specified).
- – **top=***nnn* - The browser window is displayed *nnn* number of pixels from the top of the screen (not applicable if "center" or "fullscreen" is specified).
- – **width=***nnn* - The browser window is displayed *nnn* number of pixels wide (not applicable if "fullscreen" is specified).
- – **height=***nnn* - The browser window is displayed *nnn* number of pixels high (not applicable if "fullscreen" is specified).

Example for location = "browser":

```
features="resizable,scrollbars,status,center,width=500,height=400"
```

Simple Managed Hub iFrames Subscriptions and Publications

If you are configuring a Simple Managed Hub iFrame custom interface, you can specify that the custom interface subscribe to a WCC component event, subscribe to a custom event, or publish an event, as follows.

For more information about PageBus Managed Hubs, see the "Components Events" chapter in the **`TIBCO Workspace Components Developer Guide`**, as well as the *TIBCO PgeBus™ Developer's Guide*.

- **Subscribe to a WCC component event** - This is done using a <subscribe> element, which contains one or more <topic> sub-elements, each specifying a WCC component event to which the interface subscribes. For example:

```
<simpleHubIframe location="dialog"

features="resizable,minimize,maximize,close,center,width=700,height=700"
           source="http://server:port/HubIframeSample/HubIframeSample.html">
  <subscribe>
      <topic
name="com.tibco.wcc.workspace.wccPrototype.wccWorkItems.listItemSelect"
            event="listItemSelect"></topic>
      <topic
name="com.tibco.wcc.workspace.wccPrototype.wccWorkItems.listExecute"
            event="listItemSelect"></topic>
  </subscribe>
</simpleHubIframe>
```

In this example, the custom interface is subscribing to the **listItemSelect** and **listExecute** events on the work item list component.

The **name** attribute in the <topic> element must be in the form:

```
com.tibco.wcc.workspace.wccPrototype.wccComponent.eventName
```

where:

- *wccComponent* identifies the WCC component for the event. The valid component names are:

  - wccWorkViews - Work Views

  - wccWorkItems - Work Items

  - wccProcessViews - Process Views

  - wccProcessInstances - Process Instances

  - wccBusinessServices - Business Services

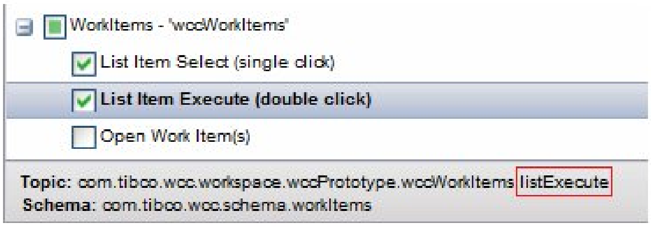
  - wccEventViews - Event Views

  - wccEventViewer - Event Viewer

  - wccProcessTemplates - Process Templates

  - wccProcessTemplatesEx - Process Templates Ex

- *eventName* identifies the event to which the custom interface is subscribing. For information about the names of the events published by each WCC component, see the "Non-WCC Subscribing to WCC Events" section in the *TIBCO Workspace Components Developer Guide*. Note that you can also view event names using the Configuration Administrator by selecting a component event in the UI and viewing the comment. For example:



⊟ ■ WorkItems - 'wccWorkItems'

☑ List Item Select (single click)

☑ List Item Execute (double click)

□ Open Work Item(s)

Topic: com.tibco.wcc.workspace.wccPrototype.wccWorkItems listExecute

Schema: com.tibco.wcc.schema.workItems

The **event** attribute in the <topic> element is the name of the event to which the interface is subscribing; this is the same as the *eventName* token in the **name** attribute. This attribute is used internally.

- **Subscribe to a custom event** - The custom interface can subscribe to a custom event (that is, a PageBus event other than a WCC component event) by including a <subscribe> element that contains one or more <topic> sub-elements, each specifying a PageBus event to which the interface subscribes. For example:

```
<simpleHubIframe location="dialog"

features="resizable,minimize,maximize,close,center,width=700,height=700"
          source="http://server:port/HubIframeSample/HubIframeSample.html">
  <subscribe>
      <topic name="com.customer.main.event.post" </topic>
      <topic name="com.customer.backup.event.post"</topic>
  </subscribe>
</simpleHubIframe>
```

- **Publish a custom event** - The custom interface can publish a custom event to PageBus by including a <publish> element, which contains one or more <topic> sub-elements, each specifying a PageBus event to publish. For example:

```
<simpleHubIframe location="dialog"

features="resizable,minimize,maximize,close,center,width=700,height=700"
           source="http://server:port/HubIframeSample/HubIframeSample.html">
```

```
  <publish>
      <topic name="com.customer.account.val.created" </topic>
  </publish>
</simpleHubIframe>
```

The WCC client application automatically subscribes to all custom events published by the custom interface.

An onHubSubscribeComplete callback method is provided in `JSXAPPS\`*`wccProjectName`*`\`*`application`*`\`*`js`*`\`*`AppMain.js`*, that is invoked when the client application subscribes to each custom event. This method can be used to verify the success of the Managed Hub subscription:

```
/**
* onHubSubscribeComplete - callback method that is invoked when the client
application
subscribes to a Simple Managed Hub iFrame custom event.
*
* @param {*} item - Id of the event for which the subscribe operation was invoked
* @param {Boolean} success -  If subscribe operation succeeded then true, else false
* @param {OpenAjax.hub.Error} errCode - If success != true, then contains a string
error
*                                       code, else is undefined
*/
        classProto.onHubSubscribeComplete = function(item, success, errCode) {
            if (!success) {
                this.logError("Subscribe failed (" + item + "): " + errCode);
            }
        };
```

When a custom event, to which the client has successfully subscribed, is published by the custom interface, the onHubEventData callback method is invoked in `JSXAPPS\`*`wccProjectName`*`\`*`application`*`\`*`js`*`\`*`AppMain.js`*. This method is used to handle a Managed Hub event that has been published by the custom interface:

```
/**
* onHubEventData - callback method that is invoked when a Simple Managed Hub iFrame
custom
*                  interface publishes a custom event.
*
* @param {String} topic - The topic on which the event was published
* @param {*} data - The event "payload" data. Can be any JSON-serializable value.
*/
        classProto.onHubEventData = function(topic, data) {
            if (topic == 'HubIframeSample.publish') {
                alert('event received: ' + topic + ' - ' + data.body.sample);
            }
        };
```

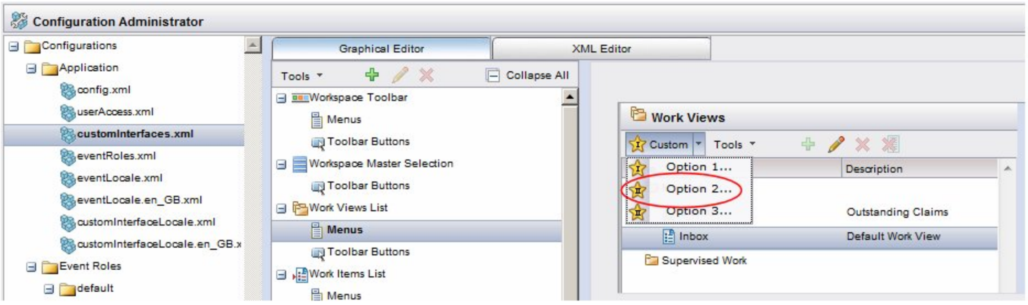
Both of the methods shown above (onHubSubscribeComplete and onHubEventData) are also included in the out-of-the-box Workspace application that is deployed to the runtime node when ActiveMatrix BPM is installed.

Configuration Administrator Interface

If you are configuring custom interfaces in a Workspace application that is deployed on a runtime node, you can use the Configuration Administrator.

For general information about using the Configuration Administrator, see Using the Configuration Administrator.

When you select the `customInterfaces.xml` file from within the Configuration Administrator, then choose either "Menus" or "Toolbar Buttons" for one of the custom interfaces locations, an illustration displays that shows where the menu or button is added if you configure a custom interface for that location:



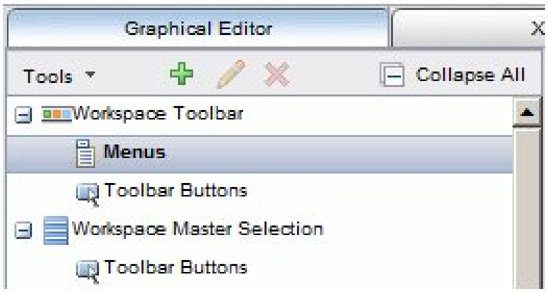
Note that you can also use the **Expand All** / **Collapse All** button to expand or collapse the entire custom interface tree.

Configuring a Custom Interface Launched from a Menu

Procedure

1. On the Configuration Administrator **Graphical Editor** tab, choose "Menus" from the location where you want the menu to appear.

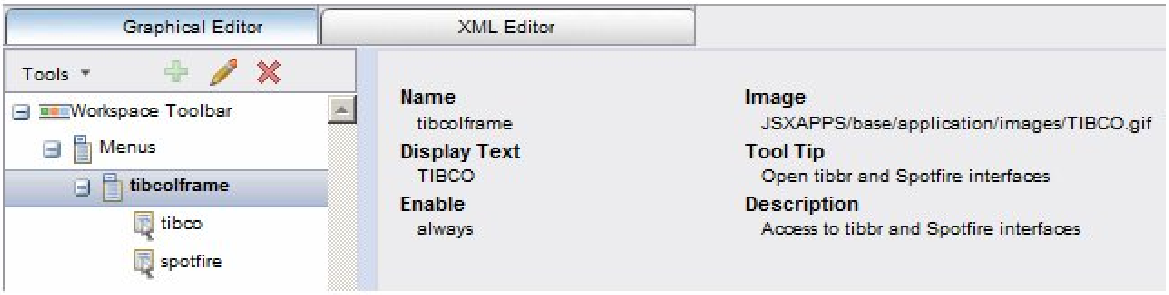
   For example, to choose that the custom interface be launched from a menu on the main Workspace toolbar:



2. Click the **Create a new Custom Interface** button (✚).
3. On the **Add Custom Interface** dialog, enter the configuration details into the fields according to the descriptions in Configuring <menu> and <menuItem> Elements (the fields on the dialog correspond with the attributes under the `<menu>` and `<menuItem>` elements).

   In the **Menu Items** section, click **Add** for each menu item you want to configure on the menu.
4. Click **OK** when you’ve completed the **Add Custom Interface** dialog.

   Once you have completed a menu configuration, it appears in the **Graphical Editor** with the configured menu items. The configuration appears in the right pane. For example:



Name
tibcolframe

Display Text
TIBCO

Enable
always

Image
JSXAPPS/base/application/images/TIBCO.gif

Tool Tip
Open tibbr and Spotfire interfaces

Description
Access to tibbr and Spotfire interfaces

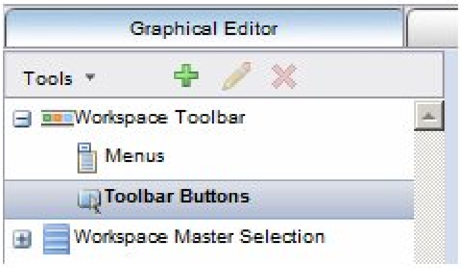
5. Click **OK** on the **Configuration Administrator** dialog to complete the configuration. You must log out and log back in for the menu to appear in the application.

Configuring a Custom Interface Launched from a Toolbar Button

Procedure

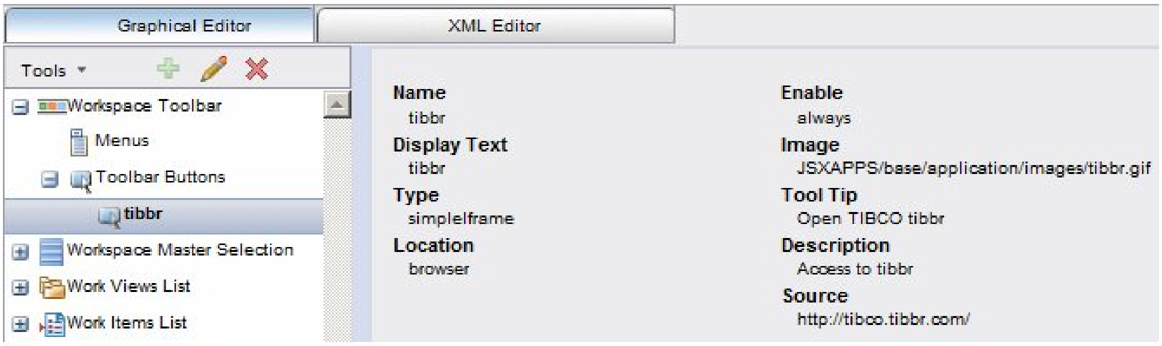
1. On the Configuration Administrator **Graphical Editor** tab, choose "Toolbar Buttons" from the location where you want the button to appear.

   For example, to choose that the custom interface be launched from a button on the main Workspace toolbar:



2. Click the **Create a new Custom Interface** button (+).
3. On the **Add Custom Interface** dialog, enter the configuration details into the fields according to the descriptions in Configuring <toolbarButton> Elements (the fields on the dialog correspond with the attributes under the `<toolbarButton>` element).
4. Click **OK** when you've completed the **Add Custom Interface** dialog.

   Once you have completed a toolbar button configuration, it appears in the **Graphical Editor**. The configuration appears in the right pane. For example:



Name
tibbr
Display Text
tibbr
Type
simpleIframe
Location
browser
Enable
always
Image
JSXAPPS/base/application/images/tibbr.gif
Tool Tip
Open TIBCO tibbr
Description
Access to tibbr
Source
http://tibco.tibbr.com/

5. Click **OK** on the **Configuration Administrator** dialog to complete the configuration. You must log out and log back in for the toolbar button to appear in the application.

Subscribing to or Publishing Events for a Managed Hub iFrame Interface

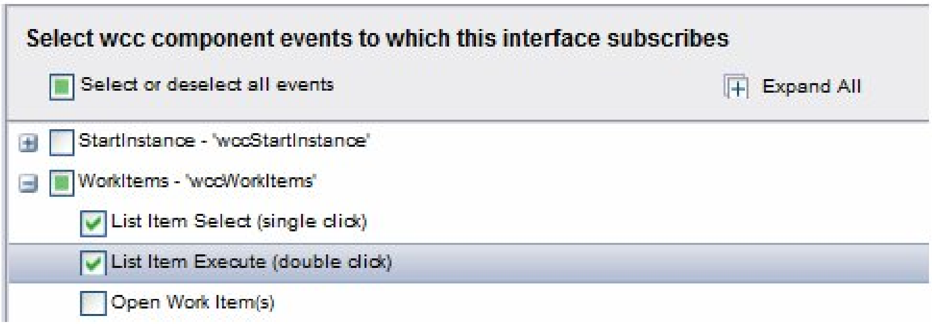
When using the Configuration Administrator to configure a custom interface of type Managed Hub iFrame, event subscriptions and publications are specified as described in this topic.

After you have configured the menu or toolbar button to launch the custom interface, follow the procedure below.

Procedure

1. Select the configured menu or toolbar button in the Configuration Administrator Graphical Editor.

   The **Select wcc component event to which this interface subscribes** pane lists all WCC components and the events published by those components.
2. Select all of the events to which you want the custom interface to subscribe:



Select wcc component events to which this interface subscribes

☑ Select or deselect all events   ⊞ Expand All

⊞ □ StartInstance - 'wccStartInstance'

⊟ ☑ WorkItems - 'wccWorkItems'

☑ List Item Select (single click)

☑ List Item Execute (double click)

□ Open Work Item(s)

3. If you want the custom interface to subscribe to a PageBus event other than those published by WCC components, enter the event topic in the **Specify custom event to which this interface subscribes** pane, by clicking the **Add** button on that pane and entering the topic.
4. If you want the custom interface to publish an event to the PageBus, enter the event topic in the **Specify custom events to this interface publishes** pane, by clicking the **Add** button on that pane and entering the topic.
5. Click **OK** to complete the custom interface configuration.

User Access Elements for Custom Interfaces

Access to toolbar buttons, menus, and menu items that launch custom interfaces is controlled via entries in the `userAccess.xml`, like all other menu items and buttons in the Workspace application.

An empty "CustomInterfaces" entry is included in the `userAccess.xml` file for the "Default User" (that is, under the AccessDefaults profile):

```
<access name="CustomInterfaces"/>
```

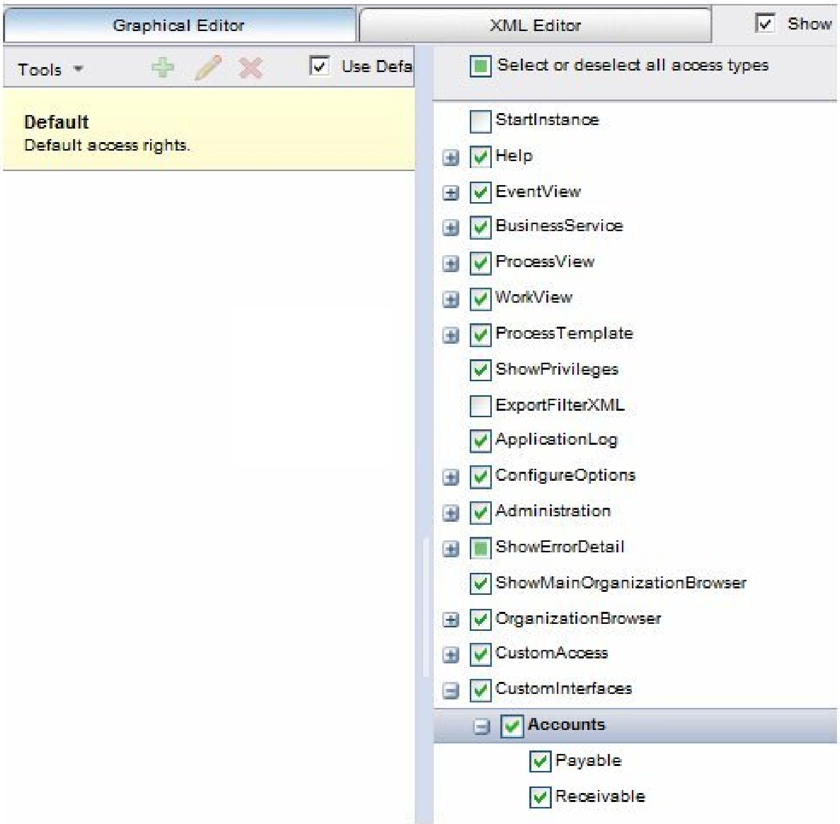
If you are configuring custom interfaces manually by editing the `customInterfaces.xml` file, you must also manually add entries to the "CustomInterfaces" access element in the `userAccess.xml` file. For example:

```
<access name="CustomInterfaces">
  <access name="accounts">
    <access name="payable"></access>
    <access name="receivable"></access>
</access>
```

And you must manually add CustomInterface access entries to other profiles in the `userAccess.xml` file.

If you are configuring custom interfaces using the Configuration Administrator, a corresponding access element is automatically added to the "CustomInterfaces" access element for every access profile.

By default, the "Default" user access profile is automatically granted access to the custom interface. For example:

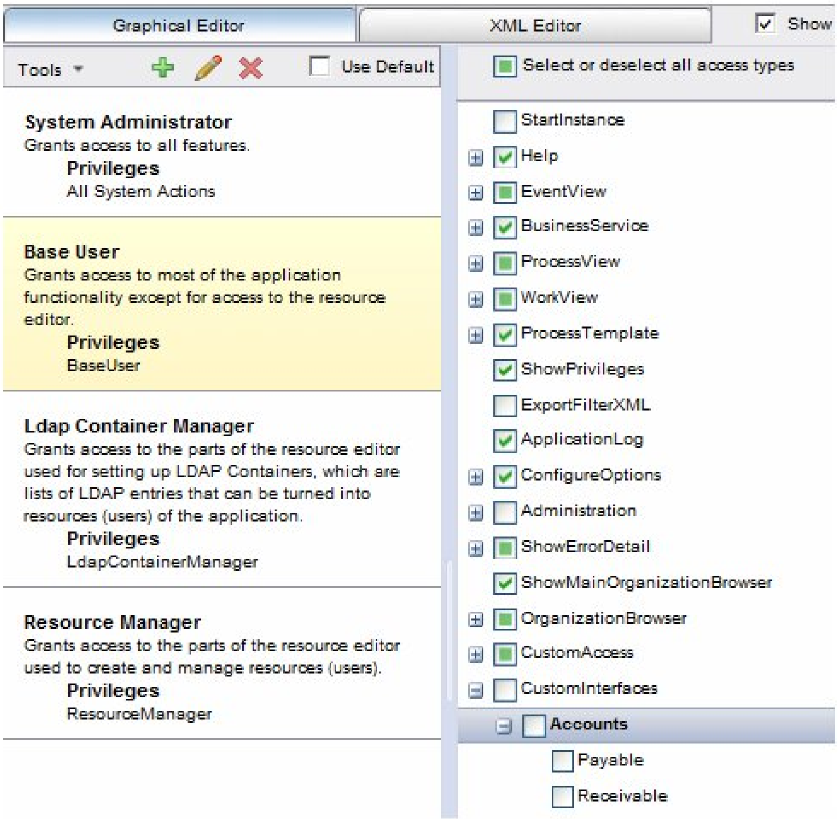


Default
Default access rights.

