



# **TIBCO iProcess® Workspace (Browser)**

## **Components Concepts**

Version 11.10.0 | May 2025

# Contents

---

<b>Contents</b>	<b>2</b>
<b>Introduction</b>	<b>4</b>
Product Overview	4
TIBCO iProcess Environment	5
TIBCO® General Interface	5
Browser	6
iProcess Application	6
TIBCO iProcess Workspace (Browser) Action Processor	7
TIBCO iProcess Server Objects	7
TIBCO iProcess Objects Server	8
TIBCO iProcess Engine	8
<b>Building Custom Applications</b>	<b>10</b>
Introduction	10
Application Configuration	11
WCC Configuration File	11
TIBCO® General Interface Builder Workspace	12
Deploying Your Custom Application	13
Deploying to a Portal	14
Launching a Custom WCC Application	14
Starting Multiple Applications from the Launch Fragment	16
Multiple Applications Referencing a Single Configuration File	19
Launching a WCC Application in an HTML Frame	20
<b>Communications</b>	<b>21</b>
Communications Overview	21
Loading PageBus	22

Publishing and Subscribing to Events .....	22
<b>Events Editor .....</b>	<b>43</b>
Using the Events Editor .....	43
Subscribing to Events .....	46
Events From External Applications .....	48
Creating the External Application Event Definition .....	48
Importing the Event Definition File .....	49
<b>Properties Editor .....</b>	<b>51</b>
Using the Properties Editor .....	51
Component Names .....	54
Changing Component/Model Names .....	55
Disabling Default Handler .....	57
<b>Tutorial .....</b>	<b>59</b>
Creating an Application .....	59
Tutorial Screen Layout .....	69
<b>Sample Applications .....</b>	<b>74</b>
Introduction .....	74
openWorkQViaURL .....	74
Setting Up and Using openWorkQViaURL .....	75
URL Parameters .....	76
openWorkItemViaURL .....	77
Setting Up and Using openWorkItemViaURL .....	78
URL Parameters .....	79
Opening a Work Item .....	79
Starting a Case .....	80
<b>TIBCO Documentation and Support Services .....</b>	<b>82</b>
<b>Legal and Third-Party Notices .....</b>	<b>84</b>

# Introduction

---

This section provides an introduction to the TIBCO iProcess® Workspace (Browser) components.

## Product Overview

The **TIBCO iProcess Workspace (Browser)** components provide a set of data-aware user interface components that may be used to quickly develop full-screen web applications, embed iProcess functionality in web pages, or serve as portlets in a portal application. They allow the developer to quickly incorporate iProcess functionality into BPM-aware AJAX applications without requiring a working knowledge of the iProcess API's that are needed to access and manage iProcess data.

Starting with the 11.6.0 version, TIBCO® General Interface Builder and the TIBCO® General Interface Runtime are bundled with the iProcess Workspace (Browser) Components. Only the GI Builder and Runtime included with the iProcess Workspace (Browser) Components should be used to build WCC applications. It is no longer required to separately install GI Builder or GI Runtime.

The TIBCO iProcess Workspace (Browser) components are created with TIBCO® General Interface Builder and are composed of standard TIBCO® General Interface components that have been integrated and enhanced to provide the functionality and user interfaces required of a typical iProcess client application. At design-time, the components are available as drag and drop tools in the Component Libraries of the TIBCO® General Interface Builder development environment.

The following are examples of the functionality you can add to your web application or portlet with the TIBCO iProcess Workspace (Browser) components:

- Log into, and out of, the server
- Display lists of procedures, work queues, cases, and work items
- Filter or sort lists of procedures, work queues, cases, and work items
- Start cases of procedures
- Process (open, keep, release, etc.) work items

A complete list of the functions available through the TIBCO iProcess Workspace (Browser) components can be found in the *TIBCO iProcess Workspace (Browser) Components Reference*.

For information about using the functions available through the TIBCO iProcess Workspace (Browser) components, see the *TIBCO iProcess Workspace (Browser) User Guide*.

Custom applications built with the TIBCO iProcess Workspace (Browser) components are configured in the same way in which the client application provided with the TIBCO iProcess Workspace (Browser) is configured. The available configuration options are described in the *TIBCO iProcess Workspace (Browser) Configuration and Customization* guide. All of the configuration options described in this configuration guide also apply to custom applications built with components (e.g., using callout methods to perform custom filtering, using the GI forms interface methods to display forms generated with TIBCO® General Interface Builder, using single authentication to bypass the **Login** screen, etc.).

## TIBCO iProcess Environment

The major items/products that are incorporated into an iProcess environment when using TIBCO iProcess Workspace (Browser) components are described in the following subsections.

## TIBCO® General Interface

TIBCO® General Interface consists of two components:

- **TIBCO® General Interface Builder** - This is a browser-based, visual development environment used to develop AJAX applications. AJAX (asynchronous communications, JavaScript, and XML) applications are run in a standard browser, but look and feel like a desktop-run application.

TIBCO® General Interface Builder includes a Components Libraries Palette that contains many standard components used to add functionality to your application, such as menus, taskbars, text boxes, etc.

After the TIBCO iProcess Workspace (Browser) components are installed, these “WCC” components<sup>1</sup> are also available in the TIBCO® General Interface Builder

---

<sup>1</sup>From within TIBCO® General Interface Builder, the iProcess Workspace (Browser) components are referred as “WCC” components, which stands for “Workspace Client Components”.

Components Libraries. The TIBCO iProcess Workspace (Browser) components provide functionality that are iProcess-specific, and provide access to the data stored in the iProcess database.

Information about using and configuring the TIBCO iProcess Workspace (Browser) components in the TIBCO® General Interface Builder is provided in the remainder of the chapters of this guide.


- **TIBCO® General Interface Framework** - This is a distributable runtime framework for running browser-based applications developed with TIBCO® General Interface Builder.

## Browser

AJAX applications developed using the TIBCO® General Interface Builder and TIBCO iProcess Workspace (Browser) components are certified to run on most recent browsers. For the specific browser versions, see the readme provided with the product.

As newer versions of browsers are released, they may work with TIBCO iProcess Workspace (Browser), depending on their backward compatibility. Check with TIBCO Support to see if there are any known issues.

No other software (applets, plug-ins, or Active X controls) is required to run applications developed with the TIBCO iProcess Workspace (Browser) components.

 **Note:** See the TIBCO® General Interface Builder documentation for browsers that can be used to develop applications using TIBCO® General Interface Builder.

## iProcess Application

AJAX applications developed using the TIBCO® General Interface Builder and TIBCO iProcess Workspace (Browser) components can be full-screen applications, portlets in a portal application, or a specific piece of iProcess functionality embedded in a web page.

The size of the component interface displayed to the user is controlled by the container in which the component is placed.

## TIBCO iProcess Workspace (Browser) Action Processor

The TIBCO iProcess Workspace (Browser) Action Processor services action requests made by applications developed using the TIBCO® General Interface Builder.

Each 'action' requested by TIBCO® General Interface Builder GUI components (including TIBCO iProcess Workspace (Browser) components) can contain one or more of the following 'request' types:

- Request for TIBCO iProcess data (e.g., a list of work items)
- Request an action be executed on the TIBCO iProcess Engine (e.g., start a case)
- Request data to be displayed by a third-party (e.g., TIBCO BusinessWorks Formflow, ASP.NET Form, JSP Form)

The TIBCO® General Interface Builder GUI components send their action details to the Action Processor via a Form POST over HTTP — the action details are sent as XML on an HTML `<input>` tag.

The Action Processor also accepts actions via an HTTP GET. When doing a GET, the action XML needs to be sent as a URL parameter named 'action'.

Upon receiving an action request, the Action Processor breaks the action into individual requests and processes each one. The resulting data from each request is then consolidated in XML under one root element (`<ap:ActionResult>`). The Action Processor then returns this XML data to the TIBCO® General Interface Builder GUI components over HTTP (or in the case of a form request, it redirects the HTML — no XML is returned).

The Action Processor is implemented as both a Microsoft .NET ASP Web Application and as a Java Servlet. This has been done so as to satisfy both the Microsoft and UNIX customer base. The implementation of the two Action Processors is obviously very different but the functionality and configuration is the same.

## TIBCO iProcess Server Objects

The TIBCO iProcess Server Objects comprise a set of objects that are used to build applications that automate business processes. These objects contain properties/methods that are used to request iProcess data and perform iProcess functions such as starting cases, processing work items, performing administrative functions, etc.

The TIBCO iProcess Workspace (Browser) Action Processor makes calls to the methods available on the iProcess Server Objects, based on requests from the iProcess application. These method calls cause messages to be sent to an iProcess Objects Server, which in turn, makes calls to an API that is exposed on the Process Engine.

The following are two types of iProcess Server Objects available:

- TIBCO iProcess Server Objects (Java) - These objects are used in conjunction with the Java Action Processor.
- TIBCO iProcess Server Objects (.NET) - These objects are used in conjunction with the .NET Action Processor.

## TIBCO iProcess Objects Server

The TIBCO iProcess Objects Server acts as a gateway between the TIBCO iProcess Server Objects and the TIBCO iProcess Engine. It processes the request, then makes the appropriate call to an iProcess Engine to initiate the desired service or obtain the desired data.

Multiple instances of the iProcess Objects Server can be run to improve load balancing and efficiency. When this is done, you would incorporate a TIBCO iProcess Objects Director. The iProcess Objects Director is a standalone program that maintains a list of TIBCO iProcess Objects Servers that are configured in a node cluster. When a client application needs access to a TIBCO iProcess Objects Server, it first establishes a connection to the TIBCO iProcess Objects Director. The TIBCO iProcess Objects Director then decides, based on a “pick method,” which TIBCO iProcess Objects Server the client should connect to. The list of known TIBCO iProcess Objects Servers is updated dynamically as TIBCO iProcess Objects Server instances are started and stopped.

## TIBCO iProcess Engine

The TIBCO iProcess Engine processes all requests from iProcess applications, as well as maintains all data in the iProcess database.

A TIBCO iProcess Engine installation on one machine is known as a TIBCO iProcess Engine node. Installing the iProcess Engine on multiple machines that all use the same database instance is known as a node cluster architecture. In the node cluster architecture, you can have a number of iProcess Engine processes running on different machines to improve load



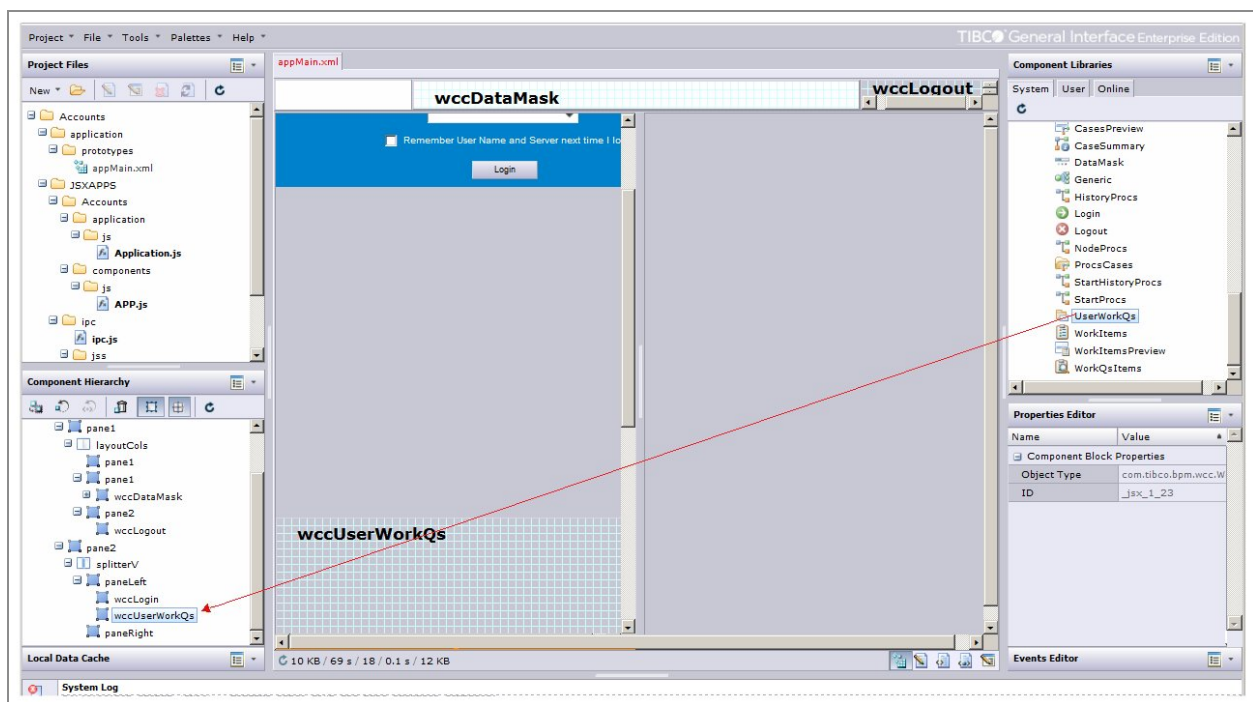
balancing and performance. However, all the iProcess Engine case data such as fields and their values in forms will be stored in one iProcess Engine database instance.

# Building Custom Applications

This chapter provides information about building custom applications using WCC components.

## Introduction

Building a custom application with iProcess Workspace (Browser) components is as easy as dragging and dropping the appropriate “WCC” (workspace client component) components into containers in TIBCO® General Interface Builder, then subscribing to the appropriate events using the Events Editor.



For details of how to do this, see the tutorial on [Tutorial](#).

# Application Configuration

It's important to understand that applications built with the iProcess Workspace (Browser) components use the same infrastructure as the iProcess Workspace (Browser) client application (see [TIBCO iProcess Workspace \(Browser\) Action Processor](#)).

When you install an iProcess Workspace (Browser) client application, a `config.xml` file is created that is used to configure the client application. And when you create a custom application with the iProcess Workspace (Browser) components, it will have its own `config.xml` file that is used to configure that application.

All configuration information that is described in the *TIBCO iProcess Workspace (Browser) Configuration and Customization* guide applies to both the iProcess Workspace (Browser) client application, as well as your custom application. For instance, in your custom application, you can use callout methods to perform custom filtering, the GI forms interface methods to display forms generated with TIBCO® General Interface Builder, single authentication to bypass the **Login** screen, etc. All of these are described in detail in the configuration guide.

- To configure the iProcess Workspace (Browser) client application, modify the `config.xml` file in the directory in which you installed the client.
- To configure your custom application, modify the `config.xml` file in the directory in which you created that application. The custom application's configuration file is located as follows:

```
WorkspaceDir\JSXAPPS\WCCProjectName\config.xml
```

where *WorkspaceDir* is the workspace directory you specified after starting TIBCO® General Interface Builder, and *WCCProjectName* is the name you gave the WCC project/application.

## WCC Configuration File

There is also 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 TIBCO® General Interface Builder. This file is located as follows:

```
WorkspaceDir\JSXAPPS\WCCProjectName\wccConfig.xml
```

where *WorkspaceDir* is the workspace directory you specified after starting TIBCO® General Interface Builder, and *WCCProjectName* is the name you gave the WCC Project when creating your custom application in TIBCO® General Interface Builder.

**Note:** The **wccapppath** attribute in the application's launch fragment (see [Launching a Custom WCC Application](#)) 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.

## TIBCO® General Interface Builder Workspace

When TIBCO® General Interface Builder is first started to build a custom WCC application, you are asked to establish a *TIBCO® General Interface Workspace*, which is the directory in which your custom application is saved.

