TIBCO iProcess® Workspace (Browser)

Components Reference

Version 11.10.0 | May 2025

Contents

Introduction

This section provides an overview of each of the components available when the TIBCO iProcess® Workspace (Browser) components are installed.

Available Components

For more information about using the functions provided by the TIBCO iProcess Workspace (Browser) components, see the *TIBCO iProcess Workspace (Browser) User Guide*.

Click the component name below to view more information about that component.

Application

- DataMask - block that displays "Loading Data…" mask while the application is retrieving data from the server/engine

Authentication

- Login - interface for performing application login
- Logout - interface for performing application logout

Case Management

These components provide the ability to independently display the "case detail" tabs that are normally displayed when the user opens a case from the case list. They include:

- CaseDetail - this single component displays all four of the other case management components listed below
- CaseSummary - displays summary information for the selected case
- CaseHistory - displays a list of all actions that have taken place in the case
- CaseOutstanding - displays a list of all outstanding work items in the case

- CaseData - displays all fields in the case, as well as the case data values for those fields

Composites

These components combine more than one component into a single component to provide an easy way to display multiple lists or elements without the need to set up the events between those lists or elements — the events between the components provided in the composites are already configured. They include:

- CasesPreview - composite of the case list component (see Cases) and the preview pane
- ProcsCases - composite of the procedure list component (**StartHistoryProcs**) — see Procedures) and the case list component (see Cases)
- WorkItemsPreview - composite of the work item list component (see Work Items) and the preview pane
- WorkQsItems - composite of the work queue list component (see UserWorkQs) and the work item list component (see Work Items)

Lists

These components provide lists of items, such as cases and work items.

- Cases
  — Cases - list of all cases for a specific procedure
- Procedures
  — HistoryProcs - lists only the procedures for which the user is permitted to view cases - lists only the procedures for which the user is permitted to view cases - list of all procedures for which the user is permitted to view the cases (i.e., the case list)

  — NodeProcs - list of all procedures defined on a server node - list of all procedures defined on a server node - list of all procedures defined on a server node

  — StartHistoryProcs - lists the procedures for which the user is permitted to view cases and/or start cases - lists the procedures for which the user is permitted to view cases and/or start cases - list of all procedures for which the user is permitted to start cases and/or view the cases

— StartProcs - lists only the procedures for which the user is permitted to start new cases - lists only the procedures for which the user is permitted to start new cases - list of all procedures for which the user is permitted to start new cases

Note: “History” permission in this context means you have permission to view the list of cases of the procedure. It does not mean you have “case history” permission; rather it means you have “procedure history” permission, that is, you can view the cases that have been started for that procedure.

- Work Items

  — WorkItems - list of all work items for a specific work queue

- Work Queues

  — UserWorkQs - list of all work queues to which a specific user has access

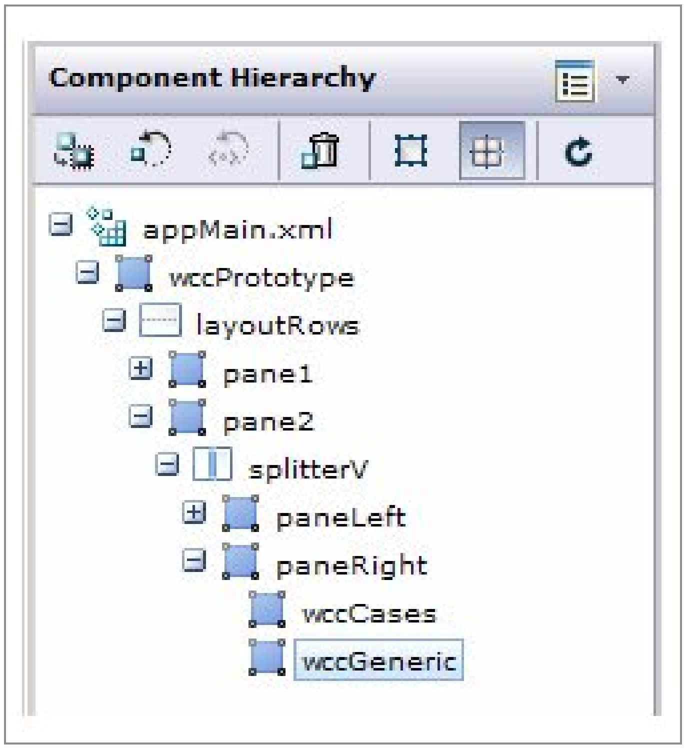
Generic Component

The **Generic** component is available as a troubleshooting tool. It allows you to view information about events as they are triggered, giving you a visual confirmation that events are occurring as you test your application.