■ Select or deselect all access types

- □ StartInstance
- ⊞ ☑ Help
- ⊞ ☑ EventView
- ⊞ ☑ BusinessService
- ⊞ ☑ ProcessView
- ⊞ ☑ WorkView
- ⊞ ☑ ProcessTemplate
- ☑ ShowPrivileges
- □ ExportFilterXML
- ☑ ApplicationLog
- ⊞ ☑ ConfigureOptions
- ⊞ ☑ Administration
- ⊞ ■ ShowErrorDetail
- ☑ ShowMainOrganizationBrowser
- ⊞ ☑ OrganizationBrowser
- ⊞ ☑ CustomAccess
- ⊟ ☑ CustomInterfaces
  - ⊟ ☑ **Accounts**
    - ☑ Payable
    - ☑ Receivable

The same access entries are automatically added for all other access profiles in the Configuration Administrator, but they are *not* granted access by default. For example, if you select the "Base User" access profile, entries for the new custom interface are added, but they are not selected:



System Administrator
Grants access to all features.
Privileges
All System Actions

Base User
Grants access to most of the application functionality except for access to the resource editor.
Privileges
BaseUser

Ldap Container Manager
Grants access to the parts of the resource editor used for setting up LDAP Containers, which are lists of LDAP entries that can be turned into resources (users) of the application.
Privileges
LdapContainerManager

Resource Manager
Grants access to the parts of the resource editor used to create and manage resources (users).
Privileges
ResourceManager

🟩 Select or deselect all access types

- ☐ StartInstance
- ☑ Help
- 🟩 EventView
- ☑ BusinessService
- 🟩 ProcessView
- 🟩 WorkView
- ☑ ProcessTemplate
- ☑ ShowPrivileges
- ☐ ExportFilterXML
- ☑ ApplicationLog
- ☑ ConfigureOptions
- ☐ Administration
- 🟩 ShowErrorDetail
- ☑ ShowMainOrganizationBrowser
- 🟩 OrganizationBrowser
- 🟩 CustomAccess
- ☐ CustomInterfaces
  - ☐ **Accounts**
    - ☐ Payable
    - ☐ Receivable

You can control access to the custom interface menus, menu items, and toolbar buttons in the same way you control access to any other menus, menu items, and buttons in the Workspace application.

Note, however, if you add subordinate menu items (for instance, "Current" and "Overdue" entries below Accounts > Receivable in the example above), controlling access to the subordinate items must be handled using custom JavaScript. (This would be done in the same way as with custom menus; for more information, see Controlling Access to Subordinate Items.)

Localization for Custom Interfaces

A `customInterfaceLocale.xml` resource file is provided in the `JSXAPPS\base\locale` directory to support localization of text strings associated with custom interfaces.

Empty `customInterfaceLocale.`*`ll_CC`*`.xml` resource files are also provided for various locales supported by Workspace, where *ll* is a lowercase, two-letter ISO 639-1 language code and *CC* is an uppercase, two-letter ISO 3166 country code.

When defining a custom interface menu, menu item, or toolbar button, three text strings may be defined that are displayed in Workspace:

- **Display Text** - This is the label text that displays on the menu, menu item, or toolbar button.
- **Description** - This is the text that describes the custom interface in the Configuration Administrator when configuring user access.
- **Tool Tip** - This is the text that displays when the cursor hovers over the custom interface menu or toolbar button (not applicable to menu items).

If localization of these strings is desired, a record must be created in `customInterfaceLocale.xml` with the English version of the text. For example:

```
<record id="displayText.Accounts" text="Accounts"/>
<record id="description.Accounts" text="Accounts user access"/>
<record id="tooltip.Accounts" text="Open Accounts interface"/>
```

Each record contains two attributes:

- **id** - A unique identifier for the record.
- **text** - The English version of the string that displays in Workspace.

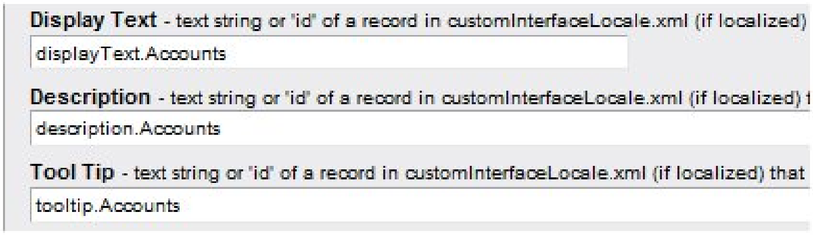
For each language that is to be supported, the record must be duplicated in the corresponding custom interface locale file, with the "text" attribute translated to the appropriate language. For instance, if Workspace is configured to support German, the `customInterfaceLocale.de.xml` file must be edited to include a translated version of the records in `customInterfaceLocale.xml`. For example:

```
<record id="displayText.Accounts" text="Konten"/>
<record id="description.Accounts" text="Konten Benutzer Zugriff"/>
<record id="tooltip.Accounts" text="offene Rechnungen Schnittstelle"/>
```

If you are configuring custom interfaces manually by editing the `customInterfaces.xml` file, enter the appropriate **id** values in the **displayText**, **description**, and **toolTip** attributes. For example:

```
<menu name="Accounts" displayText="displayText.Accounts"
      description="description.Accounts"
      toolTip="tooltip.Accounts"
```

And if you are configuring custom interfaces using the Configuration Administrator, enter the appropriate **id** values in the **Display Text**, **Description**, and **Tool Tip** fields. For example:



Display Text - text string or 'id' of a record in customInterfaceLocale.xml (if localized)

displayText.Accounts

Description - text string or 'id' of a record in customInterfaceLocale.xml (if localized) t

description.Accounts

Tool Tip - text string or 'id' of a record in customInterfaceLocale.xml (if localized) that

tooltip.Accounts

At runtime, if the German language has been selected for Workspace, the localized text in `customInterfaceLocale.de.xml` is displayed. If the record does not exist, the default English version in `customInterfaceLocale.xml` is displayed. If the values specified for Display Text, Description and Tool Tip do not exist in either locale file, the value itself will be displayed. If localization is not desired, the actual text may be entered for these values rather than the "id" of a locale record.

For more information about localizing Workspace, see Localization.

Custom Interface Helper Methods

All custom interfaces of type `<scriptedInterface>` must have a handler method in ...`\JSXAPPS\workspace\application\js\AppMain.js`. The name of the method must match the value of the **source** attribute in the scripted interface definition. The method is automatically invoked when the menu item or toolbar button of the custom interface is selected.

Each handler method must receive an *interfaceObject* as an input parameter, which is a reference to the menu item or toolbar button that launches the custom interface. This *interfaceObject* is then passed to the various "helper" methods that are used to create custom interfaces.

Also see the **loadInterface_sample** method, in ...`\JSXAPPS\workspace\application\js\AppMain.js` for examples of how to use the helper methods.

The following helper methods are available.

- **Attribute Methods** - These methods get and set the attributes used in custom interface definitions:
  - getInterfaceName
  - getInterfaceDisplayText / setInterfaceDisplayText
  - getInterfaceToolTip / setInterfaceToolTip

  - getInterfaceImage / setInterfaceImage
  - getInterfaceFeatures / setInterfaceFeatures
  - getInterfaceVisible / setInterfaceVisible
  - getInterfaceEnabled / setInterfaceEnabled
  - getInterfaceParent
  - getInterfaceLaunchType
  - getInterfaceMenuItems
  - getInterfaceLocation
  - getInterfaceSource
  - getInterfaceSelectedRecords
  - getInterfaceCaption / setInterfaceCaption / showInterfaceCaption

- **Load Methods** - These methods can be used to load each of the different types of custom interfaces:
  - loadSimpleIframe
  - loadSimpleHubIframe
  - loadSimplePrototype
  - loadCustomInterface

- **Close Method** - This method closes a custom interface:
  - closeInterfaceArea

- **Master Selection Area Methods** - These methods are used to control custom interfaces that are defined to display in the Master Selection Area. These methods support the use case of one scripted custom interface changing the attributes of another simple iFrame custom interface. An example of this use case — a button on a work item or events list using an attribute from the selected item in the list to do a tibbr search shown by a custom interface in the Master Selection Area — is illustrated in the **tibbrSearch** sample custom interface (see Sample Custom Interfaces):
  - showMasterSelection
  - getMasterSelectionSource \ setMasterSelectionSource
  - getMasterSelectionDisplayText \ setMasterSelectionDisplayText
  - getMasterSelectionImage \ setMasterSelectionImage
  - getMasterSelectionToolTip \ setMasterSelectionToolTip

All parameters listed for the custom interface methods are required unless otherwise noted as optional. An exception is thrown if a required parameter is not passed in the method call.

getInterfaceName

Returns the unique identifier for the custom interface specified in the scriptedInterface definition.

Syntax

```
getInterfaceName(interfaceObject);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.

Returns

(string) Unique identifier for the custom interface.

getInterfaceDisplayText

Returns the text string that displays on the menu item or toolbar button that launches the custom interface.

Syntax

```
getInterfaceDisplayText(interfaceObject);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.

Returns

(string) Text that displays on the menu item or toolbar button.

setInterfaceDisplayText

Sets the text string that displays on the menu item or toolbar button that launches the custom interface.

Syntax

```
setInterfaceDisplayText(interfaceObject, displayText);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.
- *displayText* - (string) Text to display on the menu item or toolbar button.

Returns

None

getInterfaceToolTip

Returns the text string that displays when the cursor hovers over the menu item or toolbar button that launches the custom interface.

Syntax

```
getInterfaceToolTip(interfaceObject);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.

Returns

(string) Text that displays when the cursor hovers over the menu item or toolbar button that launches the custom interface.

setInterfaceToolTip

Sets the text string that displays when the cursor hovers over the menu item or toolbar button that launches the custom interface.

Syntax

```
setInterfaceToolTip(interfaceObject, toolTip);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.
- *toolTip* - (string) Text to display on the menu item or toolbar button tool tip.

Returns

None

getInterfaceImage

Returns the path to the image that displays on the menu item or toolbar button that launches the custom interface.

Syntax

```
getInterfaceImage(interfaceObject);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.

Returns

(string) The path to the image that displays on the menu item or toolbar button that launches the custom interface.

setInterfaceImage

Sets the path to the image that displays on the menu item or toolbar button that launches the custom interface.

Syntax

```
setInterfaceToolTip(interfaceObject, image);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.
- *image* - (string) Path to the image that displays on the menu item or toolbar button.

Returns

None

getInterfaceFeatures

Returns a comma-separated string containing the positioning and display characteristics of the dialog or browser window that contains the custom interface (see the **features** attribute on page 50).

Syntax

```
getInterfaceFeatures(interfaceObject);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.

Returns

(string) A comma-separated string containing the positioning and display characteristics of the dialog or browser window; returns an empty string if the location of the custom interface is not "browser" or "dialog".

setInterfaceFeatures

Sets the positioning and display characteristics of a dialog or browser window that contains the custom interface.

Syntax

```
setInterfaceFeatures(interfaceObject, features);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.
- *features* - (string) Comma-separated string containing the positioning and display features to apply to a dialog or browser that contains a custom interface (see the **features** attribute on page 50).

Returns

None

getInterfaceVisible

Returns true if the menu or toolbar button that launches the custom interface is visible, or false if it is hidden.

Syntax

```
getInterfaceVisible(interfaceObject);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.

Returns

(boolean) True if the menu or toolbar button that launches the custom interface is visible, or false if it is hidden.

setInterfaceVisible

Hides or shows the menu or toolbar button that launches the custom interface.

For menus, the entire menu is hidden, not individual menu items.

Syntax

```
setInterfaceVisible(interfaceObject, visible);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.
- *visible* - (boolean) True or false to show or hide the menu or toolbar button that launches the custom interface (for menus, the entire menu is hidden, not individual menu items).

Returns

None

getInterfaceEnabled

Returns true if the menu, menu item, or toolbar button that launches the custom interface is visible, or false if it is hidden.

Syntax

```
getInterfaceEnabled(interfaceObject, menuItem);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.
- *menuItem* - (string) (Optional) The unique identifier (name) of a menu item. If not specified, the method returns whether or not the menu or the toolbar button is enabled.

Returns

(boolean) True if the menu, menu item, or toolbar button that launches the custom interface is enabled, or false if it is hidden.

setInterfaceEnabled

Enables or disables the menu, menu item, or toolbar button that launches the custom interface.

Syntax

```
setInterfaceEnabled (interfaceObject, enable, menuItem);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.
- *enable* - (boolean) True to enable, or false to disable, the menu, menu item or toolbar button that launches the custom interface.
- *menuItem* - (string) (Optional) The unique identifier (name) of a menu item. If not specified, the method enables or disables the entire menu or the toolbar button; if specified, the menu item is enabled or disabled.

Returns

None

getInterfaceParent

Returns the name of parent element, in `customInterface.xml`, for a custom interface menu item or toolbar button. The parent element represents the various locations, in the Workspace application, from which custom interfaces may be launched.

Syntax

```
getInterfaceParent(interfaceObject);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.

Returns

(string) The name of parent element, in `customInterface.xml`, for a custom interface menu item or toolbar button. Returns one of the following:

- WorkspaceToolbar
- WorkspaceMasterSelection
- WorkViewsList
- WorkItemsList
- ProcessViewsList
- ProcessInstancesList
- EventViewsList
- EventsList
- BusinessServicesList
- ProcessTemplatesList

getInterfaceLaunchType

Returns either 'menu' or 'toolbarButton' to indicate what type of object launched the custom interface.

Syntax

```
getInterfaceLaunchType(interfaceObject);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.

Returns

(string) Either 'menu' or 'toolbarButton' to indicate what type of object launched the custom interface.

getInterfaceMenuItems

Returns an array containing the names of the menu items within a menu.

Syntax

```
getInterfaceMenuItems(interfaceObject);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface. The method throws an exception if *interfaceObject* is not a menu object.

Returns

(array of strings) The names of the menu items within a menu.

getInterfaceLocation

Returns the custom interface location specified in the scriptedInterface definition.

Syntax

```
getInterfaceLocation(interfaceObject);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.

Returns

(string) Either "areaA", "areaB", "areaC", "areaD", "dialog", "browser", or "script".

getInterfaceSource

Returns the custom interface source specified in the scriptedInterface definition.

Syntax

```
getInterfaceSource(interfaceObject);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.

Returns

(string) Either a URL for simpleIframe and simpleHubIframe interfaces, a path to a TIBCO General Interface prototype for simplePrototype interfaces, or a method name for scripted interfaces.

getInterfaceSelectedRecords

Returns a collection of XML-entity records for all selected items in the Workspace list where the menu item or toolbar button that launches the interface is located. The attributes in these records contain list data that may be used as URL parameters or input to the custom interface.

Syntax

```
getInterfaceSelectedRecords(interfaceObject);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface. The method throws an exception if the list associated with *interfaceObject* is not one of the following:
  - WorkViewsList
  - ProcessViewsList

- EventViewsList
- BusinessServicesList
- WorkItemsList
- ProcessInstancesList
- EventsList
- ProcessTemplatesList

Returns

(collection of XML records) The length of the collection can be obtained with the `getLength()` method, and individual records can be obtained with the `getItem(`*`n`*`)` method, where *n* is the zero-based index of a record in the collection. Attributes values can be obtained, from records, with the `getAttribute('name')` method, where *name* is the name of the attribute.

getInterfaceCaption

Returns a string containing the HTML that displays an image and text in the caption area of the container for the custom interface.

Syntax

```
getInterfaceCaption(containerObject);
```

Parameters

- *containerObject* - (Object) The parent container of the custom interface; returned by the `loadSimpleIframe`, `loadSimpleHubIframe`, `loadSimplePrototype`, and `loadCustomInterface` methods.

Returns

(string) HTML that displays an image and text in the caption area of the container for the custom interface.

setInterfaceCaption

Sets the image and text that display in the caption area of the container for the custom interface. By default, the image and displayText settings for the custom interface are displayed in the caption. This method provides a means for changing the caption. Note that for interfaces opened in a browser window, only the text parameter is applicable; the image parameter is ignored.

Syntax

```
setInterfaceCaption(containerObject, text, image);
```

Parameters

- *containerObject* - (Object) The parent container of the custom interface; returned by the `loadSimpleIframe`, `loadSimpleHubIframe`, `loadSimplePrototype`, and `loadCustomInterface` methods.
- *text* - (string) Text that displays in the caption.
- *image* - (string) Path to the image that displays in the caption.

Returns

None

showInterfaceCaption

Hides or shows the caption bar on a custom interface that has been opened in 'areaA', 'areaB', 'areaC' or 'areaD'. Note that hiding the caption bar will also hide the button that closes the custom interface.

Syntax

```
showInterfaceCaption(containerObject, show);
```

Parameters

- *containerObject* - (Object) The parent container of the custom interface; returned by the `loadSimpleIframe`, `loadSimpleHubIframe`, `loadSimplePrototype`, and `loadCustomInterface` methods.
- *show* - (boolean) True to show, or false to hide, the caption bar.

Returns

None

loadSimpleIframe

Loads a simple iFrame into a location within Workspace.

Syntax

```
loadSimpleIframe(interfaceObject, source, location);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.
- *source* - (string) A source URL for the simple Iframe. If null or an empty string, the source specified in `customInterfaces.xml` is used.
- *location* - (string) (Optional) Either "areaD", "dialog", or "browser". If null or an empty string, the location specified in `customInterfaces.xml` is used, however, if "script" was specified in the definition, this parameter must be supplied, and an exception is thrown if location is not provided.

  The method throws an exception if the specified location is not valid, given the location of the menu item or toolbar button (*interfaceObject*) that launches the interface (see Valid Custom Interface Locations).

Returns

Returns the parent container for the iFrame that has been loaded, as follows:

- jsx3.gui.Dialog (if 'dialog' location is specified)
- jsx3.gui.Window (if 'browser' location is specified)
- jsx3.gui.Block (if 'areaD' location is specified)

loadSimpleHubIframe

Loads a simple Managed Hub iFrame into a location within Workspace.

Syntax

```
loadSimpleHubIframe(interfaceObject, source, subscribe, publish, location);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.
- *source* - (string) A source URL for the web application of the Managed Hub iFrame. If null or an empty string, the source specified in `customInterfaces.xml` is used.
- *subscribe* - (string) A comma-separated string of event topics to which this custom interface subscribes. If null, the subscriptions specified in `customInterfaces.xml` are used. If no subscriptions are desired, an empty string must be passed.
- *publish* - (string) A comma-separated string of event topics which this custom interface publishes. If null, the publish events specified in `customInterfaces.xml` are used. If no publish events are desired, an empty string must be passed.
- *location* - (string) (Optional) Either "areaD", "dialog", or "browser". If null or an empty string, the location specified in `customInterfaces.xml` is used, however, if "script" is specified in the definition, this parameter must be supplied, and an exception is thrown if location is not provided.

  The method throws an exception if the specified location is not valid, given the location of the menu item or toolbar button (*interfaceObject*) that launches the interface (see Valid Custom Interface Locations).

Returns

Returns the parent container for the Managed Hub iFrame that has been loaded, as follows:

- jsx3.gui.Dialog (if 'dialog' location is specified)
- jsx3.gui.Window (if 'browser' location is specified)
- jsx3.gui.Block (if 'areaD' location is specified)

loadSimplePrototype

Loads a TIBCO General Interface prototype into a location within Workspace.

Syntax

```
loadSimplePrototype(interfaceObject, source, location);
```

Parameters

- *interfaceObject* - (Object) Object that launches the custom interface.
- *source* - (string) A path to the TIBCO General Interface prototype. If null or an empty string, the source specified in `customInterfaces.xml` is used.
- *location* - (string) (Optional) Either "areaA", "areaB", "areaC", "areaD", "dialog", or "browser". If null or an empty string, the location specified in the custom interface definition is used, however, if "script" is specified in the definition, this parameter must be supplied, and an exception is thrown if location is not provided.

  The method throws an exception if the specified location is not valid, given the location of the menu item or toolbar button (*interfaceObject*) that launches the interface (see Valid Custom Interface Locations).

Returns

Returns the parent container for the prototype that has been loaded, as follows:

- jsx3.gui.Dialog (if 'dialog' location is specified)

- jsx3.gui.Window (if 'browser' location is specified)
- jsx3.gui.Block (if 'areaA', 'areaB', 'areaC' or 'areaD' location is specified)

loadCustomInterface

Loads a custom interface into a specified parent container. This method is useful for custom WCC applications that utilize custom interfaces, but do not have locations for displaying interfaces that correspond to the locations in the Workspace application ('Area A', 'Area B', etc.).

This method is also useful when running custom scripts in the Workspace application that must dynamically load custom interfaces that are not defined in `customInterfaces.xml`.

Syntax

```
loadCustomInterface(type, source, location, features, subscribe, publish);
```

Parameters

- *type* - (string) 'simpleIframe', 'simpleHubIframe', or 'simplePrototype'.
- *source* - (string) A source URL if *type* is 'simpleHubIframe' or 'simpleIframe', or a path to a TIBCO General Interface prototype if *type* is 'simplePrototype'.
- *location* - (string or Object) Parent container for the custom interface. Either 'dialog', 'browser', 'areaA', 'areaB', 'areaC', or 'areaD' if loading the interface into the Workspace application. An exception is thrown if the location is not valid for the type of interface (see Valid Custom Interface Locations).

  If loading the interface in a custom WCC application, this is a reference to the parent container that is to display the interface. Note that an exception is thrown if the parent container does not support the `setChild` method.
- *features* - (string) If *location* is 'dialog' or 'browser', this is a comma-separated string containing the positioning and display features to apply (see the **features** attribute on page 50).
- *subscribe* - (string) This is only applicable if *type* is 'simpleHubIframe'. A comma-separated string of event topics to which this interface subscribes. If no subscriptions are desired, an empty string must be passed.
- *publish* - (string) This is only applicable if type is 'simpleHubIframe'. A comma-separated string of event topics that this interface publishes. If no publish events are desired, an empty string must be passed.

Returns

The return object is the parent container of the custom interface that has been loaded.

closeInterfaceArea

Removes the parent container and closes the custom interface area. This is for a custom interface that has been opened in 'areaA', 'areaB', 'areaC', or 'areaD' of the Workspace application,

Syntax