Normally, your TIBCO® General Interface workspace is the installation directory of TIBCO® General Interface Builder (which is also the directory in which you install the WCC components).

**Note:** TIBCO General Interface Workspace only supports Internet Explorer 11.

**Note:** The WCC installation directory should not be the same as the webserver directory as the General Interface Builder cannot be accessed from http.

However, if you would like to use a workspace different than the TIBCO® General Interface Builder installation directory, copy the following directories from the TIBCO® General Interface installation directory to your desired workspace:

- JSXAPPS\ipc\

- JSXAPPS\wcc\
- addins\forms\
- addins\forms2\

Also copy the following files from the TIBCO® General Interface installation directory to the root of your workspace:

- externalform.htm
- jsx3.gui.window.html

## Deploying Your Custom Application

The following lists the files/directories that need to be deployed with your custom WCC application:

```
WCCProjectName.html
logger.xml
close.html
externalform.htm
jsx3.gui.window.html
\dojo-toolkit
\addins
\JSX
\JSXAPPS
  \WCCProjectName
  \wcc
  \ipc
```

where:

- **WCCProjectName.html** is the launch fragment for your custom application, where *WCCProjectName* is the name you gave the WCC project when creating your custom application in TIBCO® General Interface Builder.
- **logger.xml** handles logging iProcess Workspace (Browser) errors.
- **close.html** handles the closing of a separate browser windows when a workitem form is opened in a separate browser window.
- **externalform.htm** handles opening external workitem forms.
- **jsx3.gui.window.html** handles opening workitems in separate browser windows.

- **\dojo-toolkit** is the Dojo Toolkit which is used by TIBCO Forms.
- **\addins** contains the TIBCO Forms add-ins needed at runtime.
- **\JSX** contains only the TIBCO® General Interface runtime included with the iProcess Workspace (Browser) Components.
- **\JSXAPPS\WCCProjectName** contains your custom application, where *WCCProjectName* is the name you gave the WCC project when creating your custom application in TIBCO® General Interface Builder.
- **\JSXAPPS\wcc** contains the WCC components software.
- **\JSXAPPS\ipc** contains the base-level code needed by custom applications.

Each of these files\directories need to be deployed, in the structure shown, to the web server that will be hosting your custom application.

## Deploying to a Portal

Deploying your custom application to a portal generally involves wrapping the inline DIV created by the TIBCO® General Interface Builder Deployment Utility in the portal page, as appropriate for your portal server.

An example of launching multiple WCC applications in one launch fragment is shown in [Starting Multiple Applications from the Launch Fragment](#). This example simulates a portal environment, except that it launches two applications within one container, rather than each application being launched within a separate portal.

For details about running applications in a specific portal server, see the documentation for your portal server.

## Launching a Custom WCC Application

When a WCC application is created in TIBCO® General Interface Builder, a launch fragment is created that is used to launch your custom application. This launch fragment is located as follows:

```
WorkspaceDir\WCCProjectName.html
```

where *WorkspaceDir* is the workspace directory you specified after starting TIBCO® General Interface Builder, and *WCCProjectName* is the name you gave the WCC project when creating your custom application in TIBCO® General Interface Builder.

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.

To launch a WCC application that has been deployed to a Web server, point a browser at the application’s launch fragment, as follows:

```
http://Host:Port/ClientDir/WCCProjectName.html
```

where:

- *Host* is the machine name on which the application and Action Processor are being hosted.
- *Port* is the port number used by the Web server to communicate with web applications.
- *ClientDir* is the directory (or virtual directory alias) in which you deployed the WCC application files.
- *WCCProjectName* is the name you gave the WCC project when creating your custom application in TIBCO® General Interface Builder

For example:

```
http://Roxie:8090/AccountProject/Accounts.html
```

## 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.

- **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 Your 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 queue list when the user logs in. **AccountDetail** will subscribe to the single-click event of the **UserWorkQs** 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 queue in the work queue list displayed by **Accounts**.

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

For information about how to create a custom application using components, see [Tutorial](#).

2. With **Accounts** open in TIBCO® 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 [Events From External Applications](#).

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

4. With **AccountDetail** open in TIBCO® 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:

```
WorkspaceDir\JSXAPPS\Accounts\defs\Accounts.app.pub.xml
```

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

7. Subscribe to the **List Item Select (single click)** event on the **WorkQs** component.
8. Save **AccountDetail**.

At this point, we have created two separate applications, and they each have their own launch fragment that starts the individual application. We 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 *WorkspaceDir* directory, as file *Accounts.html*.)
10. Modify the new launch fragment as follows:

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

Uncomment the **<div />** that is located near the bottom of the file.

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"
  jsxapps="AccountDetail" jsxapppath="JSXAPPS/AccountDetail"
  wccapppath="JSXAPPS/AccountDetail" wccloadorder="2" >
</script>
</div>
```

Now we 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.

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.

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.

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® General Interface</title>
<script type="text/javascript" src="JSX/js/JSX30.js"
  wccapppath="JSXAPPS/Accounts" wccloadorder="1" >
</script>
</head>
<body SCROLL="NO" style="height: 100%; width: 100%; left: 0px; top: 0px; padding: 0px; margin: 0px; border: 0px; overflow: hidden;" >
<div id="Accounts" style="position: absolute; left: 0px; top: 0px; width: 100%; height: 30%; " >
<script type="text/javascript" src="JSX/js/JSX30.js"
  jsxapps="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"
  jsxapps="AccountDetail" jsxapppath="JSXAPPS/AccountDetail"
  wccapppath="JSXAPPS/AccountDetail" wccloadorder="2" >
</script>
</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 `userAccessProfiles.xml` located in that path.



**Note:** 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.

## Launching a WCC Application in an HTML Frame

To launch 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.

Prior to launching the application in a frame, perform the following steps:

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">body{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 (do not, however, remove the second **<script>** element following 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> -->
<script type="text/javascript" src="JSX/js/JSX30.js"
  jsappns="wccApp" jsxapppath="JSXAPPS/1pc/"
  wccapppath="JSXAPPS/1pc/" wccloaderdev="0" >
</script>
```

6. Save and close the launch script.

# Communications

---

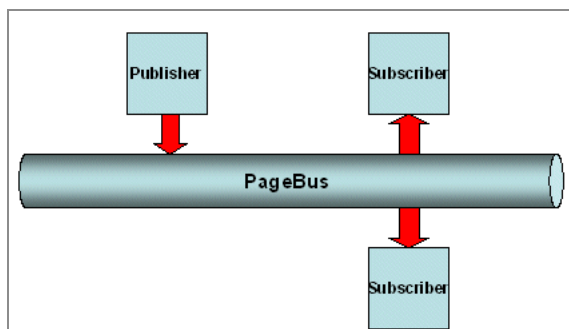
This chapter describes how the TIBCO iProcess Workspace (Browser) components communicate with one another.

## Communications Overview

The TIBCO iProcess Workspace (Browser) components (also known as WCC<sup>1</sup> components) use events to publish actions and to pass business data through the communications channel. This is accomplished using the **TIBCO PageBus™**.

The PageBus is a pure JavaScript, publish-subscribe message delivery hub that uses string subjects to identify business-level events, such as a user's selection of a work item, or when a user closes a case.

Subscribers listen (or subscribe) to subjects. Publishers send (or publish) messages on subjects. When publishers publish messages on a subject, the messages are delivered to all of the subject's current subscribers.



When you add an iProcess Workspace (Browser) component to your application, it is automatically configured to publish all of its public events to the PageBus. Then, for each component you've added, you must specify which events you want that component to subscribe to. This is accomplished using the Events Editor in TIBCO® General Interface Builder — the Events Editor shows you the events available from all of the components

---

<sup>1</sup>From within TIBCO® General Interface Builder, the iProcess Workspace (Browser) components are referred as “WCC” components, which stands for “Workspace Client Components”.

that have been added to the application. (It also allows you to import event configurations from external applications, and subscribe to those.)

For information about using the Events Editor, see [Events Editor](#).

For more information about the TIBCO PageBus, see the *TIBCO PageBus™ Developer's Guide*.

## Loading PageBus

WCC applications automatically load the TIBCO PageBus. However, it first looks to see if PageBus has already been loaded by another application on the system. If it is already loaded, the WCC application does not load it again.

Note, however, that WCC applications are tied to a specific version of PageBus. If the PageBus has already been loaded by another application, and it is not compatible with the WCC application, it may result in errors.

You can determine the version of PageBus required by the WCC application by looking in the "System Requirements" section in the *TIBCO iProcess Workspace (Browser) Installation Guide*.

## Publishing and Subscribing to Events

The following shows the ways in which you might want to communicate between components in your application using events:

- **WCC component <—> WCC component** - When communicating between WCC components, the Events Editor is used to set up the connection between the components. There is no coding required. When you add a WCC component to your application, the events available from that component are automatically pushed to the PageBus. When you add another WCC component, the Events Editor shows you all of the events that are available to which that component can subscribe.

For information about using the Events Editor, see [Events Editor](#).

- **WCC component <—> WCC component in external application** - WCC components can also subscribe to events published by other WCC components that reside in an external application. This involves creating an event definition file (using the Application Publish Definer utility — see [Events From External Applications](#)) that defines the events published by the WCC components in the external application.

The event definition file can then be imported (using the Events Editor) so that a WCC component can subscribe to the events from the external application.

For more information, see [Events From External Applications](#).

- **non-WCC component <—> WCC component** - When communicating between a non-WCC component and a WCC component, you will need to use the PageBus API to either publish to, or subscribe from, the PageBus, as follows:

- **non-WCC component publishing an event** - Non-WCC components can publish events to the PageBus using the **PageBus.publish** method. WCC components can then subscribe to those events.

For more information, see [Non-WCC Components Publishing Events](#).

- **non-WCC component subscribing to an event** - Non-WCC components can subscribe to events on the PageBus that have been published by WCC components. This is done using the **PageBus.subscribe** method.

For more information, see [Non-WCC Components Subscribing to Events](#).

- **non-WCC component <—> non-WCC component** - When communicating between non-WCC components, you must use the PageBus **publish** and **subscribe** methods to publish to, and subscribe from, the PageBus. For more information, see the *TIBCO PageBus Developer's Guide*.

## Non-WCC Components Publishing Events

Non-WCC components can publish messages (events) to the PageBus using the **PageBus.publish** method. WCC components can then subscribe to those messages.

Messages published by non-WCC components must conform to the schemas expected by the WCC components. The schemas expected by the WCC components are listed in [Schemas Expected by WCC Components](#). The schemas themselves are described in [WCC Message Schemas](#).

The **PageBus.publish** method passes the following parameters:

Name	Type	Description
section	string	section name on which to publish the message. This must not be a wildcard subject.

Name	Type	Description
message	any	Message content. This data value SHOULD be serializable as JSON. If the event cache is used, this data value MUST be serializable as JSON.

For WCC components, the section has the form:

```
com.tibco.wcc.appModelName.prototypeModelName.componentModelName.eventName
```

The *message* is a JavaScript object and has the form:

```
{
  jss : "org.pagebus.msg.Message",
  jssv : "2.0.0",
  head : {section : eventName},
  body : bodyObject
}
```

where:

- *bodyObject* is a JavaScript object that contains the data (typically an array of tags) expected by the WCC component that will subscribe to that event. The *bodyObject* must conform to one of the WCC message schemas listed below, depending on the type of data being passed by the **publish** method.

For example, if a non-WCC component publishes a message that passes a *bodyObject* that conforms to the WCC message schema, **com.tibco.wcc.schema.cases**, only WCC components that expect that schema can subscribe to that event.

For more details about using the **publish** method, see the *TIBCO PageBus™ Developer's Guide*.

## Non-WCC Components Subscribing to Events

Non-WCC components can subscribe to messages (events) that have been published to the PageBus by using the **PageBus.subscribe** method.

Messages received by the subscribing non-WCC component will conform to specific WCC message schemas. The schemas published by each WCC component event is listed in



[Schemas Published by WCC Component Events](#). For information about each of the schemas, see [WCC Message Schemas](#).

The **PageBus.subscribe** method passes the following parameters:

Name	Type	Description
section	string	section name on which to subscribe. This may be a wildcard section.
scope	object	<p>If the value is non-null, the object specified here becomes the context of the callback function. In other words, when the JavaScript <code>this</code> keyword is used in the callback function, it will point to this object.</p> <p>If the value is null, then the object window is used as the default context of the callback function.</p>
onData	function	<p>Callback function for PageBus to invoke when a message is published on the subscribed subject.</p> <p>This parameter must not be null.</p>
subscriberData	any	<p>User-defined. Null values are permitted. This is useful when the subscriber needs to access or update any data members when the callback function is invoked.</p> <p>The value of the subscriberData parameter determines whether the subscriber is cache-enabled.</p>

For WCC components, the section has the form:

```
com.tibco.wcc.appModelName.prototypeModelName.componentModelName.eventName
```

For more details about using the **subscribe** method, see the *TIBCO PageBus™ Developer's Guide*

## Schemas Expected by WCC Components

The following lists the schemas expected by each of the WCC components (note that 'schemas' in this context refers to the structure of the JavaScript object that is passed in

the **publish** method, rather than an XML schema). These are defined in:

```
JSXAPPS\wcc\components\ComponentType\defs\ComponentName.hand.xml
```

where *ComponentType* describes the type of component (e.g., Composite, ListContainer, etc.), and *ComponentName* is the name of the component (e.g., CasesPreview, WorkItems, etc.).

These are the schemas needed by the component to be able to render. For instance, the following example ...\*Cases.hand.xml* file specifies that the *com.tibco.wcc.schema.procs* schema is needed by the component to render:



**Note:** Components that expect a *com.tibco.wcc.schema.empty* schema do not need data to render, although there must have been a previous login because they need valid login credentials.

```
<component name="CasesPreview" type="com.tibco.jsp.wcc.CasesPreview">
  <handler name="render" schemaId="com.tibco.wcc.schema.procs"/>
</component>
```

For a description of the schemas, see [WCC Message Schemas](#).

## Application

- **DataMask** - None

## Authentication

- **Login** - None
- **Logout** - None

## Case Management

- **CaseData** - [com.tibco.wcc.schema.cases](#)
- **CaseDetail** - [com.tibco.wcc.schema.cases](#)
- **CaseHistory** - [com.tibco.wcc.schema.cases](#)
- **CaseOutstanding** - [com.tibco.wcc.schema.cases](#)
- **CaseSummary** - [com.tibco.wcc.schema.cases](#)

## Composites

- **CasesPreview** - [com.tibco.wcc.schema.procs](#)
- **ProcsCases** - com.tibco.wcc.schema.empty
- **WorkItemsPreview** - [com.tibco.wcc.schema.workQs](#)
- **WorkQsItems** - com.tibco.wcc.schema.empty

## Lists

- **Cases**
  - **Cases** - [com.tibco.wcc.schema.procs](#)
- **Procedures**
  - **HistoryProcs** - com.tibco.wcc.schema.empty
  - **NodeProcs** - com.tibco.wcc.schema.empty
  - **StartHistoryProcs** - com.tibco.wcc.schema.empty
  - **StartProcs** - com.tibco.wcc.schema.empty
- **Work Items**
  - **WorkItems** - [com.tibco.wcc.schema.workQs](#)
- **Work Queues**
  - **UserWorkQs** - com.tibco.wcc.schema.empty

## Schemas Published by WCC Component Events

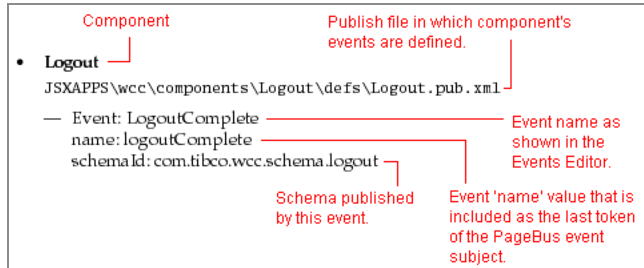
This section lists the message schema published by each of the events on the WCC components. These are defined in the files:

```
JSXAPPS\wcc\components\ComponentType\defs\ComponentName.pub.xml
```

where *ComponentType* describes the type of component (e.g., Case or ListContainer), and *ComponentName* is the name of the component (e.g., CaseHistory or WorkItems). Note that *ComponentName* does not always exactly match the component name shown in TIBCO® General Interface Builder — for example, the CaseDetail component's events are defined in the Case.pub.xml file. The name of the publish file for each component is shown in the list below.