To use the **Generic** component, perform the following steps:

1. From the **Component Libraries** palette, select and drag the **Generic** component onto the **wccPrototype** prototype in the **Component Hierarchy** palette (the **Generic** component is located in **Addins/IPC Workspace Client Components**).

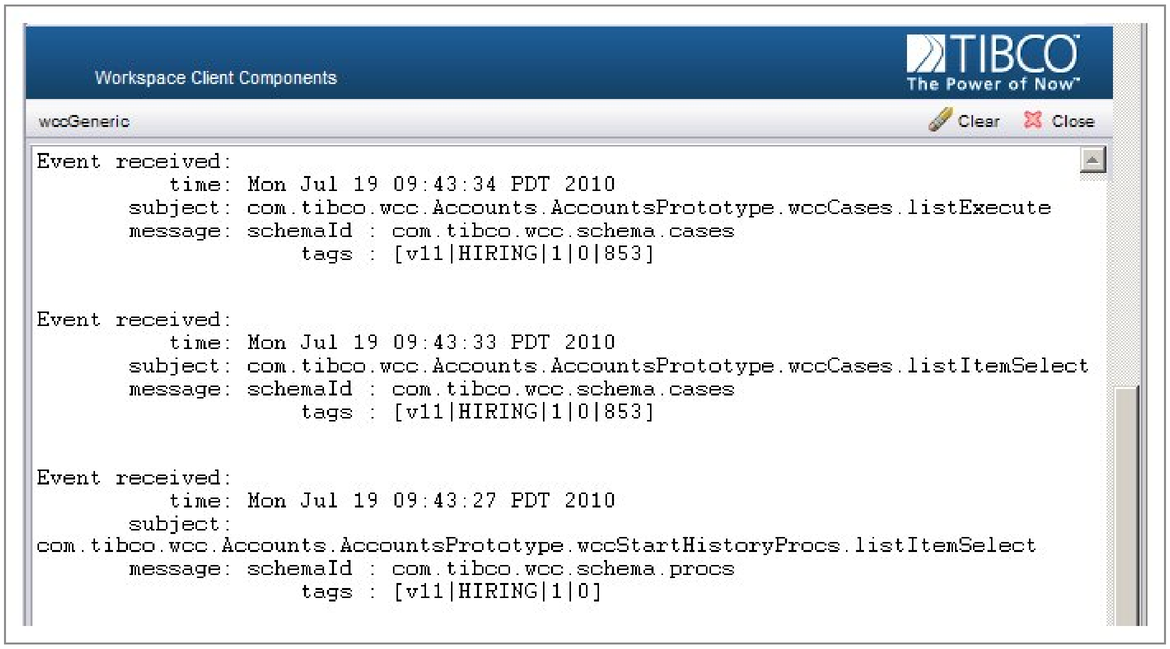
The **Component Hierarchy** looks similar to this:



2. In the **Component Hierarchy** palette, click the **wccGeneric** component.
3. Display **Properties/Events Editor** by clicking the **wccGeneric** button in the task bar.
4. Click the **Events** tab.

   The Event Editor for the **Generic** component contains events for all other components you have added to your application.
5. Select the checkboxes for the events you are interested in viewing.
6. Click **Commit** to save the changes.
7. Select **Save** from the **File** menu.
8. In Windows Explorer, navigate to your workspace directory and execute the `ProjectName.html` file, where *ProjectName* is the name you gave your project in TIBCO® General Interface Builder. (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.
9. Perform some actions in your application, then view the output caused by the **Generic** component. An example is shown below:



```
Event received:
          time: Mon Jul 19 09:43:34 PDT 2010
       subject: com.tibco.wcc.Accounts.AccountsPrototype.wccCases.listExecute
       message: schemaId : com.tibco.wcc.schema.cases
                    tags : [v11|HIRING|1|0|853]


Event received:
          time: Mon Jul 19 09:43:33 PDT 2010
       subject: com.tibco.wcc.Accounts.AccountsPrototype.wccCases.listItemSelect
       message: schemaId : com.tibco.wcc.schema.cases
                    tags : [v11|HIRING|1|0|853]


Event received:
          time: Mon Jul 19 09:43:27 PDT 2010
       subject:
com.tibco.wcc.Accounts.AccountsPrototype.wccStartHistoryProcs.listItemSelect
       message: schemaId : com.tibco.wcc.schema.procs
                    tags : [v11|HIRING|1|0]
```

The **Generic** component output provides information about each event caused by an action in your application. It includes the event subject string, as well as the message sent to the PageBus.