```
closeInterfaceArea(containerObject);
```

Parameters

- *containerObject* - (Object) The parent container of the custom interface; returned by the `loadSimpleIframe`, `loadSimpleHubIframe`, `loadSimplePrototype`, and `loadCustomInterface` methods.

Returns

None

showMasterSelection

Causes a custom interface to be displayed in the WorkspaceMasterSelection area, just as it would had the user clicked on the toolbar button in the Workspace master selection area.

Syntax

```
showMasterSelection(interfaceName);
```

Parameters

- *interfaceName* (string) The unique name of the interface to display.

Returns

None

getMasterSelectionSource

Returns the value of the **source** attribute for a custom interface defined to display in the WorkspaceMasterSelection area.

Syntax

```
getMasterSelectionSource(interfaceName);
```

Parameters

- *interfaceName* - (string) The unique name of the custom interface.

Returns

(string) The value of the **source** attribute for the custom interface.

setMasterSelectionSource

Sets the value of the **source** attribute for a custom interface defined to display in the WorkspaceMasterSelection area.

Syntax

```
setMasterSelectionSource(interfaceName, source, show);
```

Parameters

- *interfaceName* - (string) The unique name of the custom interface.
- *source* - (string) The value of the **source** attribute for the custom interface.
- *show* - (boolean) If set to true, the method automatically displays the custom interface with the newly specified source, otherwise, the source is set, but the `showMasterSelection` method must be called to display the custom interface.

Returns

None

getMasterSelectionDisplayText

Returns the value of the **displayText** attribute (text string that displays on the toolbar button that launches the custom interface) of a custom interface defined to display in the WorkspaceMasterSelection area.

Syntax

```
getMasterSelectionDisplayText(interfaceName);
```

Parameters

- *interfaceName* - (string) The unique name of the custom interface.

Returns

(string) The value of the text string that displays on the toolbar button that launches the custom interface.

setMasterSelectionDisplayText

Sets the value of the **displayText** attribute (text string that displays on the toolbar button that launches the custom interface) of a custom interface that is defined to display in the WorkspaceMasterSelection area.

Syntax

```
setMasterSelectionDisplayText(interfaceName, displayText);
```

Parameters

- *interfaceName* - (string) The unique name of the custom interface.
- *displayText* - (string) The value the text string to display on the toolbar button that launches the custom interface.

Returns

None

getMasterSelectionImage

Returns the value of the **image** attribute (the path to the image that displays on the toolbar button that launches the custom interface) for a custom interface that is defined to display in the WorkspaceMasterSelection area.

Syntax

```
getMasterSelectionImage(interfaceName);
```

Parameters

- *interfaceName* - (string) The unique name of the custom interface.

Returns

(string) The path to the image that displays on the toolbar button that launches the custom interface.

setMasterSelectionImage

Sets the value of the **image** attribute (the path to the image that displays on the toolbar button that launches the custom interface) for a custom interface that is defined to display in the WorkspaceMasterSelection area.

Syntax

```
setMasterSelectionImage(interfaceName, image);
```

Parameters

- *interfaceName* - (string) The unique name of the custom interface.
- *image* - (string) The path to the image to display on the toolbar button that launches the custom interface.

Returns

None

getMasterSelectionToolTip

Returns the value of the **toolTip** attribute (the text string that displays when the cursor hovers over the toolbar button that launches the custom interface) for a custom interface that is defined to display in the WorkspaceMasterSelection area.

Syntax

```
getMasterSelectionToolTip(interfaceName);
```

Parameters

- *interfaceName* - (string) The unique name of the custom interface.

Returns

(string) The text string that displays when the cursor hovers over the toolbar button that launches the custom interface.

setMasterSelectionToolTip

Sets the value of the **toolTip** attribute (the text string that displays when the cursor hovers over the toolbar button that launches the custom interface) for a custom interface that is defined to display in the WorkspaceMasterSelection area.

Syntax

```
setMasterSelectionToolTip(interfaceName, toolTip);
```

Parameters

- *interfaceName* - (string) The unique name of the custom interface.
- *toolTip* - (string) The text string to display when the cursor hovers over the toolbar button that launches the custom interface.

Returns

None

Sample Custom Interfaces

Samples are provided that illustrate each of the custom interface types. A directory for each custom interface type is located in the following parent directory:

`StudioHome\wcc\version\Samples\CustomInterfaces\`

where:

- `StudioHome` is the directory in which TIBCO Business Studio was installed.
- *version* is the version number of Workspace that was installed with TIBCO Business Studio.

simpleIframe Sample

This custom interface sample illustrates opening a website in an Iframe in the Workspace application.

The sample specifically adds a menu with two menu items to the Workspace toolbar, as well as a toolbar button to the master selection area.

The first menu item causes the http://www.tibco.com website to open in an Iframe in a dialog.

The second menu item causes the http://spotfire.tibco.comwebsite to open in an Iframe in a separate browser window.

The toolbar button in the master selection area causes the http://tibco.tibbr.comwebsite to open an Iframe in "Area D" of the application.

The elements and attribute values needed to implement this sample are provided in the `readme.txt` file in the following directory:

`StudioHome\wcc\version\Samples\CustomInterfaces\simpleIframe`

You can modify the provided elements and attribute values to fit your particular requirements.

simpleHubIframe Sample

This custom interface sample illustrates how to load a web application into a Managed Hub Iframe. The web application then subscribes to events published by WCC components.

To implement this sample, you must create a web application named 'HubIframeSample' using the contents of the `HubIframeSample` folder provided with the sample, then host that application on a server (the server does not have to be in ActiveMatrix or ActiveMatrix BPM).

The sample causes a toolbar button to display on the Workspace application work view list. This button launches the web application in a Managed Hub Iframe. The web application displays a list of events received when items are selected in the work view, process view, or event view lists.

The elements and attribute values needed to implement this sample are provided in the `readme.txt` file in the following directory:

`StudioHome\wcc\version\Samples\CustomInterfaces\simpleHubIframe`

You can modify the provided elements and attribute values to fit your particular requirements.

simplePrototype

This custom interface sample illustrates how to load a TIBCO General Interface prototype into a specified location when the launch component is executed.

This sample addresses the use case of a work item delivery interface, which automatically opens and presents work items to a user. The user controls when the automated process begins and ends.

When the user activates the automated process by clicking a button, the interface delivers the first available work item based on specified sort and filter parameters. When the work item has been processed and the form is closed, the next available item is automatically delivered. The user can discontinue the automated process by clicking the same button used to start the process.

The delivery process consists of two custom interfaces:

- A TIBCO General Interface prototype that provides a user interface for starting and stopping the automated process, and
- a prototype for displaying the work item form in the application.

Both of these prototypes are provided in the `\prototypes` sub-folder that is included with this sample.

`*StudioHome*\wcc\*version*\Samples\CustomInterfaces\simplePrototype`

The elements, attributes, and methods needed to implement this sample are provided in the `readme.txt` file in the following directory:

You can modify the provided elements and attribute values to fit your particular requirements.

This sample can be used in conjunction with user access controls to present only the work item delivery On/Off button to the user, preventing the user from performing any other functions.

scriptedInterface

This consists of four separate samples of custom interfaces that are managed in custom JavaScript methods. For each of these samples, a ’source’ attribute is specified in the `customInterfaces.xml` file that contains the name of a custom method that must be created in `JSXAPPS\workspace\application\js\AppMain.js`.

The custom method is automatically invoked when the menu or toolbar button that launches the custom interface is executed. This custom method may utilize "helper" methods to control display characteristics of the tool that launches the interface, as well as manage the loading, placement, and display of the custom interface. The available helper methods are described in Custom Interface Helper Methods.

The scriptedInterface samples are:

- **Helper Methods** - Demonstrates getting and setting attribute values of the menu or toolbar button that launches the custom interface.
- **simpleIframe** - Demonstrates loading a simpleIframe custom interface. The http://www. bing.com maps website is loaded into an iFrame in "Area D" of the Workspace layout. An "address" is added as a parameter on the URL, specifying the location to display on the map. This address is obtained from the value of the 'Attribute2' attribute, of the selected item in a list of work items, in the user's Inbox.
- **simpleHubIframe** - Demonstrates loading a simpleHubIframe custom interface by loading a web application into a Managed Hub iFrame. The application subscribes to selected WCC Component PageBus events and displays a list of the events received.
- **simplePrototype** - Demonstrates loading a simplePrototype custom interface by loading a TIBCO General Interface prototype into "Area C" of the Workspace application layout.

Specific information about implementing these samples is provided in the `readme.txt` file in the following directory:

`*StudioHome*\wcc\*version*\Samples\CustomInterfaces\scriptedInterface`

Each of the samples can be modified to fit your particular requirements.

tibbrSearch

This custom interface sample supports the use case of one scripted custom interface changing the attributes of another simple Iframe custom interface. Specifically, the sample extracts an attribute value from the selected item on the work item or events list, then uses that value for the subject of a tibbr search by a custom interface in the master selection area.

The source URL for the simpleIframe custom interface in the master selection area is initially set to "`https://tibco.tibbr.com`". Scripted custom interfaces on the work item and events lists use helper methods to extract attribute values, for the selected item in the list, and constructs a `tibbr.com` search URL. The source of the simpleIframe is then set to this search URL and reloaded.

The elements, attributes, and methods needed to implement this sample are provided in the `readme.txt` file in the following directory:

StudioHome\wcc\version\Samples\CustomInterfaces\tibbrSearch

You can modify the provided elements and attribute values to fit your particular requirements.

Custom Views

Supervisors can create custom work views, process views, or event views, then assign them to individual users, or to members of specified groups and positions; these custom views are called *system views*. Users that have been assigned a system view automatically get a copy of the view in their view list, along with their own user-defined views.

System views are defined with the view wizard, in much the same way as user-defined views are created, by specifying a name, description, filter, sort, columns, etc. The view wizard, however, contains an additional dialog that is used to specify the resources, groups, or positions that will inherit the system view, as well as which characteristics of the system view can be modified by the users that inherit them.

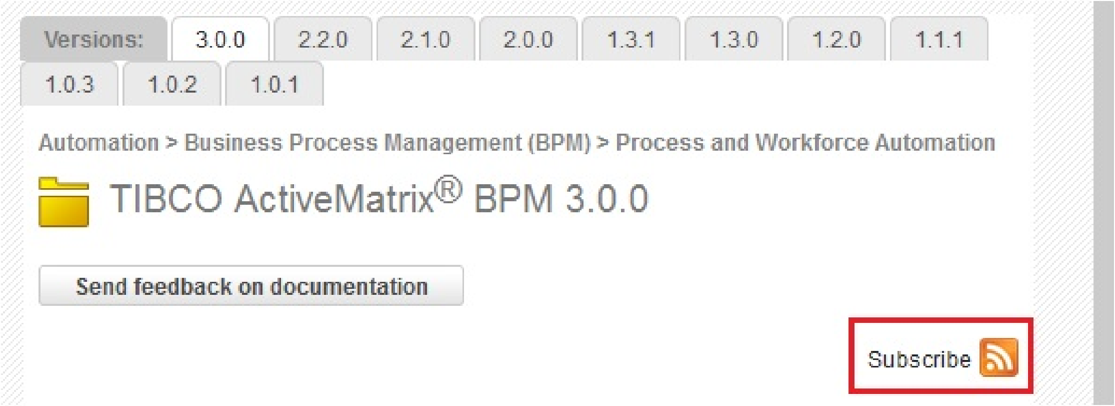
Because the system view wizard is accessed via the application, creating system views is described in detail in the "Using Views" chapter in the *Workspace User's Guide*.

Customizing Location of Help Files

By default, all TIBCO Workspace and Organization Browser documentation is available on the TIBCO Documentation web site. You can, however, download the documentation and store it locally, then configure Workspace so that the local documentation is accessed when a user selects **Help** > **Help** or **Help** > **Organization Browser Help** in the application.

This is typically used at sites that do not allow Internet access.

If you download and store the Workspace and Organization Browser documentation, you may not be aware of changes that are made to those documents on the TIBCO Documentation web site. Occasionally, out-of-cycle (OOC) updates are published for some documents, which means that they are updated between software releases. To be notified when OOC updates occur, you can subscribe to RSS feeds by using the **Subscribe** button on the "TIBCO ActiveMatrix BPM" page of the TIBCO Documentation web site:


Versions:
3.0.0
2.2.0
2.1.0
2.0.0
1.3.1
1.3.0
1.2.0
1.1.1
1.0.3
1.0.2
1.0.1
Automation > Business Process Management (BPM) > Process and Workforce Automation
TIBCO ActiveMatrix® BPM 3.0.0
Send feedback on documentation
Subscribe

Note, however, you cannot subscribe to RSS feeds for only the Workspace and Organization Browser documentation; this notifies you of changes to any TIBCO ActiveMatrix BPM documentation.

Procedure

1. Download the TIBCO ActiveMatrix BPM documentation from the TIBCO Documentation web site.

   Note that you cannot download just the Workspace and Organization Browser documentation; it is bundled with all of the TIBCO ActiveMatrix BPM documentation.

   a) With a browser, navigate to:

      https://docs.tibco.com/products/tibco-activematrix-bpm

      Note that this URL always takes you to the documentation for the most recently released version of TIBCO ActiveMatrix BPM. If you want the documentation for an earlier version, click the appropriate tab on the page.

   b) Click **Download All**.

2. Unzip the file that you downloaded and place the files in a local file system or host them on an internal web server.

3. Double-click the `index.html` file in the `html` directory.

4. In the Table of Contents, navigate to the desired guide and select it:

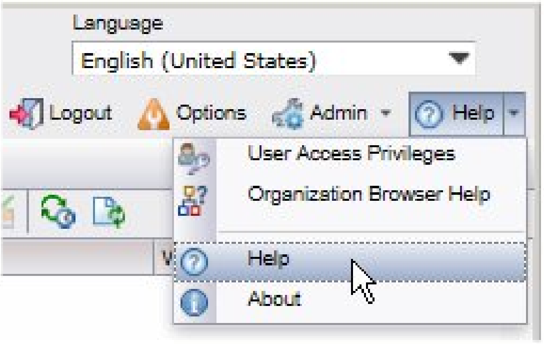
   - **Workspace** > **Workspace User's Guide**
   - **Workspace** > **Organization Browser User's Guide**

5. Copy the file path / URL from the address line in the browser, then paste that into the appropriate **`HelpFileLocations`** parameter.

   - If you are using the Configuration Administrator to configure the location of help files, see "Help File Locations" in Application.
   - If you are directly updating the `config.xml` file to configure the location of help files, see Help File Locations.

Result

Now when a user selects **Help** > **Help** or **Help** > **Organization Browser Help** from within the application, the appropriate help is opened from the local server.



openworkitem Application

The openworkitem application illustrates how to either open a work item or start a business service by passing parameters on the URL, rather than by making a selection from a list.

The openworkitem application presents the user with the login screen. After the user is authenticated, either the work item specified in the URL is opened, or the business service specified is started.

When the user submits, cancels, or closes the work item or starts the business service, the browser window is closed.

If the openworkitem application is used with Mozilla Firefox, a configuration setting must be set to the correct value for the window to close properly when the user submits, cancels, or closes the work item. To check or change the value of this setting, enter "about:config" as the URL in Firefox. Scroll down the list to the dom.allow_scripts_to_close_windows item under the Preference Name column. If the value is currently set to false, right click it and select Toggle to change the value to true.

The system uses the openworkitem application when a work item is *pushed* to a user via an email.

Configuring the openworkitem Application

The openworkitem application must be configured prior to using it.

Procedure

1. Open the application's configuration file, which is located as follows:

   ```
   StudioHome\wcc\version\JSXAPPS\openworkitem\config.xml
   ```

   where:

   - *StudioHome* is the directory in which TIBCO Business Studio was installed.
   - *version* is the version number of Workspace that was installed with TIBCO Business Studio.

2. Locate the **ActionProcessors** record in the `config.xml` file.
3. Set the **baseUrl** attribute to the URL of the Action Processor. The string in the **baseUrl** attribute must be in the form:

   ```
   http://Host:Port/bpm/actionprocessor/actionprocessor.servlet
   ```

   where:

   - `Host` is the name or IP address of the machine hosting the Action Processor. This must be the same machine on which the application is running (i.e., the Action Processor and the application must be running on the same machine).
   - `Port` is the port number used by the ActiveMatrix WebApp Implementation Type to communicate with web applications.

     Note - The **weighting** attribute is not used at this time.

4. Save and close the `config.xml` file.The

Opening a Work Item with the openworkitem Application

Procedure

- Invoke the openworkitem application by passing the following parameters in the browser URL:

  ```
  http://Host:Port/workspace/openworkitem.html?
  Id=workItemId&Version=versionNumber&ChannelId=channelId
  ```

where:

- *Host* is the name or IP address of the machine hosting the Action Processor. This must be the same machine on which the application is running (i.e., the Action Processor and the application must be running on the same machine).
- *Port* is the port number used by the ActiveMatrix WebApp Implementation Type to communicate with web applications.
- *workItemId* identifies the work item to open.
- *versionNumber* (optional) is the work item version, which indicates how many times the work item has changed state. The version number starts at 0 when the work item is created, and is incremented by one each time it changes state. If omitted, the current version is used.
- *channelId* (optional) identifies which presentation channel to use when displaying work item forms when using an instance of that application. If omitted, the channel ID specified in the **ChannelIdentifier** in the **openworkitem** application's `config.xml` file is used.

The following are examples of opening a work item with the **openworkitem** application:

- http://MyServer:8080/workspace/openworkitem.html?Id=23
- http://MyServer:8080/workspace/openworkitem.html?Id=23&Version=4&ChannelId=Default_FORM_Channel

Starting a Business Service with the openworkitem Application

Using the openworkitem application, you can start a business service that either contains, or does not contain, formal parameters (use the optional *dataFeed* parameter to pass formal parameters in the method call).

Procedure

- Invoke the **openworkitem** application by passing the following parameters in the browser URL:

```
http://Host:Port/workspace/openworkitem.html?
ProcessName=processName&ModuleName=moduleName?
Version=versionNumber&DataFeed=dataFeed&Description=description
```

where:

- *Host* is the name or IP address of the machine hosting the Action Processor. This must be the same machine on which the application is running (i.e., the Action Processor and the application must be running on the same machine).
- *Port* is the port number used by the ActiveMatrix WebApp Implementation Type to communicate with web applications.
- *processName* identifies the business service to start.
- *moduleName* (optional) is the path to the XPDL file that defines the "process package". Probably the easiest way to obtain this path would be to use the Workspace application and observe the Post and Response while navigating through the available business services. The response will include the <ap:ModuleName> element, which contains the path to the module (for example: `<ap:ModuleName>/Acme-Contact/Process Packages/ProcessPackage.xpdl</ap:ModuleName>`). If omitted, the moduleName is retrieved from the server.
- *versionNumber* (optional) is the version of the process template from which the business service is to be started. If omitted, the current *versionNumber* is used.
- *dataFeed* (optional) is data (formal parameters) used when starting the business service. This must be in JSON format.
- *description* (optional) is a description that is displayed on the form caption bar.

The following are examples of starting a business service with the **openworkitem** application:

- http://MyServer:8080/workspace/openworkitem.html?ProcessName=CSCallbackStart&ModuleName=%2FCSCallback%2FProcess%20Packages%2FCSCallback.xpdl&Version=1.0.0.201101101507
- http://MyServer:8080/workspace/openworkitem.html?ProcessName=CSCallbackStart&ModuleName=%2FCSCallback%2FProcess%20Packages%2FCSCallback.xpdl&Version=1.0.0.201101101507&Description=My%20business%20service%20description
- http://MyServer:8080/workspace/openworkitem.html?ProcessName=HelpDeskStart&ModuleName=%2FHelpDesk%2FProcess%20PackagesformalFHelpDesk.xpdl&Version=1.0.0.201101101430&DataFeed={ "items": [{"$param":"myText", "mode":"INOUT", "type":"STRING", "$value":"abc"}, {"$param":"myInt", "mode":"INOUT", "type":"NUMBER", "$value":"123"}]}

Spaces and slashes in the parameters passed on the URL must be URL encoded (space = %20, forward slash = %2F).

External Login

If there is already a login session, you can use the “externalLogin” feature in conjunction with the openworkitem application.

The following are examples of using this feature to open a work item or start a business service:

- Opening a Work Item
  - http://MyServer:8080/workspace/openworkitem.html?**externalLogin=true**?Id=51
- Starting a Business Service
  - http://MyServer:8080/workspace/openworkitem.html?**externalLogin=true**?ProcessName=CSCallbackStart&ModuleName=%2FCSCallback%2FProcess%20Packages%2FCSCallback.xpdl&Version=1.0.0.201101101507

For more information about external login, see the *TIBCO Workspace Components Developer Guide*.

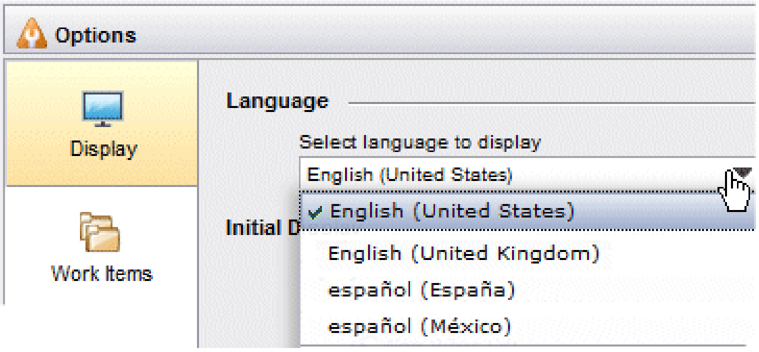
The Login Session sample application also illustrates other methods of establishing and maintaining a login session when using the **openworkitem** application. For more information, see Login Session Sample Application .

Localization

TIBCO has *language packs* available for selected languages that, when installed, localize your client application to the language for that language pack. Language packs have their own installer that handles everything that is described here.

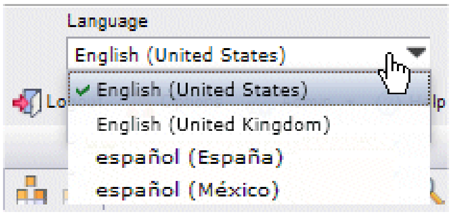
If the desired language is not available in a language pack, you can use the procedures provided in this document to manually localize your client application. This involves creating *resource files* that contain the translated strings, then deploying the resource files to your node so they are available in the deployed application.

After the appropriate resource files have been created and deployed, the user can select the language from the Options dialog...



... or dynamically from the **Languages** field on the main toolbar while running the application:

Both of these methods of choosing the language use the **setAppLocale** Application class method. For more information, see the *TIBCO Workspace Components Developer Guide*.



Default Language

A **localeKey** configuration parameter is available in the `config.xml` file that allows you to specify the default language used by the application. If your application has been localized, either via a language pack, or manually as described in this document, you may want to set the default language to the localized language.

The **LocaleKey** parameter can be set to a language code (for example, “es” for Spanish) or to a country-specific code (for example, “es_MX” for Spanish Mexico).

If the **LocaleKey** parameter is not set, the default language is set to that of the browser. If the **LocaleKey** parameter is set to a language code or country-specific code that does not match an installed language pack, or a manually localized locale, the default language defaults to “en_US” (English United States).

For more information about the **localeKey** parameter, see Options.

Language Resource Files

WCC applications use language *resource files* to store text strings, which are displayed in the application at runtime.

The following types of resource files required for each language:

- `locale.`*`ll`*`.xml` - These are the “general” resource files that contain text strings that are used throughout the application.
- `eventLocale.`*`ll`*`.xml` - These resource files contain text strings that are used specifically in the event view list and the Event Viewer. (For more information about how these event-related strings are used, see Event-Related Text.)
- `customInterfaceLocale.`*`ll`*`.xml` - These resource files contain text strings that are used with custom interfaces. (For more information about how these strings are used, see Localization for Custom Interfaces.)

  where:

  – *ll* is a lowercase, two-letter ISO 639-1 language code (for example, `locale.es.xml` for Spanish). For a list of language codes, see:

    http://www.loc.gov/standards/iso639-2/php/English_list.php

Note that unlike resource files for all other languages, the default English (United States) resource files do not include the language code in their names. The default English (United States) resource files are:

```
locale.xml
eventLocale.xml
customInterfaceLocale.xml
```

The locale file names may also include a “country code” following the language code. For example “`locale.fr_FR.xml`”. For information, see Country-Specific Translations .

Resource files consist of records with attributes that contain localized strings that are displayed in the application at runtime. For example:

```
...
<record jsxid="lblRefreshing" jsxtext="Refreshing..."/>
<record jsxid="lblSaveView" jsxtext="Save View"/>
<record jsxid="lblSaveViewAs" jsxtext="Save View As..."/>
...
```

The resource files are located in the following directory:

```
...\JSXAPPS\base\locale\
```

By default, the only resource files that contain text strings are the English (United States) locale files: `locale.xml` and `eventLocale.xml`.

All of the other provided resource files (for example, `locale.de.xml`, `eventLocale.de.xml`, etc.) are templates that do not contain any text strings—these are used as templates to create your own localized resource files.

Modifying an Existing Resource File

As this information is primarily about creating new resource files for a language, you are also free to modify the text strings in any of the existing resource files. Simply open the desired resource files in an editor, and modify the text strings as desired.

Language Resource File Deployment

Resource files created as described here must be deployed to the node to be usable in a deployed application. This can be done in a number of ways, depending on whether you already have a deployed

Workspace application, an already deployed custom WCC application, or a custom WCC application that is not yet deployed.

- **Deployed Workspace application** - The Workspace application that is installed with a BPM runtime system has an *extension point* defined in the composite. You can use this extension point to add or update files in the deployed application at a later time. Resource files can be added to the deployed Workspace application using this extension point—this process is described in the steps in Deploying Language Resource Files.
- **Deployed custom WCC application** - The way in which you deploy resource files to an already deployed custom WCC application depends on whether or not an extension point was defined in the application when it was deployed.
  - If an extension point had been defined, you can deploy the resource files using the extension point, just like you would for a deployed Workspace application, except you would use the extension point name that was specified when it was defined in the custom WCC application. In this situation, you can use the same procedure as described in Deploying Language Resource Files, except in step 21 you would use the name of the extension point for that application rather than the one shown for the Workspace application.
  - If an extension point had not been defined in the custom WCC application, the application would need to be undeployed, the resource files manually added to the appropriate directory, then the application re-deployed (when re-deployed, an extension point could be defined for later use to update files in the deployed application).
- **Undeployed custom WCC application** - In an undeployed custom WCC application, resource files can be manually added to the appropriate directories—when the application is deployed, those files are available to the deployed application, providing translated strings.

  When the custom WCC application is deployed, an extension point can be defined for later use to update files in the deployed application.

For information about deploying custom WCC application, as well as defining an extension at that time, see Deploying an Application After Customizing.

Country-Specific Translations

Resource files can also be set up to provide country-specific text strings.

For instance, you could provide translated strings for:

- Spanish
- Spanish (Spain)
- Spanish (Mexico)

This is done somewhat differently for the “general” resource files (those that provide text strings used throughout the application) than it is for the “event” resource files (those that provide text strings used in the event view list and the Event Viewer), as described below.

General Resource Files

Country-specific text strings are embedded in the general resource files (for example, `locale.es.xml`) under separate **key** attributes.

For example, the following shows a `locale.es.xml` resource file that contains country-specific text strings for both Spain and Mexico:

```
<data jsxnamespace="propsbundle"
    <locale key="es">
      <!-- DEFAULT LANGUAGE RECORD ELEMENTS HERE -->
      ...
      <record jsxid="txtClose" jsxtext="Cierre"/>
      <record jsxid="txtOpen" jsxtext="Abierto"/>
      ...
    </locale>
    <locale key="es_ES">
      <record jsxid="txtOpen" jsxtext="override value for Spain"/>
    </locale>
    <locale key="es_MX">
      <record jsxid="txtOpen" jsxtext="override value for Mexico"/>
    </locale>
</data>
```

The value in the **key** attribute identifies the language or the language and the country. These must be in the form:

```
ll
```

or

```
ll_CC
```

where:

- *ll* is a lowercase, two-letter ISO 639-1 language code (for example, `locale.es.xml` for Spanish). For a list of language codes, visit the following web site:

  http://www.loc.gov/standards/iso639-2/php/English_list.php

- *CC* is an uppercase, two-letter ISO 3166 country code. For a list of country codes, visit the following web site:

  http://www.iso.org/iso/country_codes/iso_3166_code_lists/country_names_and_code_elements.htm

The **key** value at the top of the file always identifies the general language, for example, Spanish (**key=”es”**). The text strings under that **key** are used when the general language is selected in the application (for example, Spanish).

The **key** values at the bottom of the file indicate there are override values that are country-specific: if Spanish (Spain) is chosen in the application, the strings under **key=”es_ES”** are used; if Spanish (Mexico) is chosen in the application, the strings under **key=”es_MX”** are used.

Event Resource Files

Country-specific text strings for events must be provided in a separate event resource file. For example, to provide translated strings for Spanish, Spanish (Spain), and Spanish (Mexico), the following event resource files must exist:

```
eventLocale.es.xml
eventLocale.es_ES.xml
eventLocale.es_MX.xml
```

Each of these files will contain all of the event text strings for the respective language/country. The names of these files must be in the form:

```
eventLocale.ll_CC.xml
```

where:

- *ll* is a lowercase, two-letter ISO 639-1 language code (for example, `locale.es.xml` for Spanish). For a list of language codes, visit the following web site:

  http://www.loc.gov/standards/iso639-2/php/English_list.php

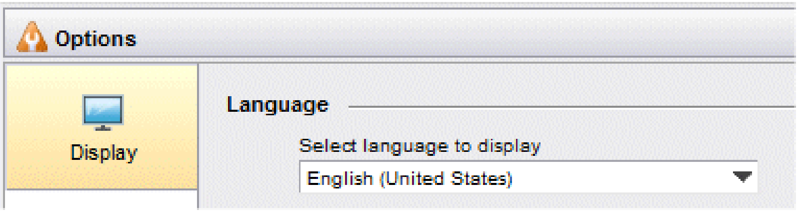
- *CC* is an uppercase, two-letter ISO 3166 country code. For a list of country codes, visit the following web site:

  http://www.iso.org/iso/country_codes/iso_3166_code_lists/country_names_and_code_elements.htm

Process of Displaying Text Strings

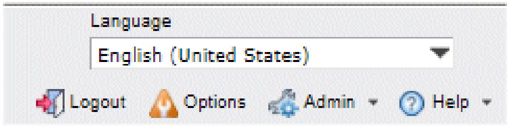
This topic lists the sequence of events for displaying text strings in Workspace.

1. The language is established at login using the selection specified in **Options**:



Select language to display: English (United States)

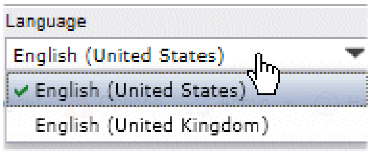
Or it can be changed dynamically using the **Language** field on the toolbar:



Language: English (United States)

The selections that are available from both of the **Language** field lists are determined by the **key** attributes in all of the general resource files (that is, the `locale.`*`ll`*`.xml` files).

Prior to any language packs being installed, or any custom localization being performed, the only **key** attributes are **key=”en_US”** in the default `locale.xml` file and the **key=”en_GB”** in the `locale.en_GB.xml` file. This results in “English (United States)” and “English (United Kingdom)” appearing in the **Language** fields:

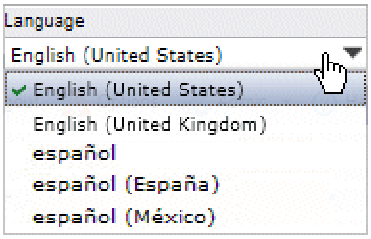


Language: English (United States)

If, however, you perform the steps provided here to localize your application, there will be additional **key** attributes in the resource files. For every **key** attribute, an additional language (possibly country specific) is shown in the **Language** fields. For example, if the following **key** attributes are found in the resource files:

- key=”en_US”
- key=”en_GB”
- key=”es”
- key=”es_EX”
- key=”es_MX”

... the **Language** fields will contain the following selections:



Language: English (United States)

The first time Workspace is started (that is, prior to determining what is specified in **Options**), the language that is specified in the browser is used.

2. For each text string that the application needs to display, it accesses the appropriate resource file based on the selected language.

   a. If the text string is displayed somewhere in the application other than in the event view list or Event Viewer, it gets the string from the resource file whose name includes the appropriate language code. The following illustrates this using Spanish and Spanish (Mexico) as an example:

      - If the selected language is just “Spanish”, it uses the strings under **key=”es”** in the `locale.es.xml` file. If the string it is looking for (based on a **jsxid** attribute) cannot be found in this file, it “falls back” to the default `locale.xml` file and uses the string from that file.
      - If the selected language is Spanish (Mexico), it uses the strings under **key=”es_MX”** in the `locale.es.xml` file. If the string it is looking for (based on a **jsxid** attribute) cannot be found under this key, it “falls back” to using the string under **key=”es”** in the `locale.es.xml` file. And if it cannot be found under **key=”es”** of that file, it again falls back to the default `locale.xml` file and uses the string from that file.

        The following illustrates this. If it is looking for the text string for **jsxid=”txtOpen”**, and the language selected is Spanish (Mexico), it will use the string under **key=”es_MX”**.

        ```
        <data jsxnamespace="propsbundle"
            <locale key="es">
              <!-- DEFAULT LANGUAGE RECORD ELEMENTS HERE -->
              ...
              <record jsxid="txtClose" jsxtext="Cierre"/>
              <record jsxid="txtOpen" jsxtext="Abierto"/>
              ...
            </locale>
            <locale key="es_ES">
              <record jsxid="txtOpen" jsxtext="override value for Spain"/>
            </locale>
            <locale key="es_MX">
              <record jsxid="txtOpen" jsxtext="override value for Mexico"/>
            </locale>
        </data>
        ```

        An exception to the process described above for “general” resource files is when “English (United States)” is selected. The default `locale.xml` file is used in this case—it does NOT have a **key=”en”** like all of the other general resource files.

   b. If the text string is in the event view list or the Event Viewer, it gets the string from the appropriately named event resource file. The following illustrates this using Spanish and Spanish (Mexico) as an example:

      - If the selected language is just “Spanish”, it uses the strings in the `eventLocale.es.xml` file. If the string it is looking for (based on an **id** attribute) cannot be found in this file, it “falls back” to the default `eventLocale.xml` file and uses the string from that file.
      - If the selected language is Spanish (Mexico), it uses the strings in the `eventLocale.es_MX.xml` file. If the string it is looking for (based on an **id** attribute) cannot be found in this file, it “falls back” to the `eventLocale.es.xml` file. And if it cannot be found in that file, it again falls back to the default `eventLocale.xml` file and uses the string from that file.

        Note that the naming convention for the English (United States) event resource file (`eventLocale.xml`) is an exception—this file does not include “en” in the name like the other event resource files. However, if you add country-specific English translations, event resource files with *ll_CC* in the name must exist, for example, `eventLocale.en_GB` for English (United Kingdom).

Creating New Resource Files

To add a new language, you must create both a “general” resource file that contains text strings used throughout the application, as well as a new event resource file. For every language, there must be one of each. (And if there are country-specific translations, there must be a separate event resource file for each of those.)

Creating New General Resource Files

“General” resource files are resource files that contain strings used throughout the application.

Procedure

1. Open the following file in your local development environment:

   ```
      ...\JSXAPPS\base\locale\locale.xml
   ```

2. Ensure that the language code for the resource file you are adding is listed in the **locales** attribute:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <data jsxnamespace="propsbundle"
         locales="de,es,es_ES,es_MX,fr,fr_FR,fr_CA,id,ja,ko,pl,pt,pt_BR,ru,th,tr,vi,
   zh,zh_CN,zh_HK,zh_TW,en_GB">
      <locale>
          .
          .
          .
   ```

   A number of the likely used language codes are included in the **locales** attribute by default. If the code for the language you are adding in not already there, add it.

   For a list of language codes, see:

   http://www.loc.gov/standards/iso639-2/php/English_list.php

   Also see Removing Unneeded Locale Keys for information about removing the locale keys for unused locales to improve performance.

3. Save the changes (if any) to the `locale.xml` file, but don’t close it yet.

4. Navigate to the following directory:

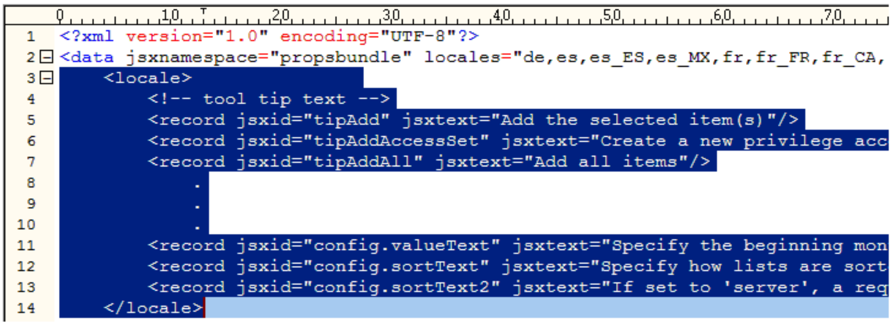
   ```
      ...\JSXAPPS\base\locale\
   ```

5. Determine if a resource file exists for the language you are adding (for instance, if you are adding Spanish, or a country-specific Spanish, you need a resource file with the name `locale.es.xml`)—various language resource files with no text strings in them are included by default. These can be used as a starting point for adding your own resource file.

   a) If the needed resource file exists, open it.

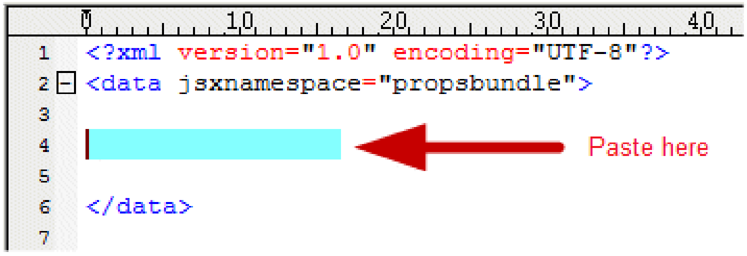
   b) If the needed resource file does not exist, copy one of the other ones that does exist (one of those without text strings yet), change the file name to include the appropriate language code (for example `locale.es.xml` for Spanish), then open it.

6. From the `locale.xml` file that you still have open from Step 1, copy all of the text string records, including the <locale> and </locale> elements:



```
<?xml version="1.0" encoding="UTF-8"?>
<data jsxnamespace="propsbundle" locales="de,es,es_ES,es_MX,fr,fr_FR,fr_CA,
    <locale>
        <!-- tool tip text -->
        <record jsxid="tipAdd" jsxtext="Add the selected item(s)"/>
        <record jsxid="tipAddAccessSet" jsxtext="Create a new privilege acc
        <record jsxid="tipAddAll" jsxtext="Add all items"/>
                .
                .
                .
        <record jsxid="config.valueText" jsxtext="Specify the beginning mon
        <record jsxid="config.sortText" jsxtext="Specify how lists are sort
        <record jsxid="config.sortText2" jsxtext="If set to 'server', a req
    </locale>
```

7. Paste the copied strings into the new resource file that you opened or created in step 5 (for example, `locale.es.xml`), inside of the existing <data> and </data> elements:



```
<?xml version="1.0" encoding="UTF-8"?>
<data jsxnamespace="propsbundle">


</data>
```

8. Inside of the <locale> element in the new resource file, add a **key** attribute and set its value to the language code for the language you are adding. For example, if you are adding Spanish, add **key="es"**:

```
<?xml version="1.0" encoding="UTF-8"?>
<data jsxnamespace="propsbundle">
<locale key="es">
    <!-- tool tip text -->
    <record jsxid="tipAdd" jsxtext="Add the selected item(s)"/>
    <record jsxid="tipAddAccessSet" jsxtext="Create a new privi
       .
       .
       .
```

9. Translate the value of every **jsxtext** attribute in the newly created resource file to language-specific values.

   Note that any record elements that are removed from the new language resource file will cause the application to "fall back" to the strings specified in the default English (United States) `locale.xml` file.

10. Optionally, localize the new language resource for specific countries. The purpose of localizing for specific countries is to provide a mechanism for overriding default language text values (translated in step 9) with text values that are specific for a country and that differ from the default.

    For each country-specific locale, create a <locale> element (within the root <data> element) and specify the language code and country code (ll_CC) as the value of the **key** attribute, then insert record elements into each new <locale> element that are to "override" default language records, with matching **jsxid** attribute values. For example:

```
<data jsxnamespace="propsbundle"
    <locale key="es">
      <!-- DEFAULT LANGUAGE RECORD ELEMENTS HERE -->
       .
       .
       .
<record jsxid="txtOpen" jsxtext="Abierto"/>
       .
       .
       .
```

```
</locale>
    <locale key="es_ES">
      <record jsxid="txtOpen" jsxtext="override value for Spain"/>
    </locale>
    <locale key="es_MX">
      <record jsxid="txtOpen" jsxtext="override value for Mexico"/>
    </locale>
</data>
```

11. Save the newly created resource file.

12. Close the new resource file and `locale.xml`.

Creating a New Event Resource File

You must have a separate event resource file for every language, as well as every country-specific language.

For instance, if you are adding a Spanish general resource file (`locale.es.xml`) and it includes **key** attributes for country-specific “es_ES” and “es_MX” translations, then you need the following event resource files:

- `eventLocale.es.xml`
- `eventLocale.es_ES.xml`
- `eventLocale.es_MX.xml`

The following illustrates creating the needed event resource files.

Procedure

1. Ensure the event resource file for the language exists (for example, `eventLocale.es.xml`).
   a) If the needed event resource file exists, open it.
   b) If the needed event resource file does not exist, copy one of the other ones that does exist (one of those without text strings yet), change the file name to include the appropriate language code (for example `eventLocale.es.xml` for Spanish), then open it.

2. Also open the default English (United States) event resource file, `eventLocale.xml`.

3. From the `eventLocale.xml` file, copy all of the text string records within the <data> </data> elements.

4. Paste those records to the `eventLocale.ll.xml` file you opened in step 1, between the <data> </data> elements.

5. Translate all of the **text** attribute string values for the appropriate language.

   Note - Any record elements that are removed from the new event resource file will cause the application to “fall back” to the strings specified in the default English (United States) `eventLocale.xml` file.