The name shown for each event below is the actual event 'name' value that is included as the last token of the PageBus event subject.

For each component event, the following is shown in the lists below:



For a description of the schemas, see [WCC Message Schemas](#).

## Application

- **DataMask**
  - No events

## Authentication

- **Login**
  - JSXAPPS\wcc\components\Login\defs\Login.pub.xml
  - Event: LoginComplete
    - name: loginComplete
    - schemald: [com.tibco.wcc.schema.login](#)
- **Logout**
  - JSXAPPS\wcc\components\Logout\defs\Logout.pub.xml
  - Event: LogoutComplete
    - name: logoutComplete
    - schemald: [com.tibco.wcc.schema.logout](#)

## Case Management

- **CaseData**
  - JSXAPPS\wcc\components\Case\defs\CaseData.pub.xml

- Event: Case Data Update  
name: caseDataUpdate  
schemald: com.tibco.wcc.schema.empty

- **CaseDetail**

JSXAPPS\wcc\components\Case\defs\Case.pub.xml

- Event: Process Jump  
name: jumpCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Trigger Event  
name: eventsCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Purge Case  
name: purgeCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Close Case  
name: closeCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Suspend Case  
name: suspendCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Activate Case  
name: activateCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: View Graphical Case History  
name: viewGAT  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Case Prediction  
name: predictCase  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Add History  
name: addHistory  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Select Case History (single click)  
name: selectCaseHistory  
schemald: [com.tibco.wcc.schema.casehistory](#)

- Event: Execute Case History (double click)  
name: executeCaseHistory  
schemald: [com.tibco.wcc.schema.casehistory](#)
- Event: Select Case Outstanding (single click)  
name: selectCaseOutstanding  
schemald: [com.tibco.wcc.schema.caseoutstanding](#)
- Event: Execute Case Outstanding (double click)  
name: executeCaseOutstanding  
schemald: [com.tibco.wcc.schema.caseoutstanding](#)
- Event: Open Case Outstanding Work Item(s)  
name: openOutstandingItems  
schemald: [com.tibco.wcc.schema.caseoutstanding](#)
- Event: Case Data Update  
name: caseDataUpdate  
schemald: com.tibco.wcc.schema.empty

- **CaseHistory**

JSXAPPS\wcc\components\Case\defs\CaseHistory.pub.xml

- Event: View Graphical Case History  
name: viewGAT  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Case Prediction  
name: predictCase  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Add History  
name: addHistory  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Select Case History (single click)  
name: selectCaseHistory  
schemald: [com.tibco.wcc.schema.casehistory](#)
- Event: Execute Case History (double click)  
name: executeCaseHistory  
schemald: [com.tibco.wcc.schema.casehistory](#)

- **CaseOutstanding**

JSXAPPS\wcc\components\Case\defs\CaseOutstanding.pub.xml

- Event: Select Case History (single click)  
name: selectCaseOutstanding  
schemald: [com.tibco.wcc.schema.caseoutstanding](#)
- Event: Execute Case History (double click)  
name: executeCaseOutstanding  
schemald: [com.tibco.wcc.schema.caseoutstanding](#)
- Event: Open Case Outstanding Work Item(s)  
name: openOutstandingItems  
schemald: [com.tibco.wcc.schema.caseoutstanding](#)

- **CaseSummary**

JSXAPPS\wcc\components\Case\defs\CaseSummary.pub.xml

- Event: Process Jump  
name: jumpCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Trigger Event  
name: eventsCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Purge Case  
name: purgeCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Close Case  
name: closeCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Suspend Case  
name: suspendCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Activate Case  
name: activateCases  
schemald: [com.tibco.wcc.schema.cases](#)

## Composites

- **CasesPreview**

- No events

- **ProcsCases**

- No events
- **WorkItemsPreview**
  - No events
- **WorkQsItems**
  - No events

## Lists / Cases

- **Cases**

JSXAPPS\wcc\components\ListContainer\defs\Cases.pub.xml

  - Event: List Item Select (single click)
    - name: listItemSelect
    - schemald: [com.tibco.wcc.schema.cases](#)
  - Event: List Item Execute (double click)
    - name: listExecute
    - schemald: [com.tibco.wcc.schema.cases](#)
  - Event: Process Jump
    - name: jumpCases
    - schemald: [com.tibco.wcc.schema.cases](#)
  - Event: Trigger Event
    - name: eventsCases
    - schemald: [com.tibco.wcc.schema.cases](#)
  - Event: Purge Case
    - name: purgeCases
    - schemald: [com.tibco.wcc.schema.cases](#)
  - Event: Close Case
    - name: closeCases
    - schemald: [com.tibco.wcc.schema.cases](#)
  - Event: Suspend Case
    - name: suspendCases
    - schemald: [com.tibco.wcc.schema.cases](#)
  - Event: Activate Case
    - name: activateCases
    - schemald: [com.tibco.wcc.schema.cases](#)



- Event: Open Case  
name: openCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: View Graphical Case History  
name: viewGAT  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Case Prediction  
name: predictCase  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Add History  
name: addHistory  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Select Case History (single click)  
name: selectCaseHistory  
schemald: [com.tibco.wcc.schema.casehistory](#)
- Event: Execute Case History (double click)  
name: executeCaseHistory  
schemald: [com.tibco.wcc.schema.casehistory](#)
- Event: Select Case Outstanding (single click)  
name: selectCaseOutstanding  
schemald: [com.tibco.wcc.schema.caseoutstanding](#)
- Event: Execute Case Outstanding (double click)  
name: executeCaseOutstanding  
schemald: [com.tibco.wcc.schema.caseoutstanding](#)
- Event: Open Case Outstanding Work Item(s)  
name: openOutstandingItems  
schemald: [com.tibco.wcc.schema.caseoutstanding](#)
- Event: Case Data Update  
name: caseDataUpdate  
schemald: com.tibco.wcc.schema.empty

## Lists / Procedures

- **NodeProcs, HistoryProcs, StartProcs, and StartHistoryProcs**  
JSXAPPS\wcc\components>ListContainer\defs\Procs.pub.xml

- Event: List Item Select (single click)  
name: listItemSelect  
schemald: [com.tibco.wcc.schema.procs](#)
- Event: List Item Execute (double click)  
name: listExecute  
schemald: [com.tibco.wcc.schema.procs](#)
- Event: Start Case  
name: startCase  
schemald: [com.tibco.wcc.schema.procs](#)
- Event: Procedure Versions  
name: procVersions  
schemald: [com.tibco.wcc.schema.procs](#)
- Event: Loading Chart  
name: loadLoadingChart  
schemald: com.tibco.wcc.schema.empty

## Lists/ Work Items

- **WorkItems**

JSXAPPS\wcc\components\ListContainer\defs\WorkItems.pub.xml

- Event: List Item Select (single click)  
name: listItemSelect  
schemald: [com.tibco.wcc.schema.workItems](#)
- Event: List Item Execute (double click)  
name: listExecute  
schemald: [com.tibco.wcc.schema.workItems](#)
- Event: Open Work Item(s)  
name: openItems  
schemald: [com.tibco.wcc.schema.workItems](#)
- Event: Forward Work Item(s)  
name: forwardItems  
schemald: [com.tibco.wcc.schema.workItems](#)
- Event: Unlock Work Item(s)  
name: unlockItems  
schemald: [com.tibco.wcc.schema.workItems](#)

- Event: Release Work Item(s)  
name: releaseItems  
schemald: [com.tibco.wcc.schema.workItems](#)
- Event: Open Case  
name: openCase  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Process Jump  
name: jumpCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Trigger Event  
name: eventsCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Purge Case  
name: purgeCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Close Case  
name: closeCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Suspend Case  
name: suspendCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Activate Case  
name: activateCases  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: View Graphical Case History  
name: viewGAT  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Case Prediction  
name: predictCase  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Add History  
name: addHistory  
schemald: [com.tibco.wcc.schema.cases](#)
- Event: Select Case History (single click)  
name: selectCaseHistory  
schemald: [com.tibco.wcc.schema.casehistory](#)

- Event: Execute Case History (double click)  
name: executeCaseHistory  
schemald: [com.tibco.wcc.schema.casehistory](#)
- Event: Select Case Outstanding (single click)  
name: selectCaseOutstanding  
schemald: [com.tibco.wcc.schema.caseoutstanding](#)
- Event: Execute Case Outstanding (double click)  
name: executeCaseOutstanding  
schemald: [com.tibco.wcc.schema.caseoutstanding](#)
- Event: Open Case Outstanding Work Item(s)  
name: openOutstandingItems  
schemald: [com.tibco.wcc.schema.caseoutstanding](#)
- Event: Case Data Update  
name: caseDataUpdate  
schemald: com.tibco.wcc.schema.empty

## Lists / Work Queues

- **UserWorkQs**

JSXAPPS\wcc\components>ListContainer\defs\WorkQs.pub.xml

- Event: List Item Select (single click)  
name: listItemSelect  
schemald: [com.tibco.wcc.schema.workQs](#)
- Event: List Item Execute (double click)  
name: listExecute  
schemald: [com.tibco.wcc.schema.workQs](#)
- Event: Loading Chart  
name: loadLoadingChart  
schemald: com.tibco.wcc.schema.empty
- Event: Participation  
name: loadParticipation  
schemald: com.tibco.wcc.schema.empty
- Event: Redirection  
name: loadRedirection  
schemald: com.tibco.wcc.schema.empty

- Event: Supervisors  
name: loadSupervisors  
schemaId: com.tibco.wcc.schema.empty

## WCC Message Schemas

Each of the WCC message schemas is defined below.



### Note:

Most of the message schemas expect a *tag* of some sort (e.g., case tag, work item tag, etc.). Tags are intentionally opaque, that is, we do not provide the information needed to build them — you are expected to acquire them in one of several ways:

- The most likely scenario is that you would get the tag using one of the WCC components, save it somewhere, then use it at a later time.
- Tags can be acquired through the iProcess Server Objects object model, specifically using the **getTag** and **makeTag** methods. For information, see the *TIBCO iProcess Server Objects (Java or .NET) Programmer's Guide*.
- You can also acquire tags through an Action Processor response XML. For information, see the *TIBCO iProcess Workspace (Browser) Action Processor Reference*.

### com.tibco.wcc.schema.cases

```
{
  schemaId : 'com.tibco.wcc.schema.cases',
  tags      : Array<String>
}
```

where:

- *tags* <String> is a case tag.

### com.tibco.wcc.schema.procs

```
{
  schemaId : 'com.tibco.wcc.schema.procs',
  tags      : Array<String>
}
```

```

    desc      : Array<String>
  }

```

where:

- *tags* <String> is a procedure tag.
- *descs* <String> is a description corresponding to each of the procedures identified in the tags array. These descriptions display in the case list title bar if the **captionCases** attribute, in `config.xml`, is set to “description”.

## com.tibco.wcc.schema.workItems

```

{
  schemaId : 'com.tibco.wcc.schema.workItems',
  tags     : Array<String>
}

```

where:

- *tags* <String> is a work item tag.

## com.tibco.wcc.schema.workQs

```

{
  schemaId : 'com.tibco.wcc.schema.workQs',
  tags     : Array<String>
  desc     : Array<String>
}

```

where:

- *tags* <String> is a work queue tag.
- *descs* <String> is a description corresponding to each of the work queues identified in the tags array. These descriptions display in the work item list title bar if the **captionWorkItems** attribute, in `config.xml`, is set to "description".

## com.tibco.wcc.schema.login

```
{
  schemaId : 'com.tibco.wcc.schema.login',
  user      : <String>,
  server    : <String>
}
```

where:

- *user* <String> is the logged-in username.
- *server* <String> is the iProcess Objects Server name.

## com.tibco.wcc.schema.logout

```
{
  schemaId : 'com.tibco.wcc.schema.logout',
  user      : <String>,
  server    : <String>
}
```

where:

- *user* <String> is the logged-in username.
- *server* <String> is the iProcess Objects Server name.

## com.tibco.wcc.schema.casehistory

```
{
  schemaId : 'com.tibco.wcc.schema.casehistory'
  tags      : Array<String>
  records   : [{
    Action : <String>
    Date   : <String>
    Description : <String>
    Message : <String>
    IsOutstanding : <String>
    Name    : <String>
    TimeOffset : <String>
    User     : <String>
    SubCaseId : <String>
    ProcMajorVersion : <String>
  }
}
```

```

        ProcMinorVersion : <String>
      }
    ]
  }

```

where:

- *tags* <String> is a case tag.
- *Action* <String> is the case history action, enumerated in SWAuditActionType, e.g., 'swProcessedTo'.
- *Date* <String> is the date the action occurred, e.g., '2006-06-30T09:57:24.0000000'.
- *Description* <String> is the description of the step from which the action occurred, e.g., 'First step'.
- *Message* <String> is the case history message from the **audit.mes** file, e.g., "First step" processed to swadmin@myserver'.
- *IsOutstanding* <String> specifies whether or not the audit step has been processed, e.g., 'false'.
- *Name* <String> is the name of the step performing the action, e.g., 'STEP1'.
- *TimeOffset* <String> is a microsecond offset of the action time, e.g., '994563'.
- *User* <String> is the user performing the action, e.g., 'swadmin@myserver'.
- *SubCaseId* <String> identifies the sub-case the audit step is from, e.g., "".
- *ProcMajorVersion* <String> is the major version number of the procedure, e.g., '0'.
- *ProcMinorVersion* <String> is the minor version number of the procedure, e.g., '2'.

## com.tibco.wcc.schema.caseoutstanding

```

{
  schemaId : 'com.tibco.wcc.schema.caseoutstanding'
  tags      : Array<String>
  records   : [{
    Version : <String>
    Workq    : <String>
    Step     : <String>
    Type     : <String>
    Path     : <String>
  }]
}

```



```

        Proc : <String>
        Arrived : <String>
        Dead : <String>
        IsDeadline : <String>
        IsDeadlineExp : <String>
        IsForwardable : <String>
        IsLocked : <String>
        IsLongLocked : <String>
        IsReleasable : <String>
        IsUnopened : <String>
        IsUrgent : <String>
        IsSuspended : <String>
        LockedBy : <String>
    }
]
}

```

where:

- *tags* <String> is a case tag.
- *Version* <String> is the major and minor version numbers of the procedure, e.g., '0.2'.
- *Workq* <String> is the work queue in which the outstanding step resides, e.g., 'swadmin'.
- *Step* <String> is the name of the outstanding step, e.g., 'SUMMARY'.
- *Type* <String> is the type of the outstanding step, e.g., 'Normal'.
- *Path* <String> is the path to the outstanding step, e.g., 'SUMMARY'.
- *Proc* <String> is the name of the procedure, e.g., 'ALLOCATE'.
- *Arrived* <String> is the date and time the outstanding step arrived in the work queue, e.g., '2007-08-23 17:11'.
- *Dead* <String> is the date and time of the deadline on the outstanding step, e.g., '2007-08-30 20:00'.
- *IsDeadline* <String> identifies if there is a deadline specified, e.g., 'true'.
- *IsDeadlineExp* <String> identifies if the outstanding step has an expired deadline, e.g., 'false'.
- *IsForwardable* <String> identifies if the step is forwardable, e.g., 'false'.
- *IsLocked* <String> identifies if the step/work item is locked, e.g., 'true'.