Use the **Clear** button to clear the current information, but continue displaying the **Generic** component output window.

Use the **Close** button to close the **Generic** component output window.

Note: The **Generic** component has two properties — **Show Clear** button and **Show Close** button — that control the visibility of the **Clear** and **Close** buttons. It also publishes two events — **Generic Closed** and **Generic Clear** — that fire when the **Close** and **Clear** buttons are clicked.

Component Reference

This section provides information about the details of each of the TIBCO iProcess Workspace (Browser) components.

This guide does not provide detailed information about using the functions provided by the TIBCO iProcess Workspace (Browser) components. For those details, see the *TIBCO iProcess Workspace (Browser) User Guide*.

Application

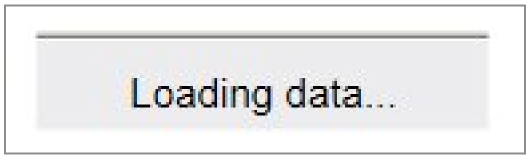
There is a single Application component available:

- DataMask - block that displays "Loading Data…" mask during operations

For more information, see the following subsection.

DataMask

This component provides a "Loading Data ..." data mask while the application is retrieving information from the server and rendering it on the screen:



This provides a visual to the user that the application is working.

Note: The **DataMask** component does not have any editable features — i.e., no properties nor events. It also does not require that data be passed to it, that is, it does not need to subscribe to an event of another component.

Authentication

The following are the Authentication components available:

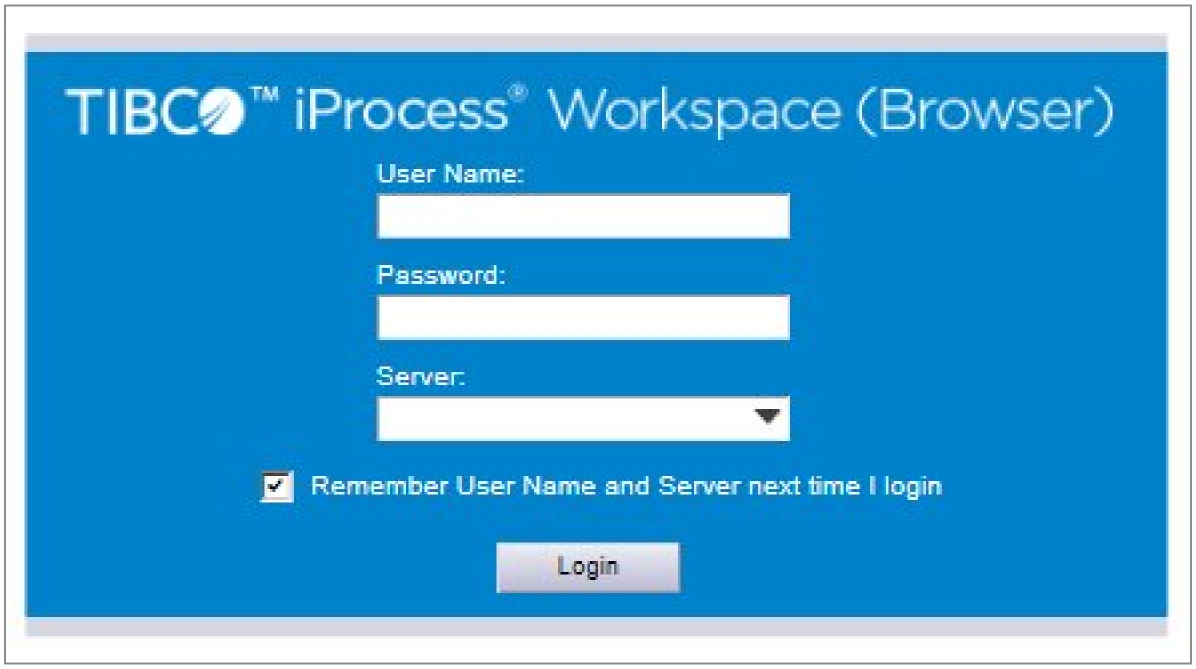
- Login - interface for performing application login
- Logout - interface for performing application logout

For more information, see the following subsections.

Login

The **Login** component provides an interface for logging into a TIBCO iProcess Objects Server.

This component displays the **Login** dialog box, which accepts the user's credentials and allows the user to choose a server to log into.