6. Save and close both the new event resource file and `eventLocales.xml`.

7. If you also adding country-specific translations, you must also create country-specific event resource files (as in this example for Spanish (Spain) and Spanish (Mexico)). To do this:
   a) Make a copy of the new event resource file you saved and closed in step 6.
   b) Rename the copied file to include the country code, for example, `eventLocale.es_ES` (for information about country codes, see Country-Specific Translations).
   c) Open the country-specific event resource file and modify the text strings in the **text** attributes for the country-specific translations. Note that any record elements that are removed from the country-specific event resource file will cause the application to “fall back” to the strings specified in the language-specific event resource file (for example, `eventLocale.es.xml`).
   d) Save and close the country-specific event resource file.

e) Repeat steps 7a-7d for each country-specific language that is being added.

Modify or Create a General Interface System Locale File

Some of the text that is displayed in Workspace application originates from the TIBCO General Interface system locale files.

By default, General Interface is already localized for many languages and countries, however, some of the text displayed in Workspace is only defined in the default General Interface system locale file (English), and must be inserted and translated into other locale files as required.

For a list of the languages supported in the current version of General Interface, see the *TIBCO General Interface Developer Guide*.

The General Interface language resource files are located as follows:

```
...\JSX\locale\
```

As with Workspace, the default General Interface language resource file is `locale.xml` (English).

Deploying Language Resource Files

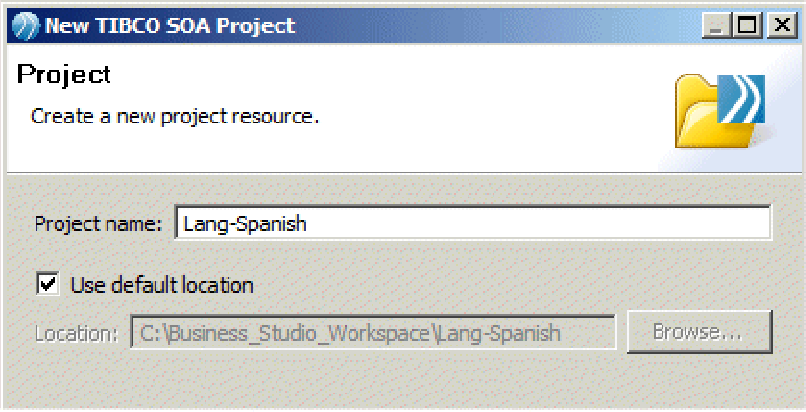
After resource files have been created (that is, translations completed, **key** attribute values set, etc.), those files must be deployed to the node before they can be used in the application.

The way in which you deploy the resource files depends the type and current deployment status of the application. For more information see Deploying Language Resource Files.

The following describes the procedure for deploying resource files when an extension point had been previously defined:

Procedure

1. In TIBCO Business Studio, create a new SOA project: **File** > **New** > **TIBCO SOA Resources**
2. On the Select a wizard dialog, select **TIBCO SOA Project** under the `TIBCO SOA Platform` folder, then click **Next**.
3. On the Project dialog, enter a project name (this example uses the name “Lang-Spanish”), then click **Next**.



New TIBCO SOA Project

Project

Create a new project resource.

Project name: Lang-Spanish

☑ Use default location

Location: C:\Business_Studio_Workspace\Lang-Spanish

4. Accept the defaults on the Asset Type Selection dialog—click **Next**.
5. On the Composite Project dialog, ensure “Empty SOA Project” is selected, then click **Finish**.

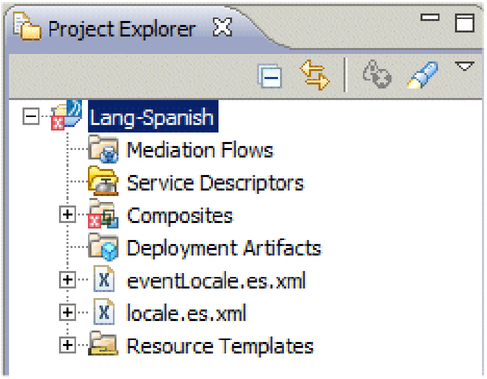
Throughout these steps, you should save your project periodically.

6. Copy the resource files to the root folder of the SOA project, as follows:

a) Copy the translated locale and eventLocale files from your file system (this example will use `locale.es.xml` and `eventLocale.es.xml`) to the clipboard.

> These steps illustrate adding resource files for a single language (Spanish). To add more than one language, modify the steps according to include all of the resource files needed.

b) In the Project Explorer in Business Studio, right click on the root of the project, then select **Paste** from the context menu. The files are added as shown below:

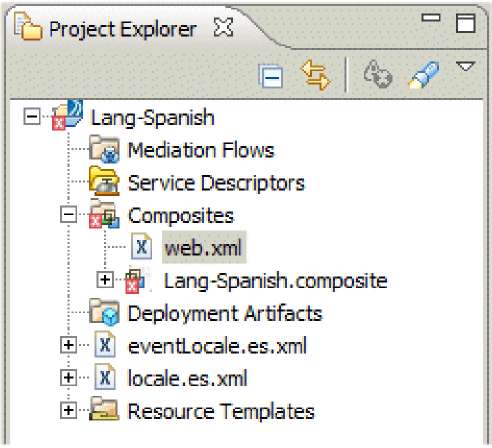


7. Add a `web.xml` deployment descriptor to the `Composites` folder in the project, as follows:

a) Right click the `Composites` folder, then select **New** > **File** .

b) Name the file `web.xml`, then **Finish**.

The file is created as shown below:



8. Right click the `web.xml` file in the Project Explorer, then select **Open With** > **Text Editor** .

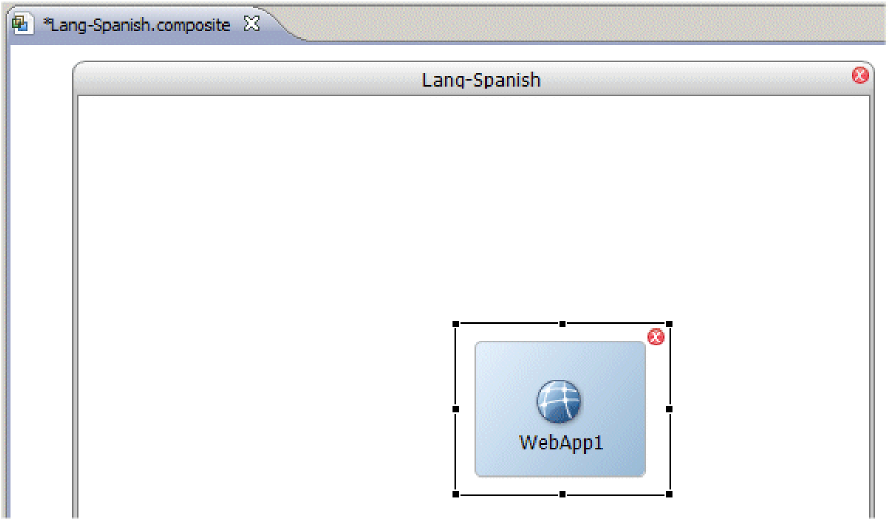
9. Paste the following text into the opened `web.xml` file:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><web-app xmlns="http://
java.sun.com/xml/ns/j2ee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
version="2.4" xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://
java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"></web-app>
```

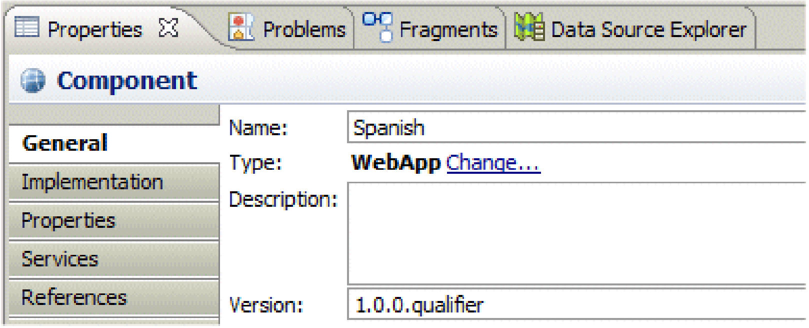
10. Save and close the `web.xml` file.

11. Add a WebApp component to the composite by selecting **WebApp** from the **Component Types** palette, then clicking in the composite.

The composite should appear as follows:



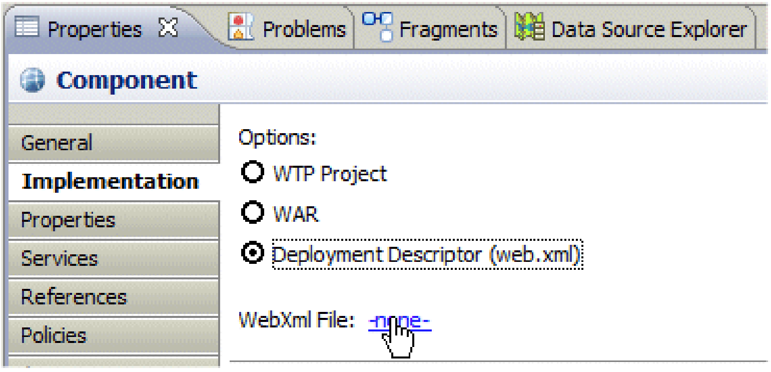
12. Display the **Properties** view for the component, select the **General** tab, then enter a name for the component in the **Name** field—this example will name the component "Spanish":



Name: Spanish
Type: **WebApp** Change...
Description:
Version: 1.0.0.qualifier

Also notice the version number for the component on the **General** tab. Later versions of the component take precedence. Therefore, if you deploy resource files to the same extension point with different component versions (for example, 1.0.0 and 1.2.0), the one with the higher version takes precedence.

13. On the **Implementation** tab, select the **Deployment Descriptor (web.xml)** option, then click the **-none-** link to the right of "WebXml File":



Options:
○ WTP Project
○ WAR
● Deployment Descriptor (web.xml)

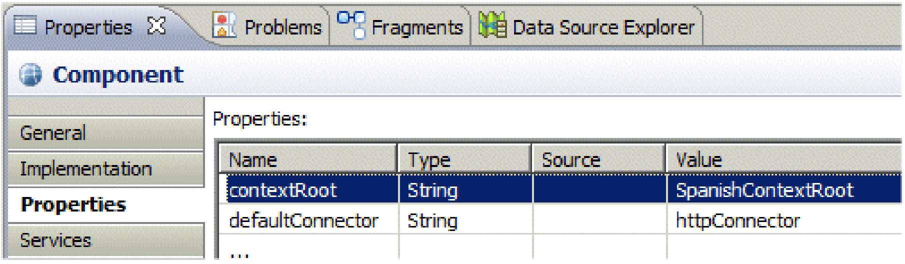
WebXml File: -none-

14. On the Select web.xml dialog, navigate to and select the `web.xml` file under the **Composites** folder in your SOA project, then click **OK**.

15. On the **Properties** tab of the **Properties** view, enter a value for the **contextRoot** and **defaultConnector**.

    The contextRoot can be any name.

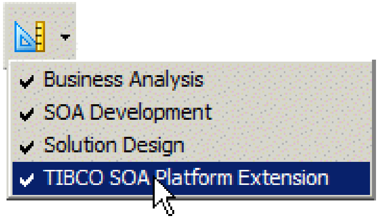
    The defaultConnector must be the same HTTP connector used by the BPM runtime (which can be seen in TIBCO Administrator—the default is "httpConnector"):



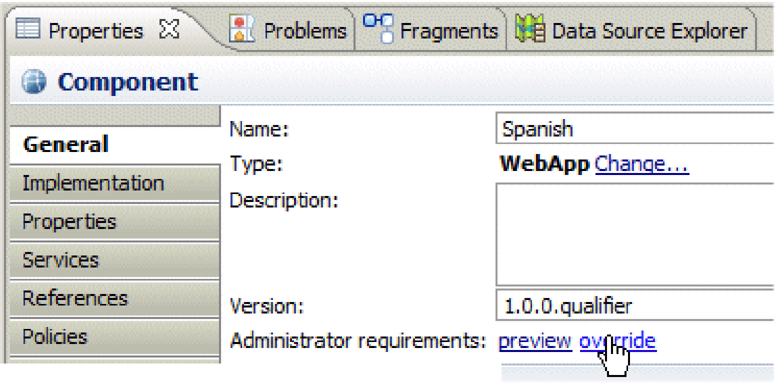
Properties:

Name	Type	Source	Value
contextRoot	String		SpanishContextRoot
defaultConnector	String		httpConnector
...			

16. Ensure that "TIBCO SOA Platform Extension Capability" is enabled:



17. On the **General** tab, click the **override** link:



Name: Spanish

Type: **WebApp**

Description:

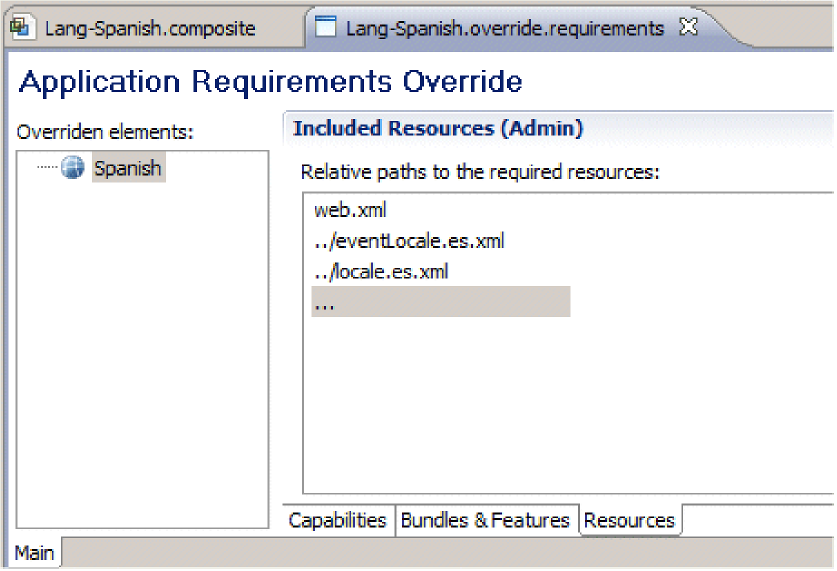
Version: 1.0.0.qualifier

Administrator requirements: preview override

    The *projectName*.**override.requirements** view is displayed.

18. Click the **Resources** tab, which is displayed on the bottom of the *projectName*.**override.requirements** view.

19. Use the Browse button to add the resource files:



Application Requirements Override

Overriden elements:

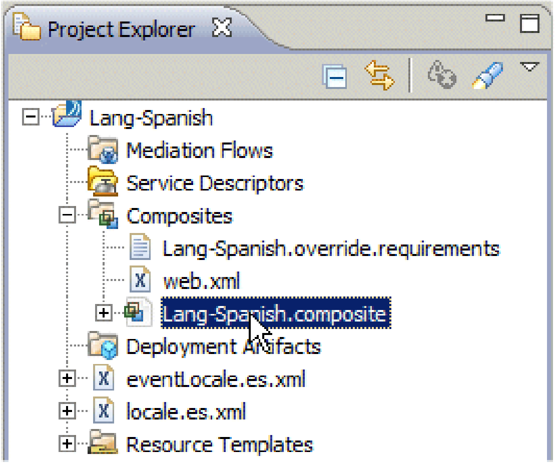
Spanish

Included Resources (Admin)

Relative paths to the required resources:

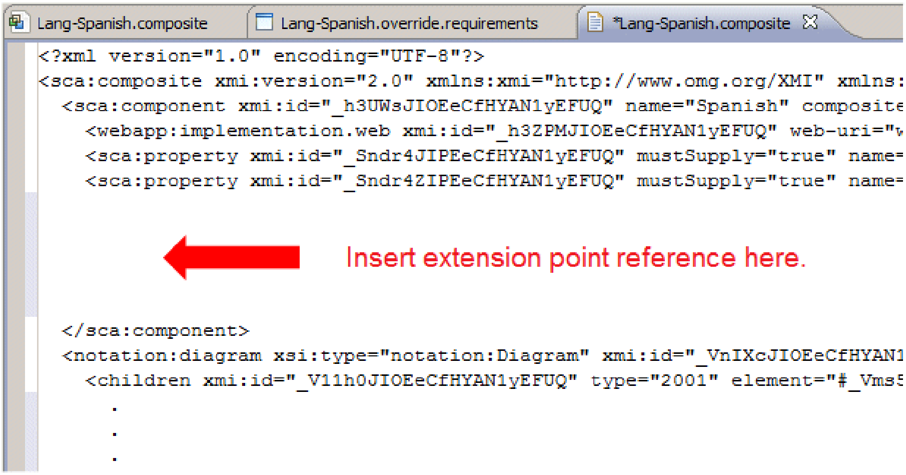
web.xml
../eventLocale.es.xml
../locale.es.xml
...

20. In the **Project Explorer**, right click the composite, then select **Open With** > **Text Editor** from the context menu:



Lang-Spanish
Mediation Flows
Service Descriptors
Composites
Lang-Spanish.override.requirements
web.xml
Lang-Spanish.composite
Deployment Artifacts
eventLocale.es.xml
locale.es.xml
Resource Templates

21. Insert an extension point reference into the composite file, inside of the <sca:Componentelement>. This specifies that resource files are being added to an application that was deployed with an extension point. The extension point reference names the extension point that is being referenced, as well as the location of the resource files in the SOA project, and the location to which they need to be mapped in the deployed application:



```
<?xml version="1.0" encoding="UTF-8"?>
<sca:composite xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:
  <sca:component xmi:id="_h3UWsJIOEeCfHYAN1yEFUQ" name="Spanish" composite
    <webapp:implementation.web xmi:id="_h3ZPMJIOEeCfHYAN1yEFUQ" web-uri="w
    <sca:property xmi:id="_Sndr4JIPEeCfHYAN1yEFUQ" mustSupply="true" name=
    <sca:property xmi:id="_Sndr4ZIPEeCfHYAN1yEFUQ" mustSupply="true" name=



  </sca:component>
  <notation:diagram xsi:type="notation:Diagram" xmi:id="_VnIXcJIOEeCfHYAN1
    <children xmi:id="_V11h0JIOEeCfHYAN1yEFUQ" type="2001" element="#_Vms5
      .
      .
      .
```

Insert extension point reference here.

Use the following text for the extension point reference:

```
<scaext:extension xmi:id="_FYNUQEntEeC1W-tBHP1Z0A"
name="LanguagePackInstallWebApp" requiredVersion="1.0.0"
extensionPoint="com.tibco.n2.workspace.language-pack.extension-point">
<webapp:web-app-update xmi:id="_FYNUQUntEeC1W-tBHP1Z0A"> <static-resource
xmi:id="_FYNUQ0ntEeC1W-tBHP1Z0A" location="/projName/eventLocale.ll.xml" path="/
JSXAPPS/base/locale/eventLocale.ll.xml"/> <static-resource
xmi:id="_FYNUREntEeC1W-tBHP1Z0A" location="/projName/locale.ll.xml" path="/
JSXAPPS/base/locale/locale.ll.xml"/> </webapp:web-app-update></scaext:extension>
```

Change the values in the following attribute, as described:

- **extensionPoint** - This is the name of the extension point. If you are updating resource files in the Workspace application, this value must be the following (as this is the extension point name that is hard-coded into the Workspace application):