- *IsLongLocked* <String> identifies if the step/work item is long locked, e.g., 'false'.
- *IsReleasble* <String> identifies if the step is releaseable, e.g., 'false'.
- *IsUnopened* <String> identifies if the step is unopened, e.g., 'false'.
- *IsUrgent* <String> identifies if the step is urgent, e.g., 'false'.
- *IsSuspended* <String> identifies if the step is in a suspended case, e.g., 'false'.

*LockedBy* <String> is the name of the user who has the step/work item locked, e.g., 'susieq'.

# Events Editor

---

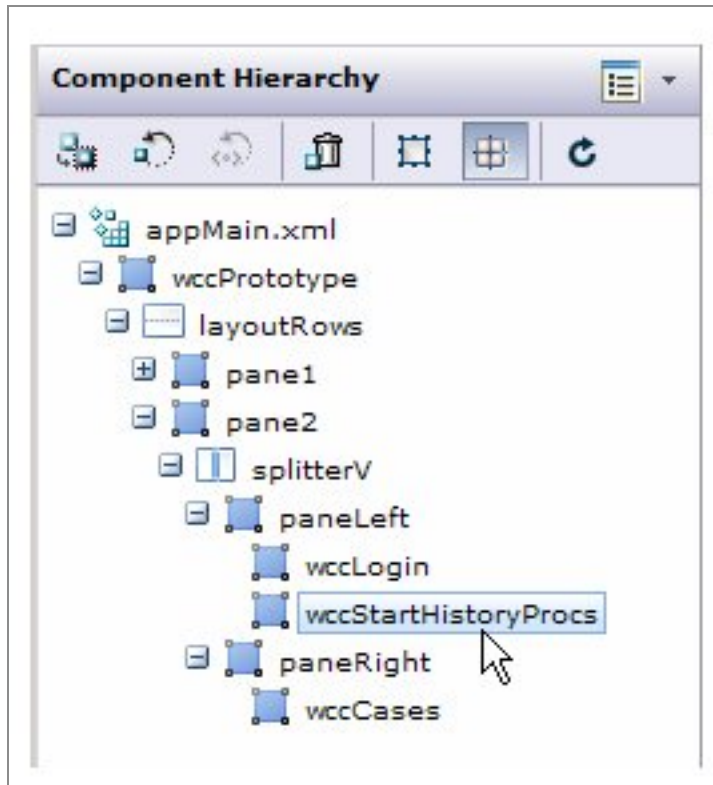
This chapter describes how events are handled in the TIBCO iProcess Workspace (Browser) components.

## Using the Events Editor

The Events Editor is used to specify the PageBus events to which you would like a WCC component to subscribe. (For information about the PageBus, see [Communications Overview](#).)

Opening the Events Editor causes it to query all of the TIBCO iProcess Workspace (Browser) components in the application to determine their public events, and present them in a dialog so that you can choose which to subscribe to. (All public events in the application are shown except for those published by the component you are configuring — it would not make sense for a component to subscribe to its own events.)

To subscribe to events for a particular component, in the **Component Hierarchy** palette, click on the component that will be subscribing to events published by other components:



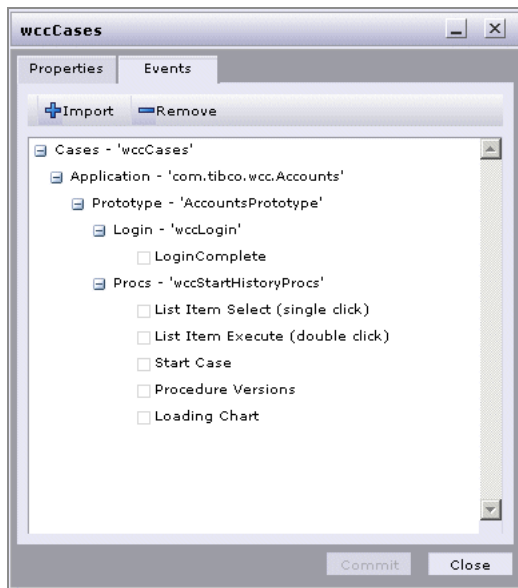
If the Properties/Events Editor is already displayed when you click on the component, it will now show the properties/events for the component you selected in the **Component Hierarchy** palette.

If the Properties/Events Editor is not currently displayed when you click on the component, a button appears in the TIBCO® General Interface Builder taskbar that, when clicked, displays the Properties/Events Editor:



If the Properties/Events Editor is not displayed on the taskbar, double click on a component in the **Component Hierarchy** palette to open the editor.

Click the **Events** tab to display the Events Editor. A dialog similar to the following is displayed.



This dialog presents all of the events published by all of the components currently defined in the application (with the exception of the component you are currently configuring). The following describes the information presented on this dialog:

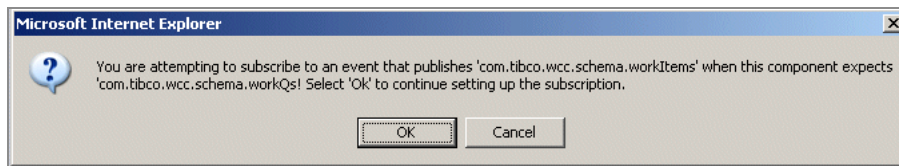
- The first line identifies the component type for which you are configuring subscriptions — “Cases” in this example, followed by the name of the particular component — “wccCases” in this example. Note that the component name is always preceded by “wcc” in the Events Editor (“wcc” means “Workspace Client Component”).
- The second line provides the fully qualified application/project name.
- The third line provides the name of the prototype.
- The subsequent information represents components that have been defined in the application, followed by the events that each component is publishing to the PageBus. That is, they are the events that the component you are configuring can subscribe to.

In the example above, the **wccCases** component (the one we are configuring) can subscribe to events published by the **wccLogin** and **wccStartHistoryProcs** components — they are the other components that have been added to the application so far. The **wccLogin** component is publishing one event to which the **wccCases** component can subscribe; the **wccStartHistoryProcs** component is publishing five different events to which the **wccCases** component can subscribe.

Note that although all events published to the PageBus are shown in the Events Editor, it may not make sense to subscribe to some events in the context of the component you are

configuring. For instance, when configuring the **wccCases** component, it does not make sense to subscribe to the “Login Complete” event on the **wccLogin** component. It does make sense, however, to subscribe to the “List Item Select (single click)” event on the **wccStartHistoryProcs** component to display the case list for the procedure that was selected.

When you attempt to subscribe to an event, the Event Editor verifies via schema files that the data that will be passed by the event is the data that the component expects. For example, if the component subscribing to the event expects a work queue tag, but the schema indicates the event is going to pass a work item tag, the Event Editor will warn you by displaying the following:



The editor will allow you to continue with the subscription if that is what you want.

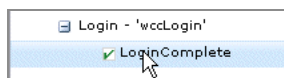
For more information about the individual events for each component, see the *TIBCO iProcess Workspace (Browser) Components Reference*.

## Subscribing to Events

The check box to the left of each event name in the Event Editor indicates whether or not the component you are configuring will subscribe to that event:

- If the check box is checked, the component subscribes to that event.
- If the check box is not checked, the component does not subscribe to the event.

To specify that you want the component to subscribe to an event, double-click on the desired event name. This causes the event’s check box to become checked:



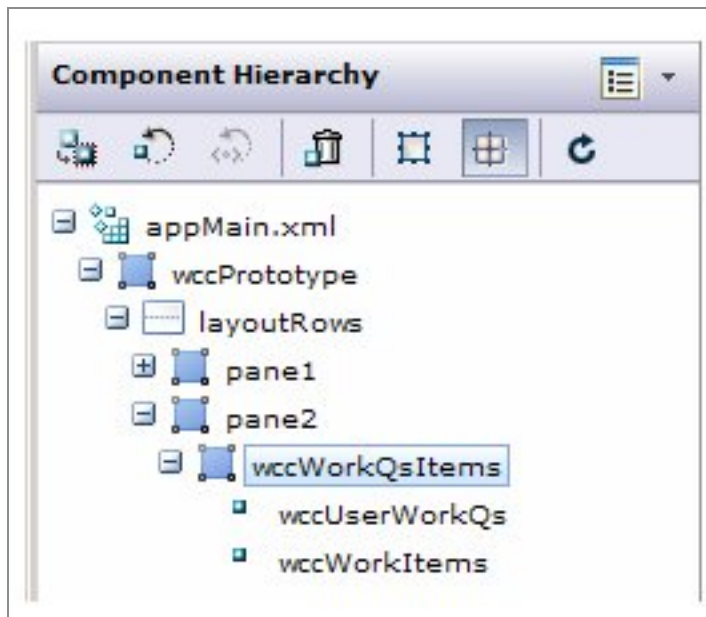
To unsubscribe to an event, double-click the event name again to remove the check mark.

After checking the appropriate event check boxes in the Events Editor, commit your changes by clicking the **Commit** button. This immediately applies the changes you’ve made.

## Composite Components Events

There are composite components available that combine more than one component into a single component to provide an easy way to display multiple lists of elements without the need to add multiple components to your project and set up the events between those components — the events between the components provided in the composites are already configured.

For example, the composite component **wccWorkQsItems** includes the **wccUserWorkQs** and **wccWorkItems** components:



With this composite component, the **wccWorkItems** component automatically subscribes to the “List Item Select (single click)” event on the **wccUserWorkQs** component — you do not need to configure any events between those components. You will only need to specify the event to which the **wccWorkQsItems** composite component will subscribe (which is likely the “Login Complete” event on the **wccLogin** component).

In fact, if you select a composite’s underlying component in the Component Hierarchy, then display the Properties/Event Editor, the **Events** tab is disabled, as those components cannot subscribe to events. (Note, however, other components may subscribe to the events published by the underlying components.)

## Events From External Applications

Because TIBCO iProcess Workspace (Browser) components need to communicate with applications that exist outside of their own application, the Events Editor provides an *Import* function that is used to import *event definition* files that contain the definitions of events in external applications. This allows you to see, and subscribe to, events in the Events Editor that are being published by external applications.

TIBCO® General Interface Builder provides an *Application Publish Definer* that is used to create an event definition file from within the external application. You can then import that file into another WCC application and subscribe to the events published by the external application.

Event definition files must conform to the schema defined in the file:

```
WorkspaceDir\JSXAPPS\WCCProjectName\defs\app.pub.schema.xsd
```

where *WorkspaceDir* is the workspace directory you specified after starting TIBCO® General Interface Builder, and *WCCProjectName* is the name you gave the WCC Project when creating your custom application with the iProcess Workspace (Browser) components.

The following subsections describe the steps necessary to create the event definition file, then import that file into another WCC application to it can subscribe to the events defined in that file.

## Creating the External Application Event Definition

1. With the external application open in TIBCO® General Interface Builder, select **Project Settings** from the **Project** menu.

The **Project Settings** dialog is displayed.

2. Click the **WCC Settings** button.
3. Click the **Open Application Publish Definer** button.

The **Create new WCC Application File** dialog is displayed. This dialog allows you to enter an application model name that will be used in the name of the event definition file, as follows:

```
WCCProjectName.app.pub.xml
```

where *WCCProjectName* is the name of the current WCC project/application.



It defaults to using the application model name for your current application.

4. Click the **Create** button.

A dialog is displayed that allows you to point to the prototype file for your external application.

5. Click the **Include** button.

The **Include Files** window is displayed.

6. Navigate to, and select, the prototype file for your external application. For example:

*WorkspaceDir\JSXAPPS\WCCProjectName\application\prototypes\appMain.xml*

where *WorkspaceDir* is the workspace directory you specified after starting TIBCO® General Interface Builder, and *WCCProjectName* is the name you gave the WCC Project when creating your application with the iProcess Workspace (Browser) components.

7. From the **Include Files** dialog, click the **Include** button.

8. Click the **Save** button to save the event definition file, then click **Close** to close the dialog.

9. Click the **Cancel** button to close the **Project Settings** dialog.

The new event definition file is saved in the following location:

*WorkspaceDir\JSXAPPS\WCCProjectName\defs\WCCProjectName.app.pub.xml*

where *WorkspaceDir* is the workspace directory you specified after starting TIBCO® General Interface Builder, and *WCCProjectName* is the name you gave the WCC Project when creating your custom application with the iProcess Workspace (Browser) components.

The new event definition file can now be imported into another WCC application.

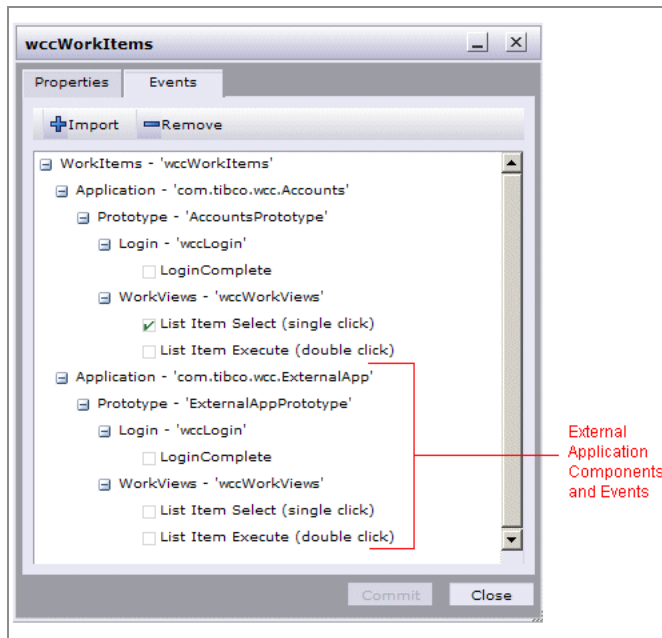
## Importing the Event Definition File

1. Open the custom WCC application from which you want to subscribe to events from an external application.
2. Display the Events Editor (see [Using the Events Editor](#)).
3. Click the **Import** button in the Events Editor.

The **Import Files** dialog is displayed.

4. Navigate to the event definition file that was created in the external application (see [Step 6](#) in the [Creating the External Application Event Definition](#)).
5. Select the event definition file, then click **Import**.

The Events Editor will now include the components and events from the external application. In the following example, the external application (named “ExternalApp”) contains two components whose events your custom application can subscribe:



6. Subscribe to the desired events from the external application.

Note - The external application’s components / events can be removed from the Events Editor by clicking on the **Remove** button. This displays the **Remove Files** dialog from which you can select the desired event definition file for removal.