TIBCO™ iProcess® Workspace (Browser)
User Name:
Password:
Server:
Remember User Name and Server next time I login
Login

The **Login** dialog contains the following fields:

- **User Name** - A user must be established on the TIBCO iProcess Objects Server for the user to be able to login.
- **Password** - A password may or may not be required, depending on how the TIBCO iProcess Objects Server has been configured.

- **Server** - This field presents a drop-down list of available TIBCO iProcess Objects Servers. This list is obtained from the application's `config.xml` file.

Required Data

None — The **Login** component does not require that any data be passed to it to be displayed, that is, it does not need to subscribe to an event of another component.

Properties

The **Login** component contains the following property:

Property	Description
Remember	Enables / disables the **Remember User Id and Server next time I login** check box, which allows users to persist their user name (**User Id** field) and the TIBCO iProcess Objects Server they chose from the **Server** drop-down list.

Events

The **Login** component publishes the following event:

Event	Description
LoginComplete	Indicates a successful login. Event fires when user authentication is complete.

Logout

The **Logout** component provides an interface for logging out of a TIBCO iProcess Objects Server.

This component displays a single toolbar button that allows the user to log out of the application / server.

Required Data

The **Logout** component requires a user name and iProcess Objects Server name to log the user out of the server. This component should subscribe to the “Login Complete” event on the **Login** component.

Properties

None

Events

The **Logout** component publishes the following event:

Event	Description
LogoutComplete	Indicates a logout request has been initiated. Event fires when you click the **Logout** button.

Login / Logout APIs

The Login and Logout components can be directly instantiated without adding the GUI component to an application prototype file.

The following tables show the methods that can be used to provide access to setting names and performing the login and logout actions available with these objects.

WCC Component Methods

Method	Description
setName(*name*)	Sets the name for the component. **Parameters**: • *name* <string> - The component name.
setCompModelName (*name*)	Sets the component model name for the component. This is used in creating the subject for the PageBus events published from the component. **Parameters**: • *name* <string> - The component model name.
setPrototypeModelName (*name*)	Sets the prototype model name for the component. This is used in creating the subject for the PageBus events published from the component. **Parameters**: • *name* <string> - The prototype model name.

Login Methods

Method	Description
doLogin(*app*, *username*, *password*, *servername*, *computername*, *ipaddress*, *tcpport*, *name*, *director*)	Allows direct login without loading the Login GUI object into the display DOM. This method creates an Action Processor request for the login and user profile name value. It validates a successful login and sets the user profile authorizations. If the login fails, a message to the user is displayed with error details. **Parameters**: • *app* <com.tibco.bpm.wcc.Application> - The application instance. • *username*<string> - The user's ID.

Method	Description
	• *password* <string> - The password. • *servername* <string> - The **displayNodeName** value to lookup in `config.xml` that defines the server node **nodeCtx** values. If null, the remainder of the parameters need to be specified, otherwise, they are optional. • *computername* <string> - The iProcess Objects Server name. (optional) • *ipaddress* <string> - The iProcess Objects Server IP address. (optional) • *tcpport* <string> - The iProcess Objects Server TCP port. • *name* <string> - The iProcess Objects Server node name. • *director* <boolean> - Is server a director? **Returns**: <boolean> - true if no socket or iProcess Server Objects errors encountered.
doSingleAuthentication (*app*)	Allows remote login using a single authentication implemented on the server without loading the Login GUI object into the display DOM. This method creates an Action Processor request for a remote login. It validates a successful login and sets the user profile authorizations. If login fails, a message to the user is displayed with error details. **Parameters**: • *app* <com.tibco.bpm.wcc.Application> - The application instance.

Login Example

This example shows how a direct login could be made using the **doLogin** method. In this example, the **wcc.Login** instance could have been added using the **appMain** prototype (with paintable = jsx3.Boolean.FALSE) or created directly as shown. This sample might be added to the **postLoadInit()** method defined in:

— `com.tibco.bpm.wccAppCustom.Application`

```
  // Test doLogin at application load.  Timeout call is used to allow
  // the appMain prototype to load.  If login found in prototype, then
  // it is used, else an instance is created.
     var app = this;
     window.setTimeout(function(){
         var login = jsx3.GO('wccLogin');
         if (! login) {
             login = new com.tibco.bpm.wcc.Login();
             // Allow the prototypeModelName to be specified for a
             // component so that if it is not added to the DOM, this
             // value is directly specified.  For example, wcc.Login can
             // be instantiated without a GUI component to call doLogin.
             // This requires that prototypeModelName is set directly.
             login.setPrototypeModelName('wccPrototype');
             login.setName('wccLogin');
             login.setCompModelName('wccLogin');
         }
         else{
            //Set login.paintable=false -- you are performing direct
login
         login.paintable=false;
         }
         login.doLogin(app, 'username', 'password', 'iProcess Node
servername');
     }, 100);
*/
```

Logout Method

Method	Description
doLogout()	Logs the current user out of an iProcess Objects Server. It publishes the **logoutRequest** event before processing the logout request. **Parameters**: • none **Returns**: <[object]> - reference to self

Logout Example

To use the Logout component directly as an object and not as a GUI component:

- Create the **wcc.Logout** instance.
- Set the **app** property to the current application instance.
- Call the **doLogout()** method.
- Call the **postLogoutComplete()** method.