  ```
  com.tibco.n2.workspace.language-pack.extension-point
  ```

  If you are updating resource files in a custom WCC application, this must be the name of the extension point that was defined when the application was deployed. For more information, see Deploying an Application After Customizing.

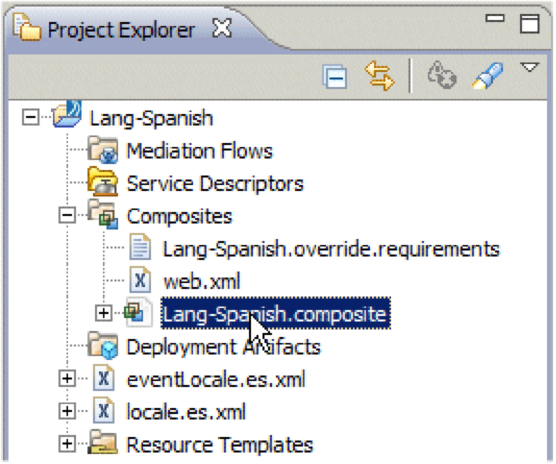
- **location** - This specifies the location of the resource files in the SOA project. Specify the name of your SOA project in the *projName* part of the value, and the appropriate country code in the *ll* part of the value.

- **path** - This specifies the relative path that the resource files are to be mapped in the deployed application. Specify appropriate country code in the *ll* part of the value.

22. Save the project.

23. Create a DAA by doing the following:

    a) In the Project Explorer, right click the composite, then select **Create DAA** from the context menu:

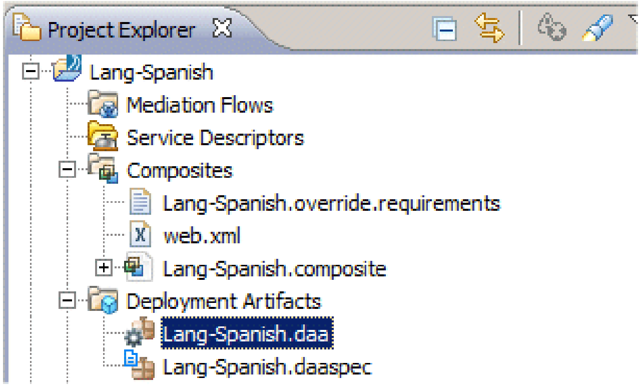


b) On the Select Archive Location dialog, select **Deployment Artifacts** in your SOA project, then click **Next**.

c) On the Select Distribution dialog, ensure the **Do not use a distribution file** option is selected, then click **Next**.

d) On the DAA Specification dialog, ensure the **Save DAA Spec** option is selected, then click **Finish**.

The DAA should now appear in the `Deployment Artifacts` folder:

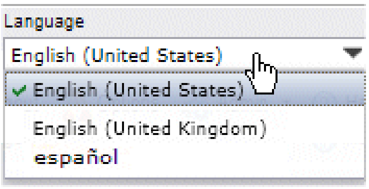


24. Using TIBCO Administrator, deploy the DAA file to the appropriate node.

For information about deploying applications using TIBCO Administrator, see the *TIBCO Administration* guide. (For a tutorial of deploying applications using TIBCO Administrator, see the *How to Implement and Deploy the WelcomeUsers Application*.)

Note that you may need to clear your browser cache to see the change in the application.

The deployed language should now be available in the application:



Sample Applications

A number of sample applications are provided with TIBCO Workspace.

The samples are located in the following directory:

`StudioHome\wcc\version\samples\`

where:

- *StudioHome* is the directory in which TIBCO Business Studio was installed.
- *version* is the version number of Workspace that was installed with TIBCO Business Studio.

The following summarizes the sample applications that are provided with Workspace. Note, however, with the exception of a couple, the samples are not described in detail here, but rather they are described in detail in other parts of the Workspace documentation, as they apply to those areas.

The sample applications provided with TIBCO Workspace are:

- **PageBus Payload Applications** - This sample consists of two applications. One (**pageBusPayloadHelperApp**) is used to create and generate the PageBus event payload that represents a work item or process instance list. The second (**pageBusPayloadPublishApp**) is used to publish the generated event so that the work item or process instance list is rendered in the desired location.

  For more information, see the “Non-WCC Code Publishing Events” section in the “Using WCC Components in Mash-Ups*”* chapter of the *TIBCO Workspace Components Developer Guide*.

- **Login Managed Hub** - This sample application (**wccLoginManagedHub**) illustrates how to do the following when using iframes and the PageBus Managed Hub:

  – Initialize a login for a WCC application running in an iframe, by utilizing a login from a previous login session.

  – Display a component (e.g., the Organization Browser) in an iframe.

  – Publish an **externalShowEvents** event, which allows an external application to add a temporary event view to the event view list.

    For more information, see the “Using WCC Components in Mash-Ups” chapter in the *TIBCO Workspace Components Developer Guide*.

- **PassThru Managed Hub** - This sample application (**wccPassThruManagedHub**) illustrates how to do the following when using iframes and the PageBus Managed Hub:

  – Initialize a login for a WCC application running in an iframe, by utilizing a login from a previous login session.

  – Publish a **passThruEvent** event, which passes through the TIBCO PageBus Managed Hub, where the IframeClient subscribes to the event and displays either a work item list or process instance list, depending on the payload in the published event.

    For more information, see the “Using WCC Components in Mash-Ups” chapter in the *TIBCO Workspace Components Developer Guide*.

- **Custom Menus** - This sample application (**customMenusSample**) illustrates how to add custom menus and/or toolbar buttons to an application.

  For more information about this sample application, see Adding Custom Menus and Toolbar Buttons.

- **Application Methods** - This sample application (**wccApplicationMethods**) provides an example of the available Tools interface methods in the Application class. It provides a UI that allows you to enter the needed information (e.g., work item ID, version number, etc.), then execute a method.

This sample application is described in the “Tools Interface Methods” section in the “WCC Public Methods” chapter of the *TIBCO Workspace Components Developer Guide*.

- **ApiSample** - This sample application (**wccApiSample**) demonstrates the use of server requests, which provide a means of accessing certain types of BPM data that are normally only exposed through visual WCC components. This sample application provides a UI that allows you to execute each of the available server requests, then receive a response from the request.

  This sample application is described in the “Server Requests” section in the “WCC Public Methods” chapter of the *TIBCO Workspace Components Developer Guide*.

- **View Distributor** - This sample application (**wccViewDistributor**) is used to distribute views that are currently in your work view, process view, or event view list to either individual users or to members of a specified group or position.

  For more information about this sample application, see View Distributor Sample Application.

- **User Access Sample** - This sample application (wccAccessSample) provides an example of:

  – Adding custom menus and toolbar buttons to a WCC application

  – Adding subordinate items to the custom menus and toolbar buttons.

  – Limiting access to specific items on custom menus and toolbar buttons.

  – Controlling access to various parts of a WCC application (including the Workspace application).

  – Starting business services through code.

  For more information about this sample application, see User Access Sample Application.

- **Callouts Sample** - This sample application illustrates the use of the available callout interface methods. These methods allow you to *callout*, or *inject*, custom specifications in WCC applications, including the Workspace application. The callout interface methods can be used to do the following:

  – Define custom filters, sorts, filter attributes, sort attributes, and display columns on various lists.

  – Restrict the list of processes that are available to a user when creating a process view or starting a process instance.

  – Restrict the list of business services or business service categories that are available to a user when starting a business service.

  – Specify characteristics of lists displayed in the application, for example, font size, font color, alignment, etc.

  This sample application is described in the “Callout Interface Methods” section in the “WCC Public Methods” chapter of the *TIBCO Workspace Components Developer Guide*.

- **Login Session** - This sample application, which works in conjunction with the openworkitem Application, illustrates ways to establish and maintain a login session so that users do not need to be authenticated every time the **openworkitem** application is invoked.

  For more information about this sample application, see Login Session Sample Application.

- **Custom Interfaces** - Samples are provided that illustrate each of the available custom interface types: Simple iFrame, Simple Managed Hub iFrame, Simple Prototype, and Scripted Interface.

  For informaiton about these samples, see Sample Custom Interfaces.

View Distributor Sample Application

The View Distributor sample application is used to distribute views that are currently in your work view, process view, or event view list to either individual users or to groups or posiitons.

The View Distributor sample application has been deprecated. *System views* should now be used to define views for users. For information about system views, see the "Using Views" chapter in the *TIBCO Workspace User's Guide.*

This sample application can distribute views to either:

- **Individual users** - The users to which you distribute a view will receive a copy of the view in their work view, process view, or event view list, depending on the type of view you are distributing.
- **Groups or positions** - Distributing a view to a group or position causes all users that are currently members of the group or position to receive a copy of the view in their work view, process view, or event view list, depending on the type of view you are distributing.

  Note that users that are added to a group or position at a later time are *not* automatically given copies of views that had been previously distributed to the group or position. A new distribution would need to be done for those users to receive a copy of the view.

Views that are going to be distributed with this sample application must first be created in either the Workspace application or in another custom WCC application—you cannot create views with this sample application—it is only used to distribute them. (Views are specific to Workspace/WCC applications, and do not apply to Openspace.)

Distributing a view gives the recipient *a copy* of the view that is currently defined in the logged-in user's view list. The copy of the view contains all filters, sort specifications, custom columns, etc., that are part of the view definition. The recipient of the view is then free to modify that view to the extent that their user access controls allow.

If you create or modify a view in Workspace or another custom WCC application, and want to distribute that view, you must first log out of the application, which causes the view to become persisted. You can then distribute the view using the View Distributor application.

To use this application, the Workspace Client Components (WCC) must have been installed when TIBCO Business Studio was installed.

Also note that if your intent is to distribute an event view to all users, you may want to consider creating the event view as a template, then specify it as a default event view. This prevents you from having to continue to distribute the event view in the future as new users are added, or users are added to groups or positions. For information, see defaultEventView.

User Access Controls

User access controls are used by the View Distributor application to control access to the distributed views.

The user access controles used are:

- Administration > Configuration - This gives you access to the View Distributor application itself. This user access control can be used to control who can run the application.
- **WorkView** - This gives you access to the list of work views that can be distributed.
- **ProcessView** - This gives you access to the list of process views that can be distributed.
- **EventView** - This gives you access to the list of event views that can be distributed.
- **OrganizationBrowser** - This gives you access to the list of users, groups, and positions to which a view can be distributed.

Note that the View Distributor application does not contain its own `userAccess.xml` file to control user access. You must use the `userAccess.xml` file associated with the application that was used to create the views that you want to distribute.

If you are running the application from your local development environment, configure user access using the following file:

```
StudioHome\wcc\version\JSXAPPS\WCCProjectName\userAccess.xml
```

where:

- *StudioHome* is the directory in which TIBCO Business Studio was installed.
- *version* is the version number of Workspace that was installed with TIBCO Business Studio.
- *WCCProjectName* is the name of the General Interface Builder project that contains your custom application. If you are working with the Workspace application, this is “workspace”.

If you are going to deploy the View Distributor application to a runtime node, then run it from there, configure user access using the Configuration Administrator from your application on the node (either the Workspace application, or a custom WCC application).

More information about running the View Distributor application from different locations is provided in Starting the View Distributor Application.

For information about configuring user access controls, see the “Configuring User Access” chapter in the *TIBCO Workspace Configuration and Customization* guide.

Setting Up the View Distributor Application

You must set up the View Distributor application prior to using it.

Procedure

1. Copy the following directory...

   ```
   StudioHome\wcc\version\Samples\wccViewDistributor\JSXAPPS\wccViewDistributor\
   ```

   ... to the following directory:

   ```
   StudioHome\wcc\version\JSXAPPS\
   ```

   where:

   - *StudioHome* is the directory in which TIBCO Business Studio was installed.
   - *version* is the version number of Workspace that was installed with TIBCO Business Studio.

2. Copy the View Distributor application’s launch fragment from here...

   ```
   StudioHome\wcc\version\Samples\wccViewDistributor\wccViewDistributor.html
   ```

   ... to here:

   ```
   StudioHome\wcc\version\
   ```

   This is the file that will be used to start the application.

3. Copy the View Distributor application’s WAR-file creation utility from here...

   ```
   StudioHome\wcc\version\Samples\wccViewDistributor
   \wccViewDistributor.create.war.cmd
   ```

   ... to here:

   ```
   StudioHome\wcc\version\
   ```

This file is a command utility that is used to create the WAR file to deploy the utility to a runtime node. This file will be used only if you deploy the View Distributor application to a runtime node. For more information, see Starting the View Distributor Application.

Configuring the baseUrl

The *baseUrl* needs to be configured if you are going to run the View Distributor application locally on your development machine. The baseUrl is the URL that points to the Action Processor to which the application will connect at runtime.

Note that when running on an ActiveMatrix BPM node, the application and the Action Processor must be running on the same machine. Therefore, the **baseUrl** attribute for the **ActionProcessors** record in the `config.xml` file is set to an empty string by default:

```
<record jsxid="ActionProcessors" type="workspace">
    <ActionProcessor
       weighting="100"
       baseUrl="">
    </ActionProcessor>
</record>
```

This causes the URL of the Action Processor to be determined at runtime based on the URL entered in the browser to start the application. So if you are going to deploy the View Distributor application to a runtime node and run it by pointing a browser to the application's URL, leave the value of the **baseUrl** attribute an empty string.

In a development/testing environment, however, you will likely be running the application on a local machine. Therefore, you will need to specify the URL to the Action Processor in the **baseUrl** attribute. To specify the baseUrl:

Procedure

1. Open the following file:

   `StudioHome``\wcc\`*`version`*`\JSXAPPS\wccViewDistributor\config.xml`

   where:

   - *StudioHome* is the directory in which TIBCO Business Studio was installed.
   - *version* is the version number of WCC that was installed with TIBCO Business Studio.

2. Locate the **ActionProcessors** record in the `config.xml` file.
3. Set the **baseUrl** attribute to the URL of the Action Processor. The string in the **baseUrl** attribute must be in the form:

   `http://`*`Host`*`:`*`Port`*`/bpm/actionprocessor/actionprocessor.servlet`

   where:

   - *Host* is the name or IP address of the machine hosting the Action Processor.
   - *Port* is the port number used by the ActiveMatrix WebApp Implementation Type to communicate with web applications.

     For example:

     ```
     <record jsxid="ActionProcessors" type="Workspace">
        <ActionProcessor
           weighting="100"
           baseUrl="http://Austin:8080/bpm/actionprocessor/
     actionprocessor.servlet">
     </ActionProcessor>
     ```

     Note that the **weighting** attribute is not used at this time. Also note that you cannot run locally if the transport protocol for the Action Processor is https; it must be http to run locally.

4. Save and close the `config.xml` file.

Namespace Configuration

By default, the View Distributor application is configured to be used with the Workspace application. That is, by default it can be used to distribute views created by users running the Workspace application.

If you are going to use the View Distributor application to distribute views created while running a custom WCC application other than Workspace, you will need to modify the following two files:

- The View Distributor application's launch fragment - A namespace in this file identifies the application from which the View Distributor application will obtain the views to distribute. It is set to "workspace" by default, and needs to be modified if the views you are distributing were created in a custom WCC application.
- The View Distributor application's `wccConfig.xml` file - The "appModelName" specified in this file is used to publish events to the PageBus. It is also set to "workspace" by default and needs to be modified if the views you are distributing were created in a custom WCC application.

Modifying the View Distributor Application's Launch Fragment

Procedure

1. Open the following file with an editor:

   `*StudioHome*\wcc\*version*\wccViewDistributor.html`

   where:

   - *StudioHome* is the directory in which TIBCO Business Studio was installed.
   - *version* is the version number of WCC that was installed with TIBCO Business Studio.
2. Locate the **jsxappns** attribute in the **Script** element (near the bottom of the file):

```
<script type="text/javascript" src="JSX/js/JSX30.js"
   jsxappns="workspace" jsxapppath="JSXAPPS/wccViewDistributor"
   wccapppath="JSXAPPS/wccViewDistributor" wccloadorder="1" >
</script>
```

3. Modify the value in the **jsxappns** attribute to fit the name of your custom WCC application. For example, if your custom application's name is "accounts", it should appear as follows:

```
<script type="text/javascript" src="JSX/js/JSX30.js"
   jsxappns="accounts" jsxapppath="JSXAPPS/wccViewDistributor"
   wccapppath="JSXAPPS/wccViewDistributor" wccloadorder="1" >
</script>
```

4. Save and close the `wccViewDistributor.html` file.

Modifying the View Distributor Application's wccConfig.xml

Procedure

1. Open the following file with an editor:

   `*StudioHome*\wcc\*version*\JSXAPPS\wccViewDistribtor\wccConfig.xml`

   where:

   - *StudioHome* is the directory in which TIBCO Business Studio was installed.
   - *version* is the version number of WCC that was installed with TIBCO Business Studio.

2. Locate the **name** attribute in the **appModelName** element:

```
<record jsxid="wcc" type="wcc">
<appClass class="com.tibco.wcc.wccViewDistributor.Application"/>
   <appRoot name=""/>
   <mainPrototype path="JSXAPPS/wccViewDistributor/application/prototypes/
appMain.xml"/>
   <appModelName name="workspace" />
   <resources>
      <jss copyFromNamespace="first" alwaysLoad="false" />
      <config copyFromNamespace="first" />
   </resources>
</record>
```

3. Modify the value in the **name** attribute to fit the name of your custom WCC application. For example, if your custom application's name is "accounts", it should appear as follows:

```
<record jsxid="wcc" type="wcc">
<appClass class="com.tibco.wcc.wccViewDistributor.Application"/>
   <appRoot name=""/>
   <mainPrototype path="JSXAPPS/wccViewDistributor/application/prototypes/
appMain.xml"/>
      <appModelName name="accounts" />
   <resources>
      <jss copyFromNamespace="first" alwaysLoad="false" />
      <config copyFromNamespace="first" />
   </resources>
</record>
```

4. Save and close the `wccConfig.xml` file.

Starting the View Distributor Application

The View Distributor application ican be started locally on your development machine, or it can be started on the runtime node.

- **Locally from your development machine** - You can run the application locally to test it, or possibly because you don't want to deploy it to a runtime machine and have it available and accessible by other users.

  To run the application locally, ensure that you first specify the **baseUrl** attribute as described in Configuring the baseUrl. The application can then be started by double clicking on the application's launch fragment, which is located as follows:

  StudioHome\wcc*version*\wccViewDistributor.html

  Depending on the type and version of browser you use, running from the local file system may violate rules imposed by the browser for *cross-domain scripting*. For more information, see Cross-Domain Scripting .

- **From a runtime node** - This requires that you deploy the application to a runtime node; it can then be run by pointing a browser at the application's launch fragment.

  Deploying to a runtime node involves first creating a WAR file using the `workViewDistributor.create.war.cmd` file. Using the WAR file, you can then create an application and DAA using TIBCO Business Studio; the DAA is then uploaded using TIBCO ActiveMatrix Administrator, and deployed to the runtime node. For detailed instructions on how to do this, see Deploying an Application After Customizing.

  Once the application is deployed to a runtime node, you can run it by pointing a browser at the application's launch fragment, as follows:

  http://*Host*:*Port*/*contextRoot*/wccViewDistributor.html

  where:

  – *Host* is the name or IP address of the machine hosting the BPM runtime.

- *Port* is the port number used by the ActiveMatrix WebApp Implementation Type to communicate with web applications.
- *contextRoot* is the root of the deployed application. This is the value that was entered in the **contextRoot** field when deploying the application—this is described in the deployment instructions in the *TIBCO Workspace Configuration and Customization* guide.

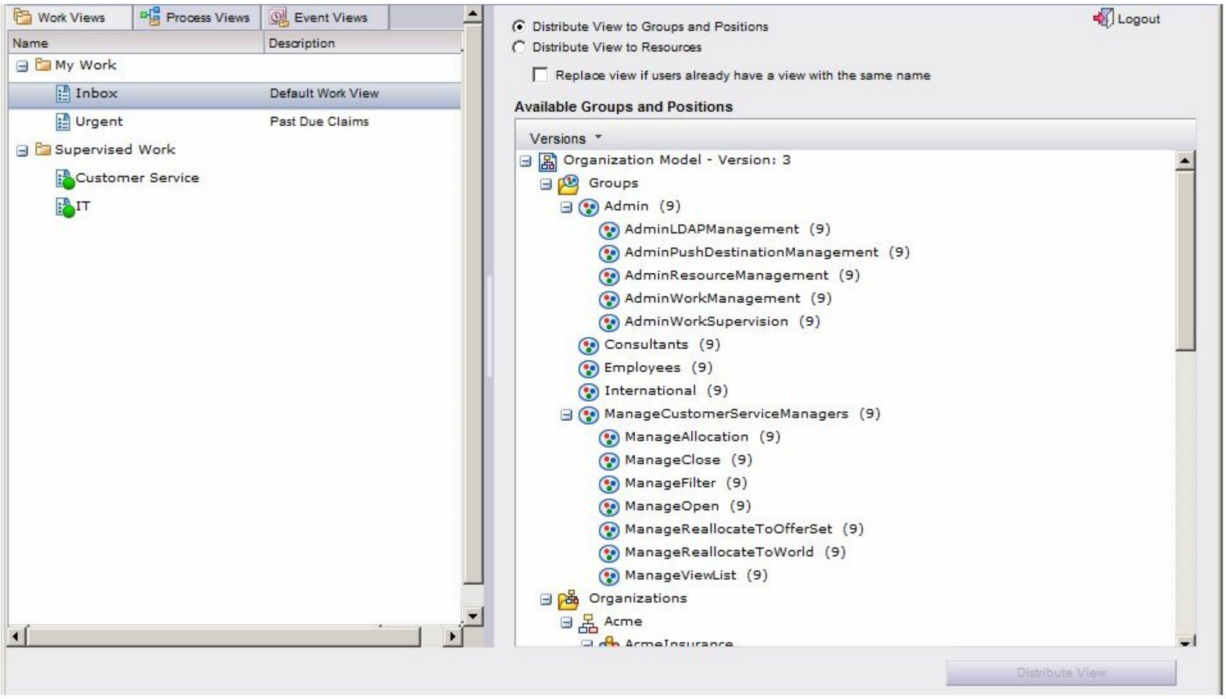
  For example:

  http://*Liberty*:*8080*/*wccViewDistributor*/wccViewDistributor.html

Using the View Distributor Application

When you start the View Distributor application, the Login dialog is displayed. You must enter a valid user name and password, then click the **Login** button.

After successfully logging in, a dialog similar to the following displays:



● Distribute View to Groups and Positions
○ Distribute View to Resources
□ Replace view if users already have a view with the same name

This dialog is divided into the following three panes:

- Left Pane - This pane shows you the views to which you (the logged-in user) have access.

  This pane can contain up to three tabs: **Work Views**, **Process Views**, and **Event Views**. Each of these tabs displays its respective views, allowing you to distribute the desired type of view.

  Note that one or more of these tabs may not be displayed if you do not have access to that type of view, that is, you do not have the user access control for that view type. For more information, see User Access Controls.
- Upper Right Pane - This pane contains options used to specify whether you are distributing views to individual users (resources), or to members of a group or position. Modifying these options causes the view in the lower-right pane to change.

  This pane also contains a check box that allows you to authorize the replacement of a view with the same name as the one you are distributing.
- Lower Right Pane - This pane lists either groups and positions (if the **Distribute View to Groups and Positions** option is selected in the upper-right pane), or individual users (if the **Distribute View to Resources** option is selected in the upper-right pane).

Distributing a View

Procedure

1. In the left pane, select the tab for the type of view you want to distribute.
2. On the selected view tab, select the view you want to distribute (only a single view can be distributed at one time).

   > If you distribute a supervised work view to a user who is not authorized to view it (that is, the user does not have the needed system actions), the view will appear in the work view list of the user to whom you distributed it, but a runtime error is displayed when the user clicks on the work view.

3. In the upper-right pane, choose how to distribute the view, as follows:

   - **Distribute View to Groups and Positions** - Select this option if you want to distribute the view to *all members* of one or more groups and positions.

     Selecting this option causes the **Available Groups and Positions** list to be displayed in the lower-right pane so that you can select the groups and/or positions whose members will receive the selected view.

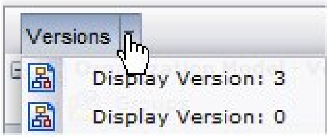
   - **Distribute View to Resources** - Select this option if you want to distribute the view to one or more individual users (resources).

     Selecting this option causes the **Available Resources** and **Selected Resources** lists to be displayed in the lower-right pane so that you can select the users who will receive the selected view.

4. Choose to whom the view is to be distributed, as follows:

   If you selected the **Distribute View to Groups and Positions** option:

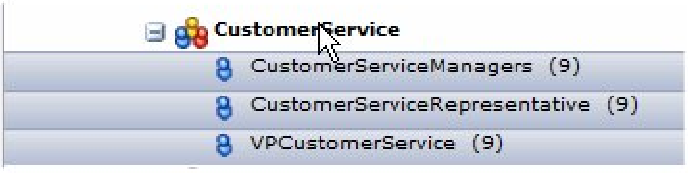
   a) Ensure that the correct version of the organization model is displayed in the lower-right pane. The most recent version is displayed by default. You can select a different version using the **Versions** drop-down list:

   b) In the lower-right pane, select the groups, positions, and/or organization units whose members you want the view distributed.

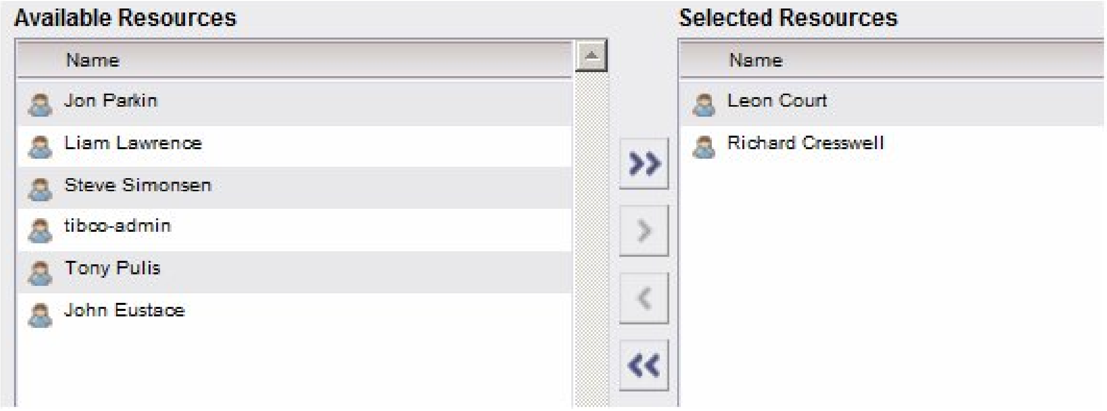
   You can select multiple groups and positions by pressing and holding the **Ctrl** key while selecting individual groups or positions. You can also select a group of items by clicking on the first one, then pressing and holding the **Shift** key while clicking on the last item in the group.

   Also note that if you select a group with subordinate groups, all of the subordinate groups are automatically selected. Likewise, if you select an organizational unit, any subordinate organizational units, as well as all subordinate positions, are automatically selected.

   
   CustomerService
   CustomerServiceManagers (9)
   CustomerServiceRepresentative (9)
   VPCustomerService (9)

   If you selected the **Distribute View to Resources** option:

a) In the lower pane, select one or more of the users in the **Available Resources** list to whom you want the view distributed, then click the **>** button to move them to the **Selected Resources** list (you can use the **Ctrl** and **Shift** keys to select multiple users as described above).



You can also move all users from the **Available Resources** list to the **Selected Resources** list by clicking on the **>>** button.

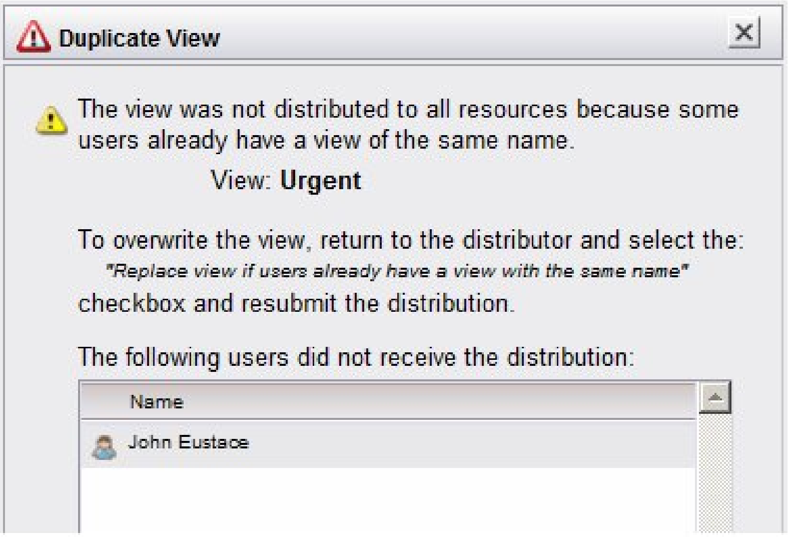
If you have moved users to the **Selected Resources** list, but decide not to distribute to them, they can be moved back to the **Available Resources** list using the **<** and **<<** buttons.

5. Click the **Distribute View** button. One of two dialogs is displayed:

- **Duplicate View** dialog - see Step 5a.
- **Distribution View** dialog - see Step 5b.

If you are distributing a view to a very large number of users, the process can take some time to complete. If the distribution is taking a long time, the browser may display a dialog that indicates there is a problem with a script running on the page. For information about preventing these dialogs from being displayed, see Browser Script Warnings .

a) The Duplicate View dialog is displayed only if the application determined that one or more of the users to whom you are distributing a view already has a view of that name:



Duplicate View

The view was not distributed to all resources because some users already have a view of the same name.

View: **Urgent**

To overwrite the view, return to the distributor and select the:
"Replace view if users already have a view with the same name"
checkbox and resubmit the distribution.

The following users did not receive the distribution:

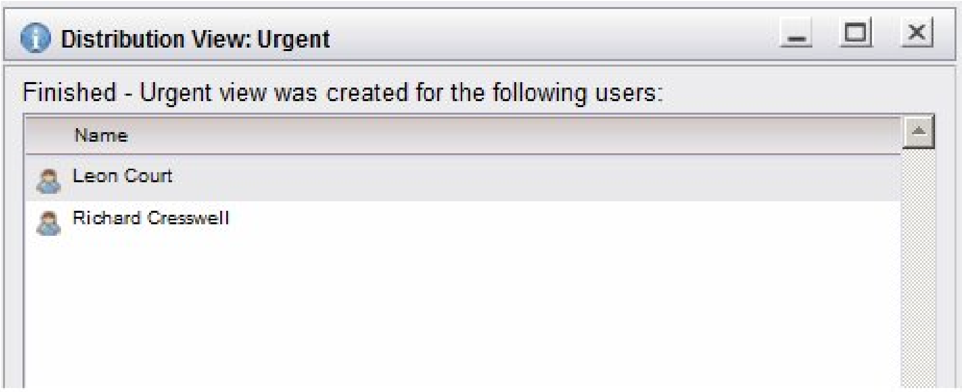
Name
John Eustace

The Duplicate View dialog lists only the users who did *not* receive the distributed view; the application will still distribute the view to the users who did not have a view by the same name (those users are listed on the Distribution View dialog, which is displayed after you close the Duplicate View dialog—see Step 5b).

If the Duplicate View dialog is displayed, you have the option of going back to the main dialog, and checking the **Replace view if users already have a view with the same name** check box, then re-attempting the distribution to the users that were listed in the Duplicate View dialog.

Click **OK** to close the Duplicate View dialog; the Distribution View dialog displays—see Step 5b.

b) The Distribution View dialog lists all of the users to whom the application successfully distributed the view:



Finished - Urgent view was created for the following users:

Name
Leon Court
Richard Cresswell

This dialog will not list any users if the application was not able to successfully distribute the view to any users.

6. Click **OK** to close the Distribution View dialog and return to the main window of the application.

   You can log out of the View Distributor application by clicking the **Logout** button, or close the application by closing the browser.

Browser Script Warnings

If you are distributing a view to a very large number of users, the process can take some time to complete. If the distribution is taking a long time, the browser may display a dialog that indicates there is a problem with a script running on the page.

All browsers provide a method to disable script timeouts; refer to your browser's help system for information about how to do that.

User Access Sample Application

The User Access sample application provides examples of controlling user access.

It provides examples of:

- Adding custom menus and toolbar buttons(1) to a WCC application
- Adding subordinate items to the custom menus and toolbar buttons1.
- Limiting access to specific items on custom menus and toolbar buttons1.
- Controlling access to various parts of a WCC application (including the Workspace application).
- Starting business services through code.

[6] For additional information about custom menus and toolbar buttons, see Adding Custom Menus and Toolbar Buttons.

Setting Up the User Access Sample Application

The User Access sample application must be set up prior to using it.

Procedure

1. Copy the following directory...

   `StudioHome\wcc\version\Samples\wccAccessSample\JSXAPPS\wccAccessSample\`

   ... to the following directory:

   `StudioHome\wcc\version\JSXAPPS\`

   where:

   - *StudioHome* is the directory in which TIBCO Business Studio was installed.
   - *version* is the version number of Workspace that was installed with TIBCO Business Studio.

2. Copy the User Access sample application's launch fragment from here...

   `StudioHome\wcc\version\Samples\wccAccessSample\wccAccessSample.html`

   ... to here:

   `StudioHome\wcc\version\`

   This is the file that will be used to start the sample application.

Deploying the wccAccessSample Application

A `wccAccessSample.create.war.cmd` file is provided with the User Access sample application that can be used to create a WAR file that can be used to deploy the sample application to a runtime node.

For detailed instructions on how to do this, see Deploying an Application After Customizing.

The Sample Application's User Access File

The User Access sample application comes with its own `userAccess.xml` file that has been customized to illustrate how access to various parts of a WCC application can be specified, including access to custom menu items and toolbar buttons.

If you are running the User Access sample application from your local development environment, you can view and modify, as needed, the `userAccess.xml` file in the following location:

`StudioHome\wcc\version\JSXAPPS\wccAccessSample\userAccess.xml`

If you have deployed the User Access sample application to a runtime node, you must use the Configuration Administrator to view and modify, as needed, the `userAccess.xml` file.

For information about modifying user access, see Configuring User Access.

The `userAccess.xml` file that accompanies this sample application does not use the default access settings. Instead, various user access sets grant subsets of functionality based on privileges.

A special "All System Actions" privilege is granted through the version 0 organization model, which is always present on an ActiveMatrix BPM installation. This privilege is held by the system administrative user (tibco-admin), and by any other resources that have been placed into the version 0 "System Administrator" group. A "System Administrator" user access set in `userAccess.xml` grants most functionality to users that have this privilege.

The `userAccess.xml` file included with the User Access sample application contains six other user access sets.

- **MinimalWorkList** - This grants a subset of functionality under work views and work item lists. It omits the ability to save views from the work list menu. The user can choose to open the next work item, but not to select and open other work items in the list. The user is also not allowed to use the Auto Open feature or to skip, pend, or allocate items to another user. They can choose to allocate an individual work item to themselves. This prevents other users from viewing or opening the allocated work item, although this user cannot open it until it reaches the top of the list, making it the next item. They can also re-offer an item previously allocated to themselves.

- **AdvancedWorkList** - This grants most of the remaining work view and work item list functionality not granted by MinimalWorkList; it does not grant any functionality outside of work views and work lists.
- **ProcessManager** - This grants functionality related to business services, process views, and process instances.
- **StartAnyBusinessService**, **StartFilteredBusinessServices** and **StartRemainingBusinessServices** - These each grant access to one custom menu item, and one custom toolbar button, implemented in the sample application. The custom menu and toolbar buttons appear on the work item list.

Each user access set is granted to a user through a privilege of the same name. These privileges are not part of the default version 0 organization model, and therefore must be separately implemented and deployed as part of a custom organization model.

Accompanying this sample application is a sample organization model named `AccessSample.om` that includes definitions for the six privileges listed above, along with corresponding groups to which users can be assigned. This organization model can be temporarily deployed alongside your own organization model and processes in order to work with the sample application. Information about this organization model is provided in Granting the Required Privileges Through an Organization Model.

Custom Menus and Toolbar Buttons

References to the `CustomWorkListToolbar.xml` and `CustomWorkListMenu.xml` files are used in the **customMenus** section of `config.xml` in the sample application to add custom menus and toolbar buttons to the work item list.

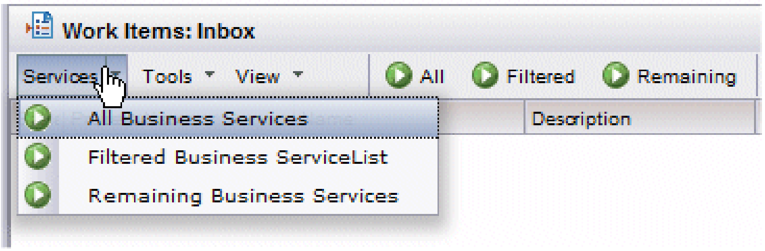
These files are located as follows:

```
.../wccAccessSample/application/prototypes/CustomWorkListToolbar.xml
.../wccAccessSample/application/prototypes/CustomWorkListMenu.xml
```

The `CustomWorkListMenu.xml` file then references the following file to add subordinate items to the custom menu:

```
.../wccAccessSample/application/prototypes/mnuCustomWorkListServices.xml
```

Each of the custom toolbar buttons, and each item on the custom menu, is for starting a business service. The different items let the user select from a different subset of business services. The following shows all of the custom menu items and toolbar buttons displayed. They would all be displayed only if the logged-in user possesses all of the required privileges, such as the system administrator (tibco-admin):



- **All Business Services Item - All Button** - Lets you select any business service. This requires the StartAnyBusinessService privilege.
- **Filtered Business ServiceList Item - Filtered Button** - Lets you select from a filtered list of business services. This requires the StartFilteredBusinessServices privilege.

- **Remaining Business Services Item - Remaining Button** - Lets you start the business services not included in the list of filtered business services. This requires the StartRemainingBusinessServices privilege.

Starting Business Services

All of the custom menu items and toolbar buttons call the **customStartBusinessService** function in the `AppMain.js` file.

This file is located as follows:

```
.../wccAccessSample/application/js/AppMain.js
```

The **customStartBusinessService** function displays a new maximized dialog with a list for selecting a business service. The contents of the list varies, depending on the button that was selected. Comments in the function describe how the list is retrieved from the server, and how it is filtered based on a parameter for the function.

When the user selects a business service in the list, the **customStartBusinessServiceSelected** function in `AppMain.js` is called.

The **customStartBusinessServiceSelected** function removes the list from the dialog and starts the selected business service using the same dialog as the parent container. It also subscribes to an event that is fired each time a form or a user task is completed in the business service. This calls the **closeFormDialog** function, which verifies whether this is the final form to be displayed, and if so, closes the dialog.

Controlling Access to Individual Custom Menu Items and Toolbar Buttons

Normally, either full access or no access is granted to all of the menu items or toolbar buttons for a particular type of list. However, in the User Access sample application, custom entries in `userAccess.xml`, and code in the application, are used to control whether individual buttons and menu items appear.

This is done through additional access tags added underneath the standard menu and toolbar items. For instance, the following is the privilege access that controls whether the items for starting any business service are displayed:

```
<PrivilegeAccessSet name="Start Any Business Service"
            description="Grants access to a custom menu/toolbar for
                         starting any business service">
   <privileges>
       <privilege name="StartAnyBusinessService"></privilege>
   </privileges>

   <access name="CustomAccess">
      <access name="CustomMenuAccess">
         <access name="menu">
            <access name="WorkItemList">
                <access name="StartAnyBusinessService"></access>
            </access>
          </access>
          <access name="toolbar">
             <access name="WorkItemList">
                 <access name="StartAnyBusinessService"></access>
             </access>
           </access>
       </access>
   </access>
</PrivilegeAccessSet>
```

The entries for <access name="StartAnyBusinessService"></access> have been added where subordinate items normally do not appear.

Note that the example here shows the custom access items being added directly to the `userAccess.xml` file, which is what you would do if you are customizing an undeployed application in your local development environment. If you were customizing an application that is deployed to a runtime node, you would use the Configuration Administrator— for more information, see Configuring User Access .

The other two user access sets that control access to the custom menu items and toolbar buttons (StartFilteredBusinessServices and StartRemainingBusinessServices) are similar.

Code in the following file controls whether or not the custom items are visible:

```
.../wccAccessSample/application/js/Application.js
```

In the **postLoadInit** function, the application subscribes to the **renderComplete** event of the work item list. When the work item list is initially displayed, the **createCustomServicesMenu** function in the same file is called. That function checks each of the custom settings to see if it has been granted through user access sets in `userAccess.xml`. For items that have not been granted, it deletes the corresponding custom menu item or toolbar button (comments inside the **createCustomServicesMenu** function provide more detail).

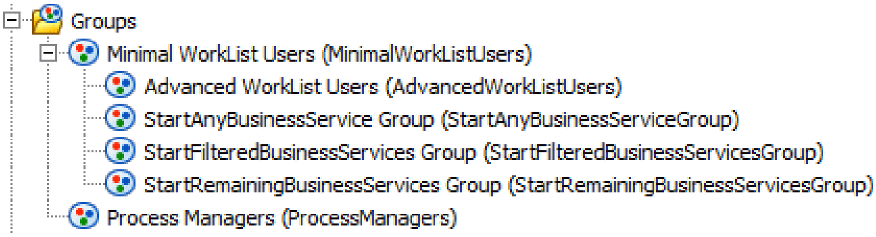
Granting the Required Privileges Through an Organization Model

The User Access sample application includes a sample organization model, `AccessSample.om`, that defines the privileges and groups used in the sample application.

Unless you deploy an organization model that grants the privileges referenced in the `userAccess.xml` file included in this sample application (such as AccessSample.om), only the tibco-admin user (or other System Administrators) will be able to do anything in the sample application.

`AccessSample.om` can co-exist with other organization models, so you can deploy it onto a test system that already has an organization model and processes, or you might choose to implement something into your own test organization model that is similar.

`AccessSample.om` includes the following groups:



- Groups
  - Minimal WorkList Users (MinimalWorkListUsers)
    - Advanced WorkList Users (AdvancedWorkListUsers)
    - StartAnyBusinessService Group (StartAnyBusinessServiceGroup)
    - StartFilteredBusinessServices Group (StartFilteredBusinessServicesGroup)
    - StartRemainingBusinessServices Group (StartRemainingBusinessServicesGroup)
  - Process Managers (ProcessManagers)

Each group is named for the privilege that it grants.

Note that several of the groups are nested underneath the Minimal WorkList Users group (which provides access to the work list itself, via the **MinimalWorkList** user access set in the `userAccess.xml` file). This is done because it would not make sense to grant the privilege associated for the subordinate without granting the privilege for the parent group. If the user cannot view a work list, it would not make sense to give them the ability to view individual custom menu items and toolbar buttons under that list, or to grant a few advanced work list-related functions. Privileges are inherited from parent groups, so if you make a person a member of Advanced WorkList Users, they will be granted both the MinimalWorkList and AdvancedWorkList privileges.

If you deploy the sample organization model (or incorporate something like it on your own) and run the sample application as tibco-admin, you can then add resources to the various groups.

When you log in as someone who is only a member of Minimal WorkList Users they will only be able to work with work views and work lists; some functions (those only granted through AdvancedWorkList) will not be available.

For a user that is a member of one or more of the Start*BusinessServices groups, the appropriate custom menu items and toolbar buttons are displayed.

The “filtered” list of business services includes those that include an “a” somewhere in the process name, and the “remaining” list is those that do not include the letter “a” in the process name.

Note that filtering business services on the process name containing an “a” is obviously not a realistic business case. It is used in the sample application merely as an example. It is expected that you would modify the following files to tie access to a more realistic business case.