# Properties Editor

---

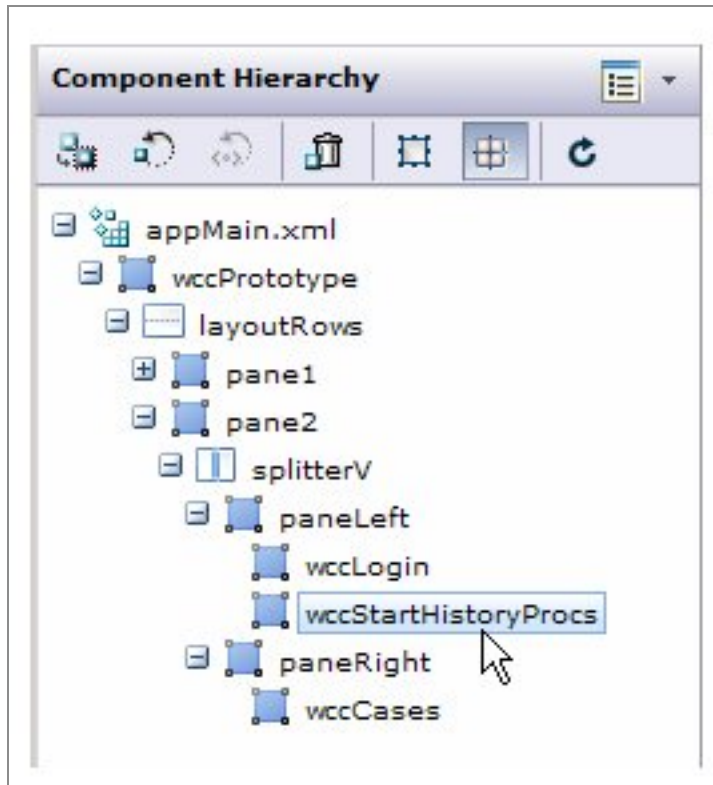
This chapter describes how to configure properties available through the TIBCO iProcess Workspace (Browser) components.

## Using the Properties Editor

The Properties Editor is used to specify property definitions for TIBCO iProcess Workspace (Browser) components.

The primary purpose of component properties is to specify how access to functionality exposed by the component is handled. This is done in conjunction with the user access profiles defined in an iProcess Workspace (Browser) user access profiles configuration file, `userAccessProfiles.xml`. User access profiles control access in the application based on the logged-in user's *type*. For information about user access profiles, see *TIBCO iProcess Workspace (Browser) Configuration and Customization*.

To edit the properties for a particular component, click on the component in the **Component Hierarchy** palette.



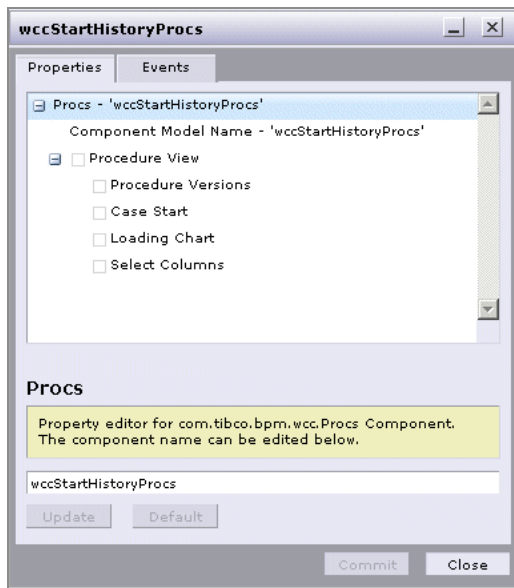
If the Properties/Event Editor is already displayed when you click on the component, it will now show the properties/events for the component you selected in the **Component Hierarchy** palette.

If the Properties/Events Editor is not currently displayed when you click on the component, a button appears in the TIBCO® General Interface Builder taskbar that, when clicked, displays the Properties/Events Editor:



If the Properties/Events Editor is not displayed on the taskbar, double click on a component in the **Component Hierarchy** palette to open the editor.

Click the **Properties** tab to display the Properties Editor.



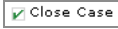
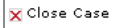
The Properties Editor presents the following:

- The first line identifies the component type for which you are configuring properties — “Procs” in this example, which represents a procedures list. Following the component type is the name of the component. Note that the component name is always preceded by “wcc” in the Properties Editor (“wcc” means “Workspace Client Component”).
- The second line identifies the component model name. For more information, see [Component Names](#).
- The subsequent information is a hierarchical tree view of all properties for the component. Each property represents a piece of functionality that is exposed by the component.

The state of the check box for each property specifies how access to that functionality is handled, as follows:

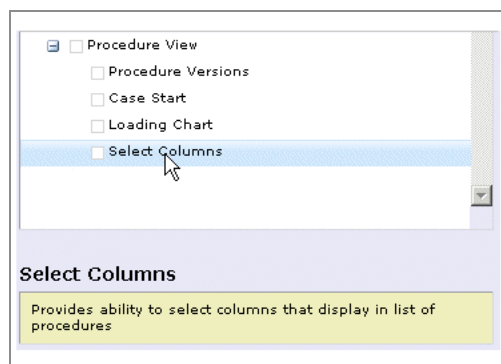
State	Description
Unchecked <input type="checkbox"/> Close Case	Access to the functionality (the <b>Close Case</b> button/menu selection in this example) is controlled totally through the user access profiles <sup>1</sup> .

<sup>1</sup>For information about user access profiles, see *TIBCO iProcess Workspace (Browser) Configuration and Customization*.

State	Description
Checked 	Access to the functionality is <i>always</i> allowed, regardless of the setting in the user access profiles.
Disabled 	Access to the functionality is <i>never</i> allowed, regardless of the setting in the user access profiles.  Note that when you disable access to a parent property in the Properties Editor, all child properties of that parent are also disabled.

You can change the state of the property check boxes by double-clicking on the property name/check box. It will cycle through the different states — unchecked, checked, and disabled.

Clicking on the property name causes the field below the property list to display a brief description of the functionality defined by that property. For example:



For more information about the specific functionality available for each component, see the *TIBCO iProcess Workspace (Browser) Components Reference*.

## Component Names

Each WCC component has both a name and a model name. The model name is used in the subject of a PageBus event. The PageBus subject consists of a number of 'tokens' separated by a dot '.' character. For example:

```
com.tibco.wcc.Accounts.AccountsPrototype.wccStartHistoryProcs.listItemSelect
```

The tokens are in the following order:

- organization prefix (usually three tokens: always **com.tibco.wcc** for WCC component events)
- application model name (e.g., Accounts)
- prototype model name (e.g., AccountsPrototype)
- component model name (e.g., wccStartHistoryProcs)
- event name (e.g., listItemSelect)

By default, the name and the model name will initially be the same. There are times, however, where it make sense to treat several components with unique names as the same source for subscribing to their PageBus events.

For example, a prototype might contain two procedure list components, where each list component has a unique name. The developer wants the subscribing method to react to events from both components in exactly the same way using a single PageBus subscription. This is accomplished by setting the model name of both components to the same name.

This same concept can be applied to the application model name and the prototype model name.

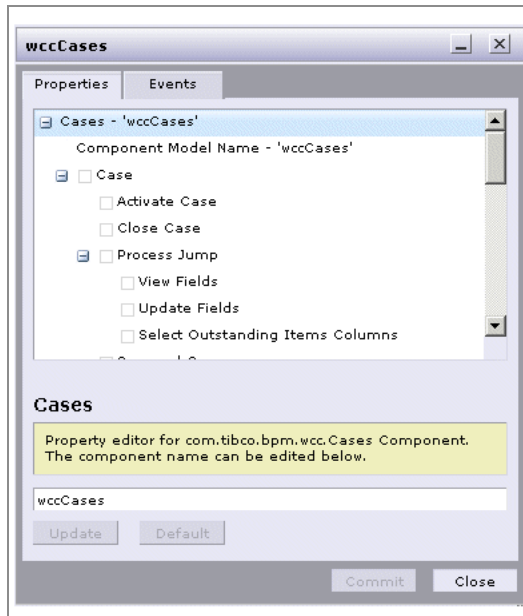
For information about the PageBus, see [Communications Overview](#).

## Changing Component/Model Names

The Properties Editor can be used to change both the component name and the component model name.

### Changing Component Name

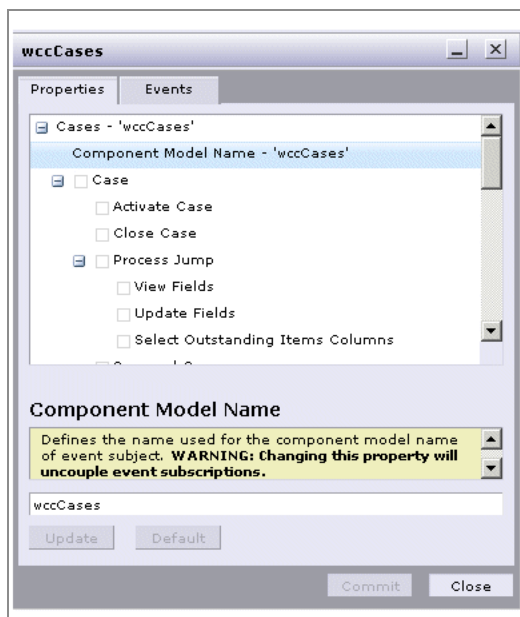
To change the component name, display the Properties Editor and highlight the first line, which shows the component name:



After making a change to the component name, the **Update** and **Default** buttons become active, allowing you to either save your changes, or to revert back to the default name for the component. The **Commit** button must also be clicked to save all changes made in the Properties Editor.

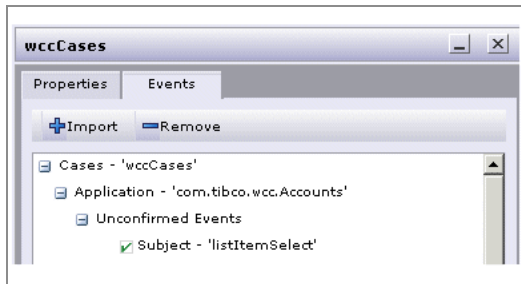
## Changing Component Model Name

To change the component model name, display the Properties Editor and highlight the second line, which shows the component model name:





Note the warning that any events that have already been subscribed to using the model name you are changing will now be uncoupled from the component that had subscribed to it. For instance, if you change a component model name, then, using the Events Editor, look at the component that had subscribed to an event published by the component you changed, it will show the uncoupling, as follows:



This is telling you that the **wccCases** component is subscribing to an event that the editor can no longer resolve because the component model name identified in the event subject no longer exists. You can leave it as is if that was your intention, or you can uncheck the unconfirmed event, then subscribe to the appropriate event.

After making a change to the component model name, the **Update** and **Default** buttons become active, allowing you to either save your changes, or to revert back to the default model name for the component. The **Commit** button must also be clicked to save all changes made in the Properties Editor.

## Disabling Default Handler

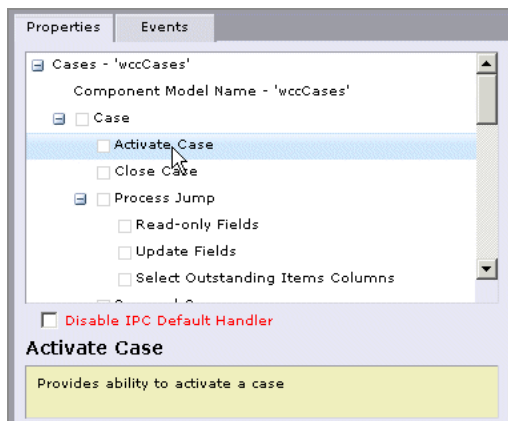
You can disable the default handler for any action that results in an event being triggered. For example, you may want to perform some sort of custom function when the user clicks on the **Activate** button on the case list. You can disable the normal “activate” function, subscribe to the **Activate** button event, and perform your custom function — that is, your application must handle the event.

Disabling the default handler can also be used in conjunction with the WCC methods to perform a function at a later time. For more information, see the *WCC Methods* chapter in the *TIBCO iProcess Workspace (Browser) Components Reference*.

To disable the default handler for an event, perform the following steps:

1. Open the Properties Editor for the component that includes the property corresponding to the functionality whose event you want to disable.

2. Click the property that corresponds to the function. For example:



Notice that when you click on a property that has a corresponding event, the **Disable IPC Default Handler** check box appears on the Properties Editor dialog.

3. Click in the **Disable IPC Default Handler** check box.
4. Click **Commit** to save your changes.

Now when the **Activate Case** event fires, the normal case activation function will not take place.

# Tutorial

---

This chapter provides the step-by-step procedure of creating a simple application using the TIBCO iProcess Workspace (Browser) components.

## Creating an Application

The following tutorial steps you through the process of creating an application that does the following:

- Displays the **Login** screen and a **Logout** button.
- Displays the **procedure list** after the user is authenticated using the **Login** screen.
- Displays the **case list** when the user clicks on a procedure in the procedure list.
- Shows a “Loading Data” message while the application is retrieving information from the server and rendering it on the screen — this is accomplished with the **wccDataMask** component.

Once the procedure and case lists are displayed, access to the functions on those lists is determined by a combination of the user’s access profile and the properties on the applicable components. For more information, see [Properties Editor](#).

This tutorial assumes you have installed all of the software listed in [TIBCO iProcess Environment](#), and your iProcess Engine, iProcess Objects Server, and Action Processor are running.

1. Start TIBCO® General Interface Builder and create a workspace if you haven’t created one yet (the TIBCO® General Interface workspace should be set to the directory that contains the `GI_Builder.html` file, as shown below).

TIBCO® General Interface Builder can be started by executing the following:

```
InstallDir\GI_Builder.html
```

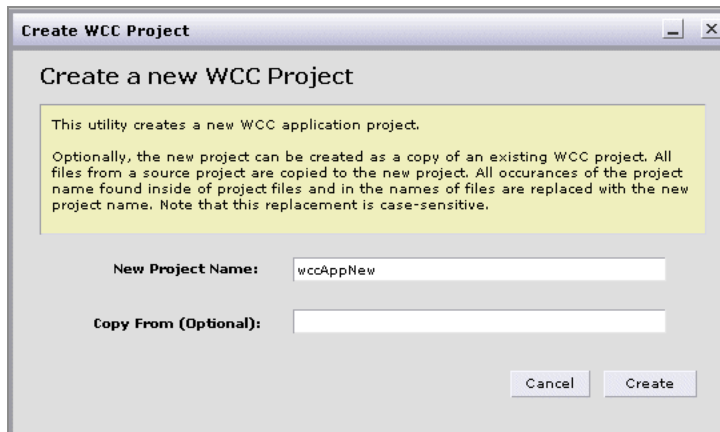
where:

- *InstallDir* is the location where the iProcess Workspace (Browser) Components is installed.

For more information about starting TIBCO® General Interface Builder and creating workspaces, see the *Starting TIBCO® General Interface Builder* section in *TIBCO® General Interface Getting Started*.

2. From the **Project** menu in TIBCO® General Interface Builder, select **New Project**.  
The **Choose a Project Type** dialog is displayed.
3. Select a "General Interface Application", then click **Next**.  
The **Choose a Project Template** dialog is displayed.
4. Select an "Empty Project" template for this tutorial, then click **Next**.  
The **Choose a Project Path** dialog is displayed.
5. Enter a name for a new project, then click **Finish**. This can be any name because this new project will be closed when you start a "WCC" (Workspace Client Components) project.
6. On the **Project** menu, select **Project Settings**.  
The **Project Settings** dialog is displayed.
7. Click on the **Add-ins** icon on the left side of the **Project Settings** dialog.  
Check boxes for the available add-ins are displayed.
8. Click in the **IPC Workspace Client Components** check box, then click **Save**.  
A dialog is displayed informing you that you must restart TIBCO® General Interface Builder for the add-in change to take effect. Click **OK** on this dialog.
9. Refresh the browser window to restart TIBCO® General Interface Builder (in Internet Explorer, select **Refresh** from the **View** menu). This causes the new add-in to be loaded.
10. From the **Project** menu, select **New Project**.  
The **Choose a Project Type** dialog is displayed.
11. Choose a "Workspace Client Components (WCC) Application", then click **Next**.

The following dialog is displayed:



This dialog is used to create a new WCC project. Creating a custom application that includes TIBCO iProcess Workspace (Browser) components requires that you create a WCC project that is based on an existing WCC project.

A *base-level* WCC project upon which you can base your project is provided. This base-level WCC project includes the TIBCO iProcess Workspace (Browser) components as an Add-in. This allows you to use those components, as well as the standard TIBCO® General Interface Builder components, to create your custom application.

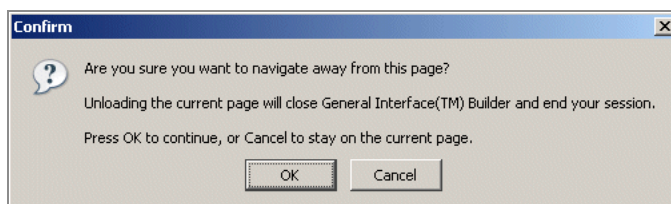
The default is to create your new WCC project based on the provided base-level WCC project. You can override this by specifying, in the **Copy From** field, the name of the existing WCC project upon which you want to base the new project.

12. In the **New Project Name** field, enter the name you would like to give your new project. Use “Accounts” for this tutorial.

In this tutorial, you will use the base-level WCC project as our starting point, and leave the **Copy From** field blank.

13. Click the **Create** button.

The following confirmation dialog is displayed:



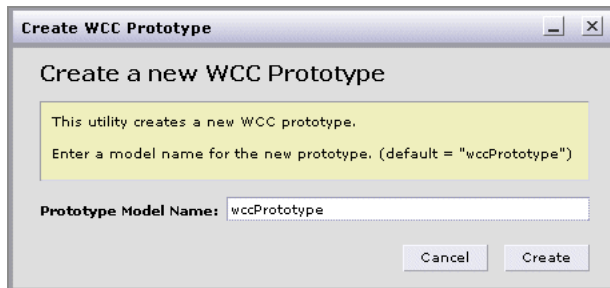
TIBCO® General Interface Builder must refresh the browser to load the new project.

14. Click **OK**.

After TIBCO® General Interface Builder reloads the new project, notice the **JSXAPPS/Accounts** link in the lower left-hand corner of the TIBCO® General Interface Builder dialog. This tells you what project you are currently in, as well as provides a way to easily navigate via the browser to the directory that contains all of your project files.

15. From the **New** menu in TIBCO® General Interface Builder, select **GUI Component**.

The following dialog is displayed:



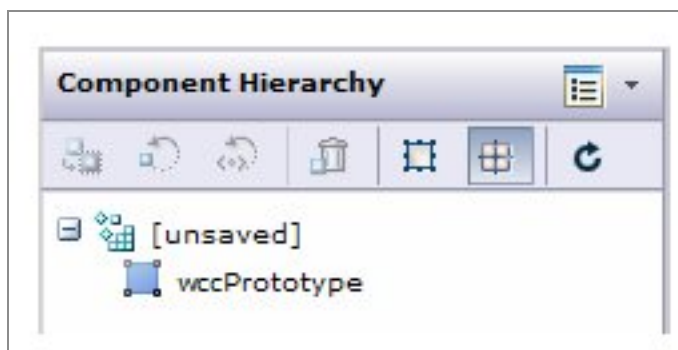
If you are going to add TIBCO iProcess Workspace (Browser) components to your application, they must be inside of a WCC prototype. You can think of the WCC prototype as a component container.

Creating a WCC prototype sets up the PageBus so that it can wire-up all of the component events you add to the prototype.

16. Enter a *model* name for the prototype, then click **Create**. Use “AccountsPrototype” in this tutorial.

The prototype model name is used in the *subject* string internally by the PageBus to identify events associated with specific prototypes/components.

After creating the WCC prototype, the **Component Hierarchy** palette will appear as follows:



Notice that the model name does not appear in the Component Hierarchy. However, if you view the source XML for the WCC prototype, the model name is defined in the **prototypeModelName** attribute:

```
<strings jsxname="wccPrototype"
  jsxwidth="100%"
  jsxheight="100%"
  appModelName="com.tibco.wcc.Accounts"

  prototypeModelName="AccountsPrototype"
  ...
```

Also, notice that the WCC prototype is actually contained within a top-level prototype object, which is unnamed at this point. This top-level prototype represents a TIBCO® General Interface user-defined prototype in our project — this prototype can contain WCC prototypes.

17. From the **File** menu, select **Save** (or right click on the work area tab and select **Save**).

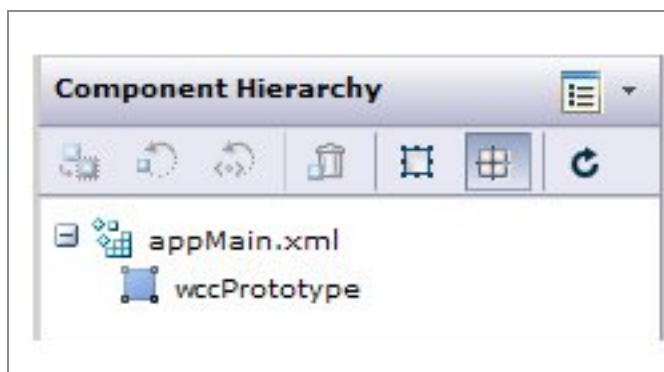
The file you are saving at this point is the TIBCO® General Interface user-defined prototype. It needs to be saved in the **prototypes** directory under your project.

18. Navigate to the following directory:

```
WCCCOMPONENTSDIR\JSXAPPS\WCCProjectName\application\prototypes
```

where *WCCCOMPONENTSDIR* is the TIBCO® General Interface workspace directory you established the first time TIBCO® General Interface Builder was started (see step 1) and *WCCProjectName* is the name you gave the WCC Project in step 9 (“Accounts” in this tutorial).

19. Save the new user-defined prototype in the `appMain.xml` file, overwriting the existing file. The **Component Hierarchy** palette now appears as follows:



The tab in the work area also now contains the name of the TIBCO® General Interface user-defined prototype, `appMain.xml`.

20. Using the standard TIBCO® General Interface Builder Layout and Splitter components, create a screen layout similar to the following:



Note - If you are not familiar with TIBCO® General Interface Builder, you can see [Tutorial Screen Layout](#) for detailed steps about how to create this layout using the standard TIBCO® General Interface Builder components.

21. From the **Component Libraries** palette, select and drag the **Login** component into the left-side pane created by the vertical splitter (the **Login** component is located in **Addins/IPC Workspace Client Components/Authentication**). (You could also drag and drop the **Login** component onto the appropriate pane in the **Component Hierarchy** palette.)

The **Login** dialog appears in the pane.

22. From the **Component Libraries** palette, select and drag the **Logout** component into the top right area created by the Layout components (the **Logout** component is located in **Addins/IPC Workspace Client Components/Authentication**).
23. From the **Component Libraries** palette, select and drag the **DataMask** component into the top center area created by the Layout components (the **DataMask** component is located in **Addins/IPC Workspace Client Components/Application**).



The **DataMask** component provides a “Loading Data ...” message while the application is retrieving data from the server and rendering it on the screen.

24. From the **Component Libraries** palette, select and drag the **StartHistoryProcs** component onto the appropriate pane in the **Component Hierarchy** palette that represents the area to the left of the vertical splitter, i.e., the pane in which the **Login** component is located (the **StartHistoryProcs** component is located in **Addins/IPC Workspace Client Components/Lists/Procedures**).

Note that you cannot place the **StartHistoryProcs** component directly in the work area pane because the **Login** component is already in that pane. (When a component is dropped into a container in the work area, it, by default, consumes 100% of the container. If you want to place another component in the same container, TIBCO® General Interface Builder requires that you place it in the pane in the Component Hierarchy tree.)

The **StartHistoryProcs** component displays a procedure list containing the procedures for which the logged-in user has permission to start and view case history. You will later configure events so that this list is displayed when the user is authenticated through the **Login** component.

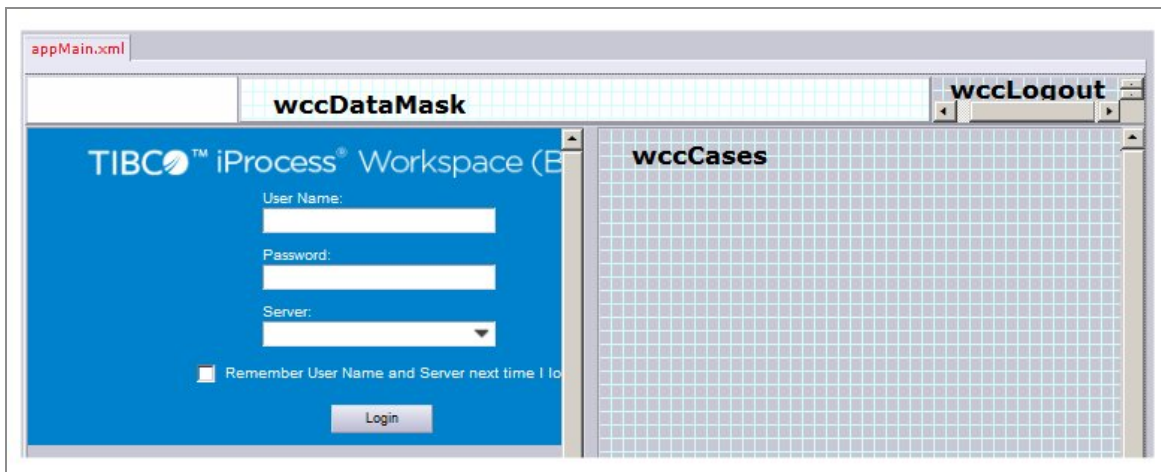
**i Note:** When dragging and dropping WCC components onto the **Component Hierarchy**, do not drop them anywhere above the WCC prototype in the component tree. WCC components must be placed in a WCC prototype.

25. From the **Component Libraries** palette, select and drag the **Cases** component into the right-side pane created by the vertical splitter (the **Cases** component is located in **Addins/IPC Workspace Client Components/Lists/Cases**).

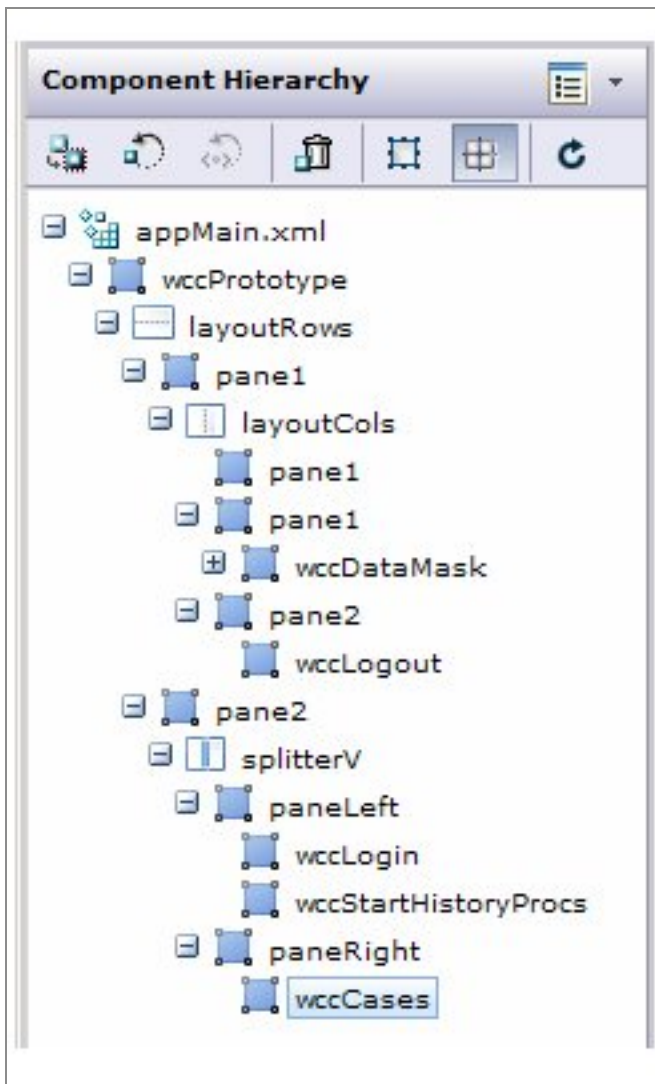
The **Cases** component represents a case list containing the cases for a particular procedure.

You will later configure events so that this list is displayed when the user clicks on a procedure in the procedure list.

Notice that a grid pattern representing each component is shown in the containers to provide visual feedback:



The **Component Hierarchy** should now look like this (if the components are not in this order, move them by dragging and dropping them in the proper location):



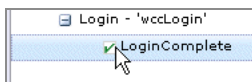
Now you will wire-up the events that will cause the procedure list to display when the user is authenticated, and the case list to display when the user clicks on a procedure in the procedure list.

26. In the **Component Hierarchy** palette, click on the **wccStartHistoryProcs** component, then click on the **wccStartHistoryProcs** button in the taskbar to display the **Properties/Events Editor** dialog.

27. Click on the **Events** tab.

**i Note:** In this tutorial, you will not change any of the property settings on the **Properties** tab, which are used to specify access to functionality provided by the component. For information about setting properties, see [Properties Editor](#).

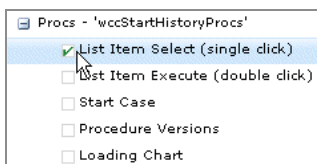
28. Double-click the **LoginComplete** event, which causes the check box to become checked:



This causes the **wccStartHistoryProcs** component to subscribe to the “LoginComplete” event, which the **Login** component publishes.

This results in the procedure list being displayed when the user’s login credentials are authenticated.

29. Click **Commit** to save the changes. You can optionally minimize or close the **Properties/Events Editor** dialog, or keep it open.
30. In the **Component Hierarchy** palette, click on the **wccCases** component. If you kept the **Properties/Events Editor** dialog open from the previous step, it automatically changes to show the **wccCases** component; if you minimized or closed it, click on the **wccCases** button in the taskbar to display the **Properties/Events Editor** dialog.
31. Click the **Events** tab.
32. Double-click on the “List Item Select (single click)” event for the **wccStartHistoryProcs** component.



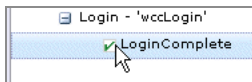
This causes the **wccCases** component to subscribe to the “List Item Select (single click)” event, which the **wccStartHistoryProcs** component publishes.

This results in the case list being displayed when the user selects (single clicks) a procedure in the procedure list.

33. Click **Commit** to save the changes.

34. In the **Component Hierarchy** palette, click on the **wccLogout** component.

35. In the **Properties/Events Editor**, double-click the “LoginComplete” event, which causes the check box to become checked:



This causes the **wccLogout** component to subscribe to the “LoginComplete” event, which the **wccLogin** component publishes.

This results in the **Logout** button being displayed when the user’s login credentials are authenticated.