```
.../wccAccessSample/application/prototypes/CustomWorkListToolbar.xml
.../wccAccessSample/application/prototypes/mnuCustomWorkListServices.xml
```

Also refer to the comments in the **customStartBusinessService** function in the following file for more information about how the sample application controls filtering:

```
.../wccAccessSample/application/js/AppMain.js
```

Login Session Sample Application

The Login Session sample application provides methods of establishing and persisting a login session.

This sample application is intended to be used in conjunction with the Sample Applications so that users do not need to provide a username and password every time the openworkitem application is invoked. For information about the openworkitem application, see Sample Applications.

Note that the openworkitem application already provides a means of using a current login session by passing “externalLogin” in the URL (see External Login). The Login Session sample application provides additional methods of establishing and using a login session.

Although the Login Session sample application is designed to work with the openworkitem application, it can be modified to work with any WCC application, including the Workspace application.

The Login Session sample application is provided with Workspace in the following directory:

```
StudioHome\wcc\version\samples\LoginSessionSample
```

where:

- *StudioHome* is the directory in which TIBCO Business Studio was installed.
- *version* is the version number of Workspace that was installed with TIBCO Business Studio.

The following files are included in the Login Session sample application:

- **amxbpm_item.js** - Contains code for determining the login state and to perform login and logout.
- **amxbpm_item.html** - A sample page that can be called instead of the **openworkitem** application.
- **amxbpm_login.html** - A simple login page that can maintain a login session.
- **amxbpm_item.create.war.cmd** - A command file that can be used to package the sample into a WAR file for deployment.
- **WEB-INF\web.xml** - A deployment descriptor file for inclusion in the WAR file.

The HTML files included in the Login Session sample application can be modified for your own use, or `amxbpm_item.js` can be incorporated for use in your own web page.

However, whether you are using the samples files as distributed, working with a modified version, or incorporating similar techniques in your own page, you must replace the base URL in the HTML files with one for your own ActiveMatrix BPM server.

Also, depending on the type and version of browser you use, you may need to host the pages on the ActiveMatrix BPM server to avoid issues with cross-domain scripting. For more information, see Cross-Domain Scripting.

amxbpm_item.js

The `amxbpm_item.js` file contains the code for detecting a login session, performing a login, and launching openworkitem or another WCC application.

When this file is loaded by an HTML page, an object named **amxbpm_port** is created and can be used by other script code in the page. The following functions are made available through that object:

- **amxbpm_port.setBaseUrl(strURL)**

  This function sets the base portion of the address of the ActiveMatrix BPM server. (Note that all other functions make use of this base URL, so this function must be called before any others.)

  If you are deploying the sample application to the same ActiveMatrix BPM server that is hosting Workspace, and the sample application is using the same httpConnector (the same port), specify a blank string for the base URL. This causes the URL to be determined at runtime based on the URL used to launch the application.

  If you are deploying the sample application to a different ActiveMatrix BPM server than the one hosting Workspace, or the sample application will be using a different httpConnector (port number), specify the ActiveMatrix BPM server and port number using this function. (Keep in mind cross-domain scripting issues when deploying to a different server/port than the one used by Workspace; see Cross-Domain Scripting.)

  For example:

  ```
  amxbpm_port.setBaseUrl("http://server_name:8080");
  ```

- **amxbpm_port.ping()**

  This function returns true if the browser currently has a login session established with the ActiveMatrix BPM server. This login session may exist because the user is already logged into another WCC application (including the Workspace application), or another web page could be open that has performed a login using the other methods provided in `amxbpm_item.js`.

- **amxbpm_port.login(userId, password)**

  This function attempts to log into the ActiveMatrix BPM server using the user ID and password that have been provided. If the login is successful, the function returns true.

- **amxbpm_port.logout()**

  This function logs out on the ActiveMatrix BPM server.

- **amxbpm_port.openWorkspace()**

  This function launches the Workspace application in a separate browser window.

- **amxbpm_port.open()**

  This function launches the **openworkitem** application in a separate browser window. If any of the following URL parameters were specified when launching the current page, they are forwarded to the **openworkitem** application:

  - Id;
  - Version;
  - ChannelId;
  - ProcessName;
  - ModuleName;
  - Description;

  The `externalLogin` parameter is automatically included in the information sent to the **openworkitem** application, but is based on the existence of a login session. It will be true if a login session is detected and false if not.

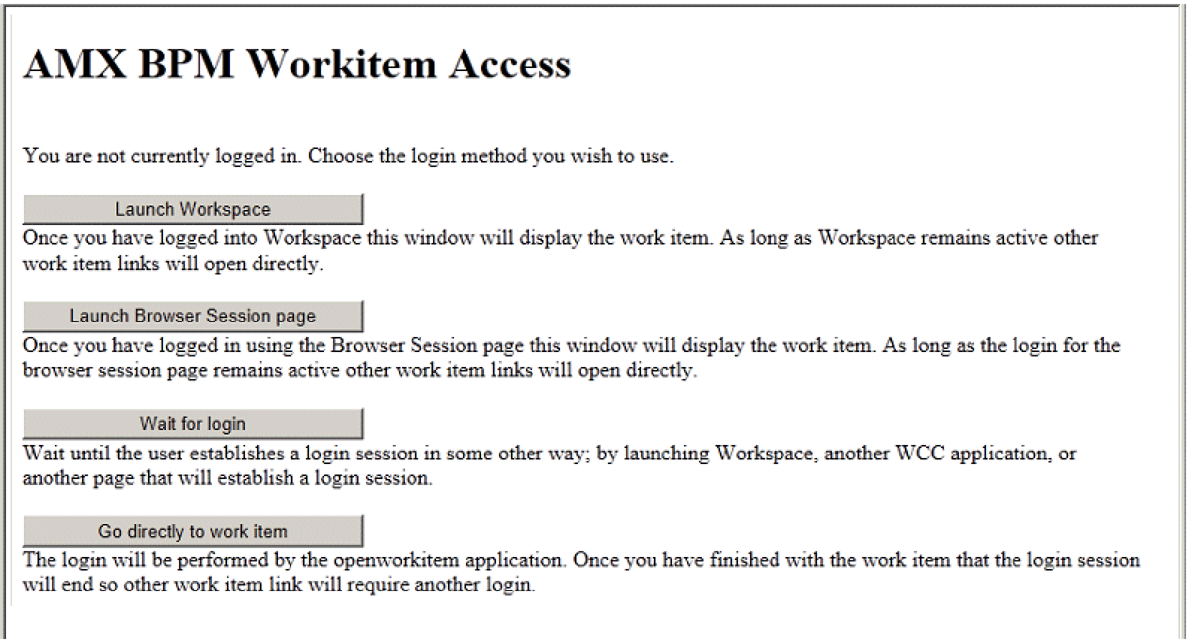
If none of the parameters listed above are specified, the first work item in the logged-in user's Inbox is automatically displayed.

amxbpm_item.html

The `amxbpm_item.html` file shows ways in which the functions in `amxbpm_item.js` can be used.

If there is a current login session when this page is invoked, the page uses the `amxbpm_port.open()` function to load the openworkitem application in the current browser window. The URL parameters used to open the current page are forewarded to the openworkitem application as described earlier to display the approriate work item or business service.

If there is no current login session when this page is invoked, the following page is displayed:



AMX BPM Workitem Access

You are not currently logged in. Choose the login method you wish to use.

Once you have logged into Workspace this window will display the work item. As long as Workspace remains active other work item links will open directly.

Once you have logged in using the Browser Session page this window will display the work item. As long as the login for the browser session page remains active other work item links will open directly.

Wait until the user establishes a login session in some other way; by launching Workspace, another WCC application, or another page that will establish a login session.

The login will be performed by the openworkitem application. Once you have finished with the work item that the login session will end so other work item link will require another login.

Each of the buttons on this page illustrates a different way of establishing and using a login session, as follows:

- **Launch Workspace** - This calls the `amxbpm_port.openWorkspace()` function to start the Workspace application in a separate browser window. The original window takes the same action as the **Wait for login** button (see below), calling `amxbpm_port.ping()` every couple of seconds, waiting for a login session to be established. As soon as a login is established on the Workspace Login dialog, the **openworkitem** application is started.
- **Launch Browser Session page** - This executes `amxbpm_login.html`, causing a separate browser window to display, containing **Name** and **Password** fields. The original window takes the same action as the **Wait for login** button (see below), calling `amxbpm_port.ping()` every couple of seconds, waiting for a login session to be established. As soon as a login is established via the **Name** and **Password** fields, the **openworkitem** application is started.
- **Wait for login** - This causes the `amxbpm_port.ping()` function to be called every couple of seconds, waiting for a login session to be established. As soon as a login is established, the openworkitem application is started.
- **Go Directly to work item** - This calls the `amxbpm_port.open()` function to start the openworkitem application. If there is already a login session, the first work item in the logged-in user’s Inbox is displayed; if there is no current login session, the Login dialog is displayed.

Before launching the axbpm_item.html page, specify the base URL using the setBaseURL function, as follows (depending on where the application is deployed, you may want to specify an empty string with this function; for more information, see amxbpm_item.js):

```
amxbpm_port.setBaseURL (“http://myServer:port”);
```

If you do not want the user to select the method of logging in, the code in the doOnLoad() function can be modified to call one of the functions associated with a button.

There is also a doCustomLogin() function in amxbpm_item.html that loads a custom WCC application. To use this function, you need to alter the URL in the function to point to the actual deployed application.

amxbpm_login.html

The amxbpm_login.html file is a simple HTML page that uses amxbpm_item.js code to perform a login. The only purpose of the page is to maintain a login session.

amxbpm_item.create.war.cmd

The amxbpm_item.create.war.cmd file can be used to bundle the files in the Login Session sample application into a WAR file so that they can be deployed to a node.

For details about how to do this, see Deploying an Application After Customizing. (If you follow these instructions to create a WAR file, use the amxbpm_item.create.war.cmd file in the first step. Also, as this is not an actual WCC application, do not perform the optional step for adding an extension point for language pack support.)

The ...\WEB-INF\web.xml file included with the sample application is a deployment descriptor file for inclusion in the WAR file.

Workspace Logs

There are two types of Workspace logs available—the Application Log and the Application Monitor.

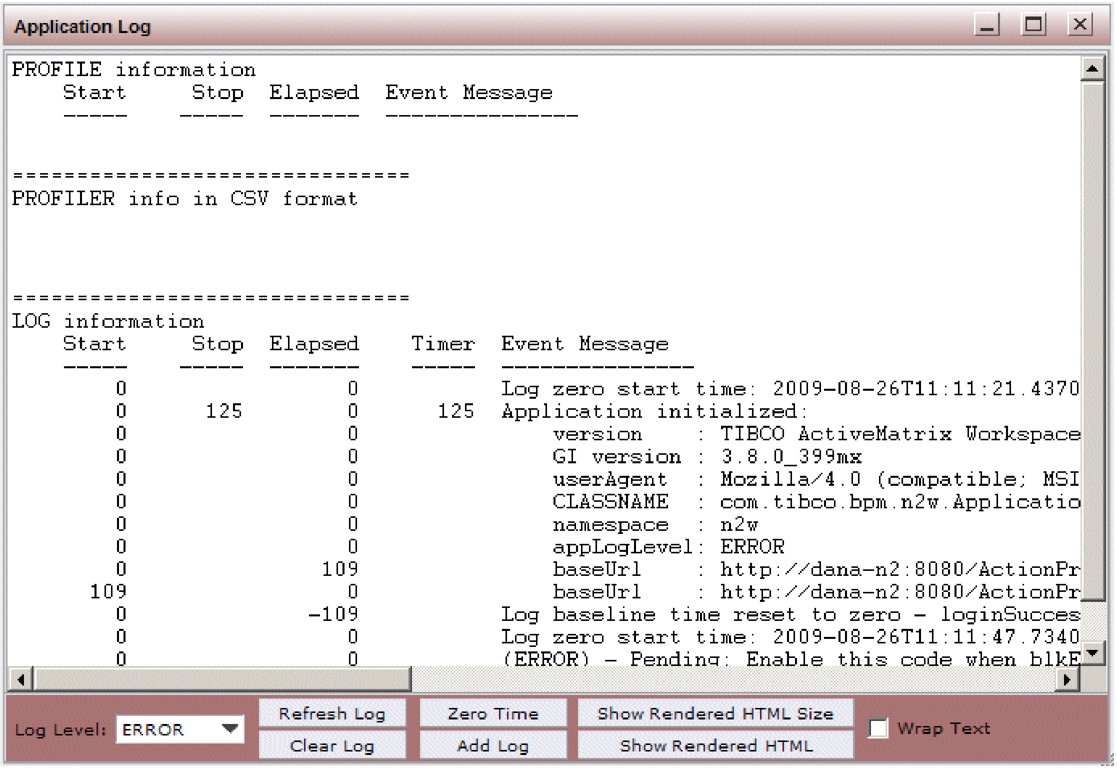
- **Application Log** - This log provides detailed debug information, as well as communications between the application and the Action Processor. For more information, see Application Log.
- **Application Monitor** - This log provides debug information on error conditions and exceptions encountered. For more information, see Application Monitor.

Application Log

The Application Log is available to assist with troubleshooting the application. This log provides detailed debug information generated by the application, as well as information about communications between the application and the Action Processor.

To be able to display the Application Log, the logged-in user must have the **ApplicationLog** user access enabled (for more information, see Available Functions).

To display the Application Log, press the **F12** function key while the application is running. A window similar to the following is displayed:



```
PROFILE information
    Start     Stop  Elapsed  Event Message
    -----    -----  -------  ---------------


===============================
PROFILER info in CSV format



===============================
LOG information
    Start     Stop  Elapsed    Timer  Event Message
    -----    -----  -------    -----  ---------------
        0                 0           Log zero start time: 2009-08-26T11:11:21.4370
        0      125        0      125  Application initialized:
        0                 0               version    : TIBCO ActiveMatrix Workspace
        0                 0               GI version : 3.8.0_399mx
        0                 0               userAgent  : Mozilla/4.0 (compatible; MSI
        0                 0               CLASSNAME  : com.tibco.bpm.n2w.Applicatio
        0                 0               namespace  : n2w
        0                 0               appLogLevel: ERROR
        0               109               baseUrl    : http://dana-n2:8080/ActionPr
      109                 0               baseUrl    : http://dana-n2:8080/ActionPr
        0              -109           Log baseline time reset to zero - loginSucces
        0                 0           Log zero start time: 2009-08-26T11:11:47.7340
        0                 0           (ERROR) - Pending: Enable this code when blkE
```

After initiating a function you want to view in the log, click **Refresh Log** to add it to the display.

The Application Log can be closed by clicking in the X in the upper right corner of the Application Log window.

Note that you can specify the level of log messages that are written to the log by selecting the desired level in the **Log Level** field drop-down list. The default log level is specified in the `config.xml` file—see Configuring the Application Log.

Configuring the Application Log

The Application Log can be configured using the **logging** record in the application's `config.xml` file.

Procedure

1. Open the `config.xml` file.

   If you are configuring a deployed application, open `config.xml` via the Configuration Administrator; if you are configuring a non-deployed application, open `config.xml` via the file system. For more information, see Introduction.

2. Locate the **logging** record:

```
<record jsxid="logging" type="Workspace"
    appLogLevel="ERROR"    echoToJsxLog="false">
</record>
```

3. Set the **logging** record's attributes as follows:

   a) Set the **appLogLevel** attribute to indicate the default value for the **Log Level** drop-down list in the Application Log. The valid entries are:

   - OFF
   - FATAL
   - ERROR
   - WARN
   - INFO
   - DEBUG
   - TRACE

   b) Set the **echoToJsxLog** attribute to indicate if the log contents should be echoed to the Application Monitor (see Application Monitor), as follows:

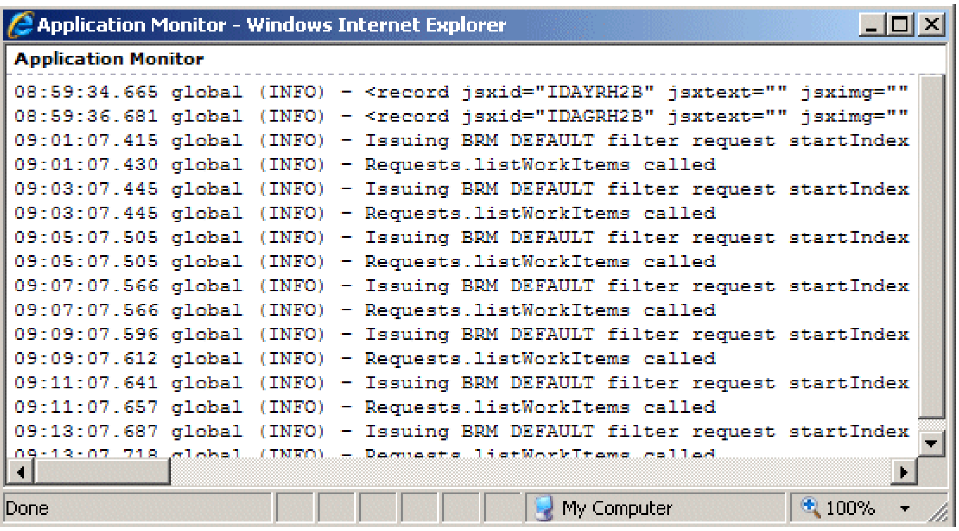
   - "true" causes the contents of the Application Log to be echoed to the Application Monitor.
   - "false"causes the contents of the Application Log to not be echoed to the Application Monitor.

4. Save and close the `config.xml` file.

Application Monitor

The Application Monitor is available to assist with troubleshooting the application. This monitor provides debug information on error conditions and exceptions encountered.

The Application Monitor is displayed in a separate browser window, which shows details of actions performed in the application. An example is shown below:



```
08:59:34.665 global (INFO) - <record jsxid="IDAYRH2B" jsxtext="" jsximg=""
08:59:36.681 global (INFO) - <record jsxid="IDAGRH2B" jsxtext="" jsximg=""
09:01:07.415 global (INFO) - Issuing BRM DEFAULT filter request startIndex
09:01:07.430 global (INFO) - Requests.listWorkItems called
09:03:07.445 global (INFO) - Issuing BRM DEFAULT filter request startIndex
09:03:07.445 global (INFO) - Requests.listWorkItems called
09:05:07.505 global (INFO) - Issuing BRM DEFAULT filter request startIndex
09:05:07.505 global (INFO) - Requests.listWorkItems called
09:07:07.566 global (INFO) - Issuing BRM DEFAULT filter request startIndex
09:07:07.566 global (INFO) - Requests.listWorkItems called
09:09:07.596 global (INFO) - Issuing BRM DEFAULT filter request startIndex
09:09:07.612 global (INFO) - Requests.listWorkItems called
09:11:07.641 global (INFO) - Issuing BRM DEFAULT filter request startIndex
09:11:07.657 global (INFO) - Requests.listWorkItems called
09:13:07.687 global (INFO) - Issuing BRM DEFAULT filter request startIndex
09:13:07.718 global (INFO) - Requests.listWorkItems called
```

The Application Monitor can be configured using the following configuration file:

StudioHome\wcc\version\logger.xml

where:

- *StudioHome* is the directory in which TIBCO Business Studio was installed.
- *version* is the version number of Workspace that was installed with TIBCO Business Studio.

Default settings are specified by the following **handler** element in the `logger.xml` file:

```
<handler name="workpsaceAppMonitor" class="jsx3.app.Monitor" require="true">
   <property name="serverNamespace" value="workspace"/>
   <property name="disableInIDE" eval="true" value="true"/>
   <property name="activateOnHotKey" eval="true" value="true"/>
   <property name="format" value="%t %n (%l) - %M"/>
</handler>
```

A reference to this handler is added to the <**handler-ref**> element under the global **logger** element:

```
<logger name="global" level="INFO">
   <handler-ref name="memory"/>
   <handler-ref name="ide"/>
   <handler-ref name="fatal"/>
   <handler-ref name="workspaceAppMonitor"/>
</logger>
```

By default, both the Application Monitor and its hotkeys are enabled.

- To disable the Application Monitor, comment out the entire <**handler**/> element, as well as the <**handler-ref**/> element under the global **logger** element. (Note that if you comment out the Application Monitor, you must comment out both the <**handler**/> element, as well as the <**handler-ref**/> element that references it. If the **<handler/** element is commented out, but the **<handler-ref/** element is not commented out, it results in a fatal error—the application will not load.)
- If the **activateOnHotKey** property's **value** attribute is set to "false", the Application Monitor is displayed automatically upon application start. If the **activateOnHotKey** property's **value** attribute is set to "true", the hotkey sequence (**<Ctrl** > **+<Alt** > **+<m** > by default), must be pressed to display the Application Monitor.

The level of the log messages can be set by changing the value of the **level** attribute in the **<logger name="global"** record. The valid levels are:

- FATAL
- ERROR

- WARN
- INFO
- DEBUG
- TRACE

You can also specify that Application Log data be echoed to the Application Monitor. This is accomplished using the **echoToJsxLog** attribute in the **logging** record in the application’s `config.xml` file. For more information, see Configuring the Application Log.

Performance

This topic describes things you can do to improve the performance of your WCC applications (including the Workspace application).

Removing Unneeded Locale Keys

Preventing unused locales from loading increases the speed that WCC applications load, especially over slow connections, such as Wide Area Networks (WANs) and Virtual Personal Networks (VPNs).

Procedure

1. Open the `locale.xml` file.

   If you are opening this file for a WCC application in your development environment, it is located in the following directory:

   ```
   StudioHome\wcc\version\JSXAPPS\WCCProjectName\JSXAPPS\base\locale
   ```

   where:

   - *StudioHome* is the directory in which TIBCO Business Studio was installed.
   - *version* is the version number of Workspace that was installed with TIBCO Business Studio.
   - *WCCProjectName* is the name of the General Interface Builder project that contains your custom application. If you are working with the Workspace application, this is “workspace”.

     If you are opening the `locale.xml` file for the Workspace application that is deployed to the runtime node, see Location of Files on a BPM Runtime Machine for information about the location of the `\JSXAPPS\base\locale` directory on the runtime node.

2. Locate the following line:

   <data jsxnamespace="propsbundle" locales="de,es,es_ES,es_MX,fr,fr_FR,fr_CA,id,ja,ko,pl,pt,pt_BR,ru,th,tr,vi,zh,zh_CN,zh_HK,zh_TW,en_GB">

3. Remove all of the locale keys that you are not using from the **locales** attribute.

   For example, if your application is using only the various English and French locales, it should appear as follows:

   <data jsxnamespace="propsbundle" locales="fr,fr_FR,fr_CA,en_GB">

4. Save and close the `locale.xml` file.

   There is no need to delete any of the language-specific locale files that are not used.