36. Click **Commit** to save the changes, then minimize or click **Close** to close the editor dialog.

37. From the **File** menu, select **Save**.

You have now created an application that will allow an iProcess user to log into an iProcess Objects Server and view cases of procedures that have been started on that server, and to perform any of the available functions to which that user has access.

38. In Windows Explorer, navigate to your workspace directory and execute the *ProjectName.html* file. In this tutorial, the file is *Accounts.html*. (Note that you can also test/run the application from within TIBCO® General Interface Builder, rather than via Windows Explorer.)

The **Login** dialog is displayed.

39. Enter a valid user name and password, that is, one that has been set up on the iProcess Objects Server.

40. Click on the **Server** field drop-down list, select the iProcess Objects Server you want to log into, then click **OK**.

The **Server** drop-down list presents all of the iProcess Objects Servers that have been specified in the iProcess Workspace (Browser) configuration file, *config.xml*. One iProcess Objects Server is specified when the iProcess Workspace (Browser) is installed — additional servers can be specified later. For more information about specifying servers in the configuration file, see the *Server Nodes* section in the *iProcess Workspace (Browser) Configuration and Customization* guide.

When the user's name and password are authenticated, the procedure list is displayed.

41. Click a procedure in the list.

The screen should now look similar to the following:

Stat	Name	Description	Version	Permission	Active	Total
	ALLOCATE	Allocate Resourc	0.5	Start / History	10	11
	BONUS	Bonus	0.2	Start / History	0	0
	CARPOOL	Company Car Al	1.0	Start / History	28	45
	CREDCHK		0.0	History	1	4
	DYN1		0.0	History	6	9
	DYN2		0.0	History	0	0
	DYNAMIC	Dynamic Sub Pr	0.0	Start / History	7	8
	HIRING	Hiring Personnel	1.0	Start / History	19	22
	INVENTOR	Inventory	0.0	Start / History	2	2
	ORDER	Brand orders	0.9	Start / History	20	20
	QUANTITY	Quantity	0.0	Start / History	2	2
	TIMEOFF	TimeOff	0.3	Start / History	6	6

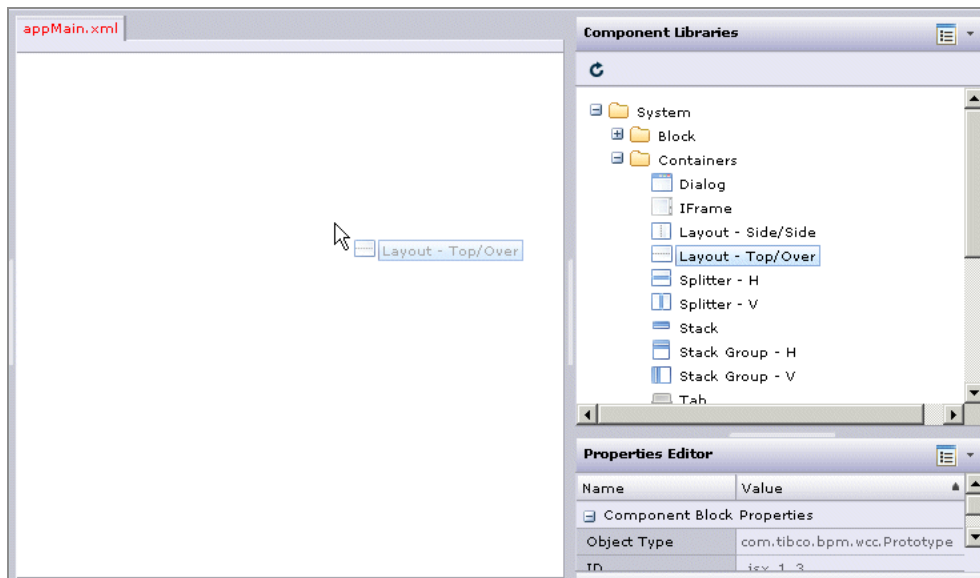
Stat	Number	Description	Started By	Started Date/Time	Terminated Date/Time
	53	Franklin Morse - HR rep	bobby@v11	2009-03-31 11:50:03	2009-12-09 08:56:11
	101	Annie Smith - Clerk II	bobby@v11	2009-03-31 11:40:30	2009-12-09 08:58:11
	252	Cindy Manson - Marketir	bobby@v11	2009-03-31 13:38:27	2010-07-16 09:57:11
	301	Linda Horner - Engineer	swadmin@v11	2009-03-31 13:20:13	
	503	General Admin III	bobby@v11	2009-04-06 11:09:05	
	753	John Jackson - Dev II	swadmin@v11	2009-06-11 14:10:06	
	853	Sandy Nelson - Dev II	swadmin@v11	2009-06-11 14:24:22	
	903	Mona Whetland - HR	swadmin@v11	2009-06-11 14:25:07	
	953	Region Mgr - 03-13-09	swadmin@v11	2009-06-11 14:25:40	
	3603	Anne White - Sales 7725	swadmin@v11	2010-06-28 15:17:14	
	3653	Rob Robertson	bobby@v11	2010-06-29 07:49:15	
	3703	Hal Holbrook	bobby@v11	2010-06-29 07:50:35	

42. Perform any of the available functions for which you have permission.

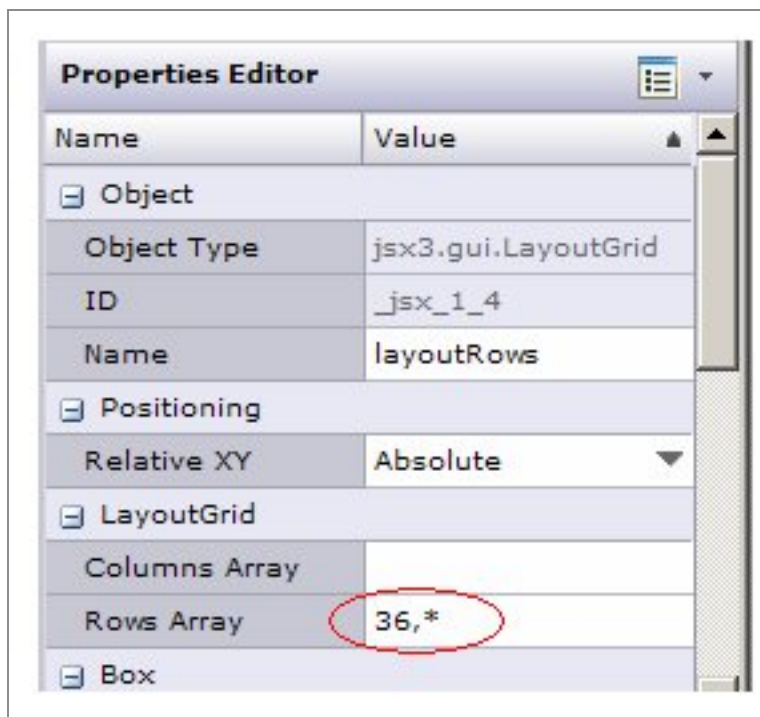
## Tutorial Screen Layout

In [Step 20](#) of the tutorial, you are asked to create a layout using the standard TIBCO® General Interface Builder components. If you are not familiar with TIBCO® General Interface Builder, you can use the following steps to help you create the layout:

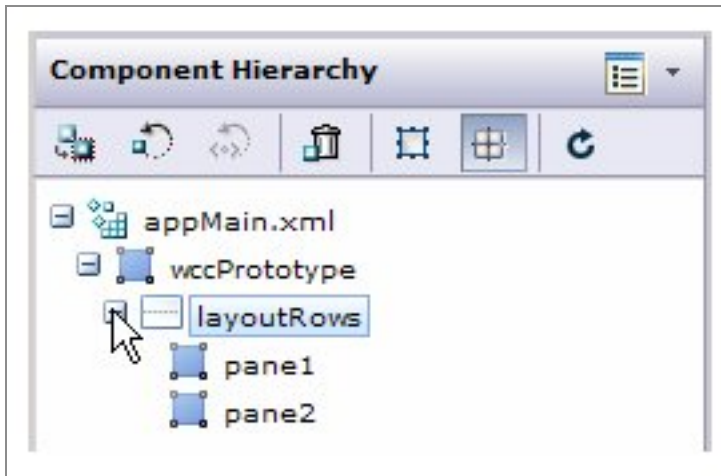
1. From the **Component Libraries** palette in the upper right, expand **Containers** and drag a **Layout - Top/Over** component into the **appMain.xml** canvas.



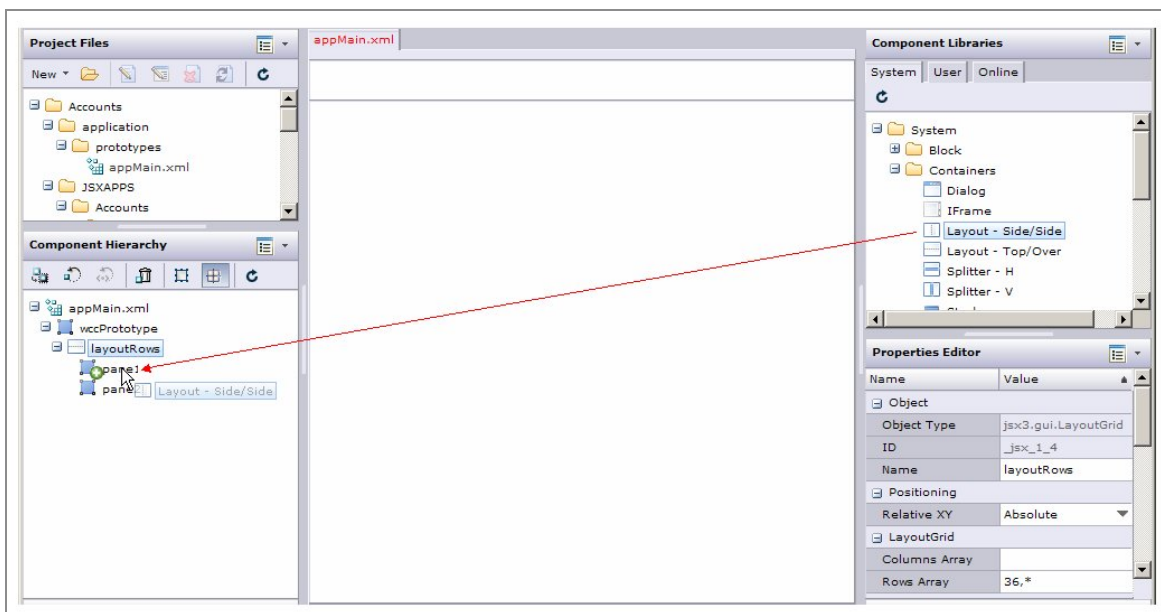
2. In the **Properties Editor** palette in the lower right, change the **Rows Array** property value to “36,\*”.



3. Expand the **layoutRows** component in the **Component Hierarchy** palette, in the lower left.

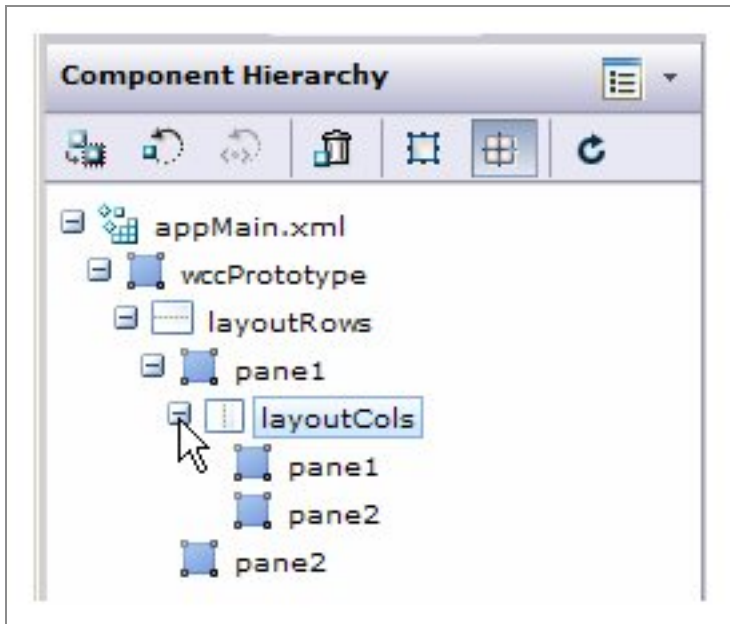


4. From the **Component Libraries** palette, drag a **Layout - Side/Side** component into **pane1** of the **layoutRows** component in the **Component Hierarchy** palette.

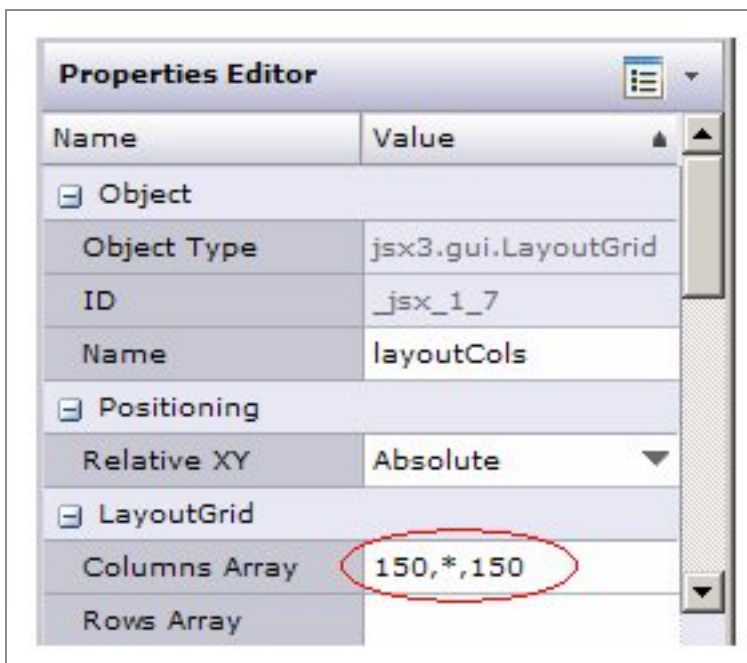


5. Expand and then select the **layoutCols** component in the **Component Hierarchy** palette.



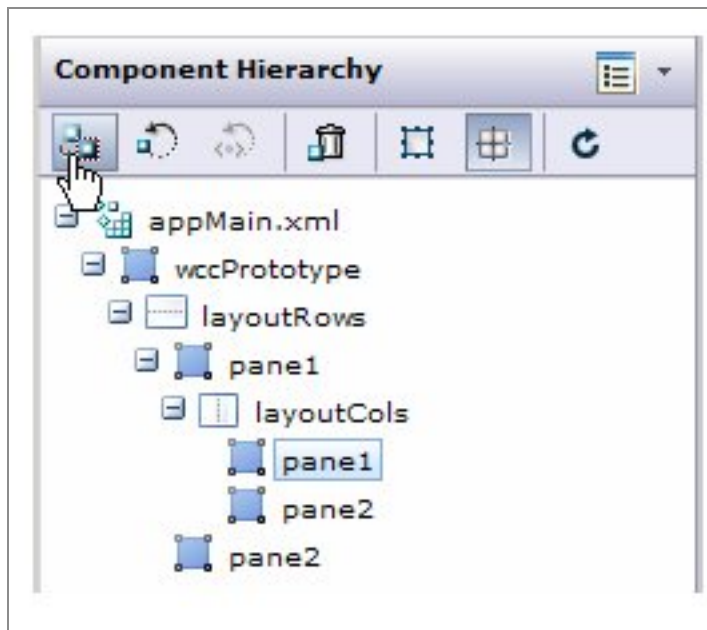


6. In the **Properties Editor** palette, change the **layoutCols** component **Columns Array** property value to “150,\*,150”.

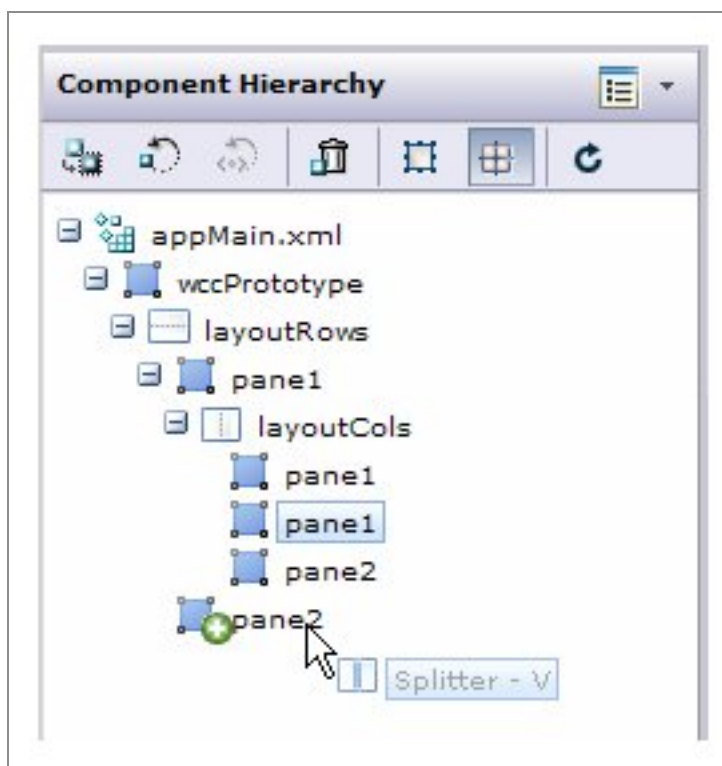


7. In the **Component Hierarchy** palette, select the **pane1** component, under **layoutCols** and then click the **Clone** tool.





8. From the **Component Libraries** palette, drag a **Splitter - V** component into **pane2** of the **layoutRows** component in the **Component Hierarchy** palette.



9. You can now proceed with [Step 21](#) of the tutorial.

# Sample Applications

---

This chapter describes the WCC-related sample applications provided in TIBCO iProcess Workspace (Browser).

## Introduction

There are two WCC-related sample applications installed if you select "Samples" when installing the TIBCO iProcess Workspace (Browser). They are:

- **openWorkQViaURL** - This application opens a work item list and preview pane based on a work queue tag parameter in the URL.
- **openWorkItemViaURL** - This application allows you to do one of the following:
  - open a work item form using URL parameters to specify the work item that is to be locked and opened, or
  - start a case using URL parameters to specify the procedure for which to start a case.

These are described in detail in the following sections.

## openWorkQViaURL

The **openWorkQViaURL** application is a WCC application that demonstrates how to login and automatically load a **WorkItemsPreview** component, containing a list of work items and a preview pane, using a URL parameter to specify the work queue that is to be opened.

Rather than selecting a work queue from a list, the user passes the necessary identifying information in the URL. Immediately upon logging in, the identified work queue is opened.

The openWorkQViaURL application is a WCC application that is provided in the "samples", i.e., you must specify to install the samples when installing the iProcess Workspace (Browser). It is installed in the following directory:

```
InstallationHomeDir\iproccsclientbrowser\samples\openWorkQViaURL
```

where *InstallationHomeDir* is the directory in which the installer places administrative files, such as the uninstaller, documentation, and sample code. This defaults to C:\tibco on Windows systems, and /opt/tibco on UNIX systems, but can be specified as a different directory when the iProcess Workspace (Browser) is installed.

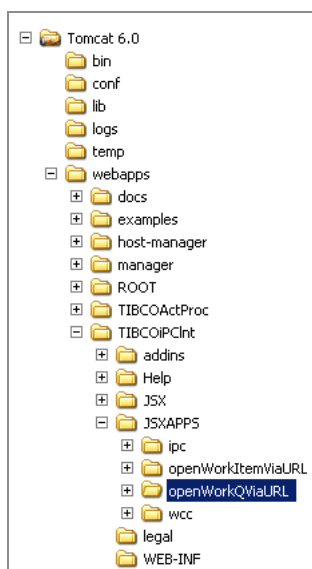
Note that since the openWorkQViaURL application is a WCC application, you must also install the “components” when installing the iProcess Workspace (Browser).

For step-by-step instructions on setting up and using this WCC application, see the [Setting Up and Using openWorkQViaURL](#) section on [Setting Up and Using openWorkQViaURL](#)

## Setting Up and Using openWorkQViaURL

To set up and use the **openWorkQViaURL** application, follow the steps below. Note that this procedure assumes the openWorkQViaURL application will be run in Tomcat. If you are running in a different application server, or locally, modify the locations accordingly.

1. Copy the samples\openWorkQViaURL directory to TOMCAT\_HOME\webapps\ClientInstallDir\JSXAPPS.
2. Modify JSXAPPS\openWorkQViaURL\config.xml to specify the **ServerNodes** and **ActionProcessors** records. These specify the servers and action processors you can connect to when running the **openWorkQViaURL** application. (You may be able to copy these from the JSXAPPS\ipc\config.xml file.)



For information about specifying these records, see the *Configuring the Client Application* chapter in the *TIBCO iProcess Workspace (Browser) Configuration and Customization* guide.

3. Move the `openWorkQViaURL.html` file from the `JSXAPPS\openWorkQViaURL` directory to the client installation directory (the directory above `JSXAPPS` — in this example, `TIBCOiPCInt`).
4. Open `openWorkQViaURL.html` in a browser with URL parameters identifying the work queue.

For information about the allowable URL parameters, see [URL Parameters](#).

## URL Parameters

To open a work queue via a URL, use the following form:

```
http://Host:Port/AppDir/openWorkQViaURL.html?workQTag=WorkQTag
```

where:

- *Host* is the machine name on which you are hosting the application.
- *Port* is the port used by the application server.
- *AppDir* is the location of the **openWorkQViaURL** application on the application server.
- *WorkQTag* is a tag that identifies the work queue to open. This must be in the following form:

```
nodeName|workQName|releasedState
```

where:

- *nodeName* is the name of the TIBCO iProcess Engine / iProcess Objects Server to which the user can log in. This is the “nodename” that is assigned to the iProcess Engine when it is installed.
- *workQName* is the name of the work queue to open.
- *releasedState* is either “R” or “U” to indicate if the work queue is released

For example:

```
http://CorpServer:7676/TIBCOiPCLnt/openWorkQViaURL.html?workQTag=
liberty|swadmin|R
```

## openWorkItemViaURL

The **openWorkItemViaURL** application is a WCC application that demonstrates how to do one of the following:

- open a work item form using URL parameters to specify the work item that is to be locked and opened.
- start a case using URL parameters to specify the procedure for which to start a case.

Rather than selecting a work item or procedure from a list, the user passes the necessary identifying information in the URL. Immediately upon logging in, the identified work item form is displayed, or a case of the specified procedure is started.

The openWorkItemViaURL application is a WCC application that is provided in the “samples”, i.e., you must specify to install the samples when installing the iProcess Workspace (Browser). It is installed in the following directory:

```
InstallationHomeDir\iprocessclientbrowser\samples\openWorkItemViaURL
```

where *InstallationHomeDir* is the directory in which the installer places administrative files, such as the uninstaller, documentation, and sample code. This defaults to `C:\tibco` on Windows systems, and `/opt/tibco` on UNIX systems, but can be specified as a different directory when the iProcess Workspace (Browser) is installed.

Note that since the openWorkItemViaURL application is a WCC application, you must also install the “components” when installing the iProcess Workspace (Browser).

Work items using either “GI Forms” or “External Forms”<sup>1</sup> can be opened in this way. However, since the form is opened before the application layout has been painted, the form cannot be opened in the Preview Pane; it may only be opened in a floating window (dialog or separate browser window, depending on the type of form and configuration parameters in the `config.xml` file). For more information about forms, see the *Forms* chapter in the *TIBCO iProcess Workspace (Browser) Configuration and Customization* guide.

---

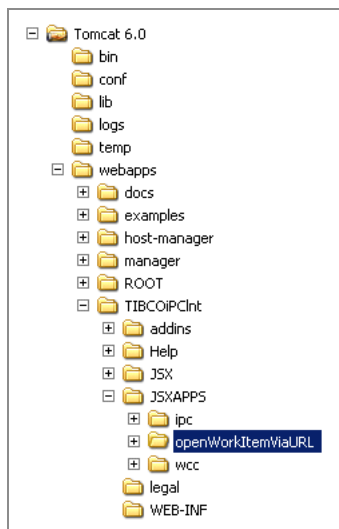
<sup>1</sup>“GI Forms” include TIBCO® General Interface Forms and TIBCO Forms. “External Forms” include ASP Forms, JSP Forms, BusinessWorks FormBuilder Forms, and iProcess Modeler Forms.

For step-by-step instructions on setting up and using this WCC application, see the [Setting Up and Using openWorkItemViaURL](#) section on [Setting Up and Using openWorkItemViaURL](#).

## Setting Up and Using openWorkItemViaURL

To set up and use the **openWorkItemViaURL** application, follow the steps below. Note that this procedure assumes the openWorkItemViaURL application will be run in Tomcat. If you are running in a different application server, or locally, modify the locations accordingly.

1. Copy the samples\openWorkItemViaURL directory to TOMCAT\_HOME\webapps\ClientInstallDir\JSXAPPS.
2. Modify JSXAPPS\openWorkItemViaURL\config.xml to specify the **ServerNodes** and **ActionProcessors** records. These specify the servers and action processors you can connect to when running the **openWorkItemViaURL** application. (You may be able to copy these from the JSXAPPS\ipc\config.xml file.)



For information about specifying these records, see the *Configuring the Client Application* chapter in the *TIBCO iProcess Workspace (Browser) Configuration and Customization* guide.

3. Move the openWorkItemViaURL.html file from the JSXAPPS\openWorkItemViaURL directory to the client installation directory (the directory above JSXAPPS — in this example, TIBCOiPCInt).
4. Open openWorkItemViaURL.html in a browser with URL parameters identifying the work item.

For information about the allowable URL parameters, see [URL Parameters](#).

**i Note:** The **openWorkItemViaURL** application makes use of the WCC static method, **openWorkItemEx**, to open work items via URL parameters. For information about this method, see the *WCC Methods* chapter in the *TIBCO iProcess Workspace (Browser) Components Reference*.

## URL Parameters

The following subsections describe the URL parameters to pass for opening a work item and starting a case.

### Opening a Work Item

You can open a work item with the **openWorkItemViaURL** application by passing parameters in the URL in two ways: either as individual parameters or as a work item tag, as described below.

#### Passing Individual Parameters

To open a work item via a URL by passing individual parameters, use the following form:

```
http://Host:Port/AppDir/openWorkItemViaURL.html?caseNumber=
CaseNum&procName=ProcName&stepName=StepName&queueName=QName&
isReleased=ReleasedState
```

where:

- *Host* is the machine name on which you are hosting the application.
- *Port* is the port used by the application server.
- *AppDir* is the location of the **openWorkItemViaURL** application on the application server.
- *CaseNum* is the case number in which the work item was started.
- *ProcName* is the name of the procedure for the work item.
- *StepName* is the name of the step to which the work item relates.
- *QName* is the name of the work queue in which the work item resides

- *ReleasedState* is either "Y" or "N" to indicate if the work queue is released

For example:

```
http://CorpServer:7676/TIBCOiPCInt/openWorkItemViaURL.html?caseNumber=8384&procName=ALLOCATE&stepName=STEP1&queueName=swadmin&isReleased=Y
```

Note that it is possible for multiple work items to match the arguments. If this occurs, a “Duplicate work items” message is displayed and the form fails to open.

## Passing a Work Item Tag

If the work item tag is available, it can be passed as a single argument. A work item tag may be specified in one of the following forms:

```
workItemTag=nodeName|procName|workQName|isReleased|caseNumber|mailId|stepName|procMajorVer|procMinorVer
```

or

```
workItemTag=nodeName|procName|workQName|isReleased|caseNumber|requestId|requestIdHost|stepName|procMajorVer|procMinorVer
```

For example:

```
http://CorpServer:8080/TIBCOiPCInt/openWorkItemViaURL.html?workItemTag=i2tagtest|ALLOCATE|swadmin|R|10928|439124|STEP1|0|0
```

## Starting a Case

You can start a case with the **openWorkItemViaURL** application by passing a single procedure tag in the following form:

```
procTag=nodeName|procName|procMajorVer|procMinorVer
```

To start a case of the current version of the procedure, specify -1 for **procMajorVer** and **procMinorVer**, otherwise, include a specific procedure version.

The following is an example of a URL for starting a case:



```
http://CorpServer:7676/TIBCOiPCInt/openWorkItemViaURL.html?procTag=MySer  
ver|CARPOOL|-1|-1
```

# TIBCO Documentation and Support Services

---

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

## How to Access TIBCO Documentation

Documentation for TIBCO products is available on the [Product Documentation website](#), mainly in HTML and PDF formats.

The [Product Documentation website](#) is updated frequently and is more current than any other documentation included with the product.

## Product-Specific Documentation

Documentation for TIBCO iProcess® Workspace (Browser) is available on the [TIBCO iProcess® Workspace \(Browser\) Product Documentation](#) page.

## How to Contact Support for TIBCO Products

You can contact the Support team in the following ways:

- To access the Support Knowledge Base and getting personalized content about products you are interested in, visit our [product Support website](#).
- To create a Support case, you must have a valid maintenance or support contract with a Cloud Software Group entity. You also need a username and password to log in to the [product Support website](#). If you do not have a username, 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 [TIBCO Community](#).

# Legal and Third-Party Notices

---

SOME CLOUD SOFTWARE GROUP, INC. (“CLOUD SG”) SOFTWARE AND CLOUD SERVICES EMBED, BUNDLE, OR OTHERWISE INCLUDE OTHER SOFTWARE, INCLUDING OTHER CLOUD SG SOFTWARE (COLLECTIVELY, “INCLUDED SOFTWARE”). USE OF INCLUDED SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED CLOUD SG SOFTWARE AND/OR CLOUD SERVICES. THE INCLUDED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER CLOUD SG SOFTWARE AND/OR CLOUD SERVICES OR FOR ANY OTHER PURPOSE.

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

This document 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 Cloud Software Group, Inc.

TIBCO, the TIBCO logo, the TIBCO O logo, and iProcess are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or 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. You acknowledge that all rights to these third party marks are the exclusive property of their respective owners. Please refer to Cloud SG’s Third Party Trademark Notices (<https://www.cloud.com/legal>) for more information.

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: <https://scripts.sil.org/OFL>

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

Cloud SG 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 a specific version of Cloud SG software 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. CLOUD SG MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S), THE PROGRAM(S), AND/OR THE SERVICES DESCRIBED IN THIS DOCUMENT AT ANY TIME WITHOUT NOTICE.

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 "README" FILES.

This and other products of Cloud SG may be covered by registered patents. For details, please refer to the Virtual Patent Marking document located at <https://www.cloud.com/legal>.

Copyright © 2005-2025. Cloud Software Group, Inc. All Rights Reserved.