Note: No subscription is set up for **logoutComplete** to call this.

```
var logout = new com.tibco.bpm.wcc.Logout();
var appNameSpace = 'wccAppNew';
var ipcApp = com.tibco.bpm.ipc.apps[appNameSpace];
// Note: The app value needs to be set on logout if used directly.
logout.app = ipcApp;
logout.doLogout();
logout.postLogoutComplete();
```

Case Management

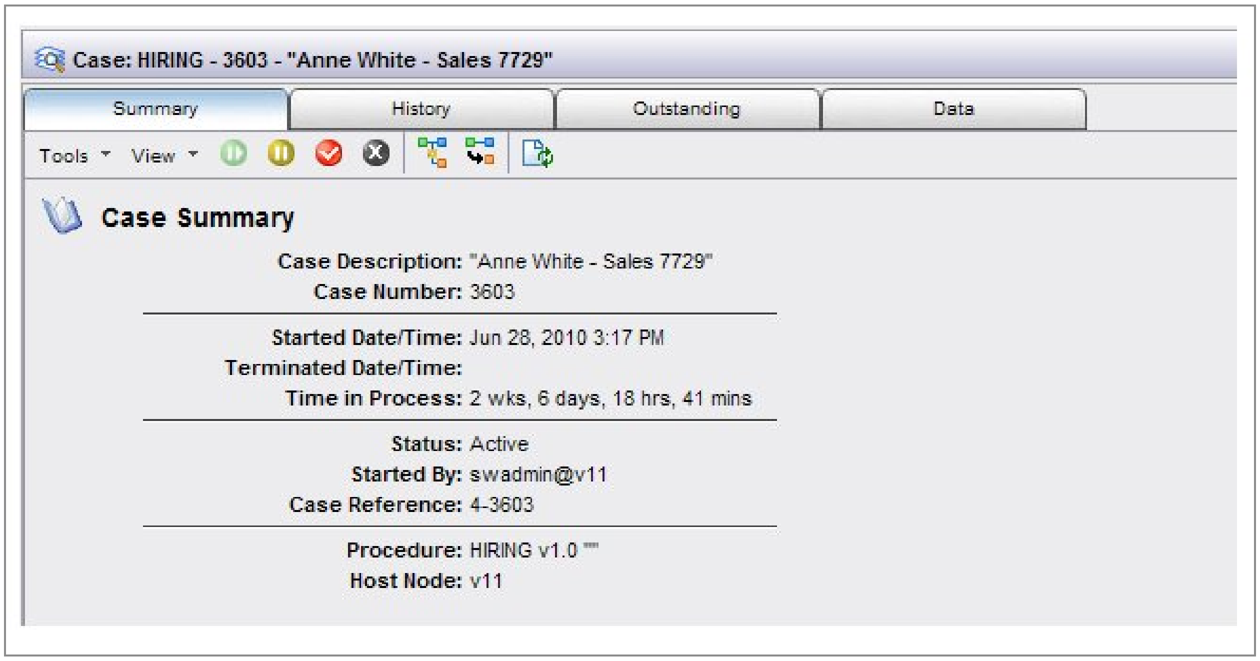
The following are the Case Management components available:

- CaseDetail - displays all four of the other Case Management components in a tabbed format
- CaseSummary - displays a summary of the case
- CaseHistory - displays the history of a case
- CaseOutstanding - displays a list of the outstanding work items in the case
- CaseData - displays the current case data values for the fields in the selected case

These are described in the following subsections.

CaseDetail

The **CaseDetail** component combines all four of the other Case Management components into a single component. This component displays the other four components in a tabbed format.



Case Summary

Case Description: "Anne White - Sales 7729"
Case Number: 3603

Started Date/Time: Jun 28, 2010 3:17 PM
Terminated Date/Time:
Time in Process: 2 wks, 6 days, 18 hrs, 41 mins

Status: Active
Started By: swadmin@v11
Case Reference: 4-3603

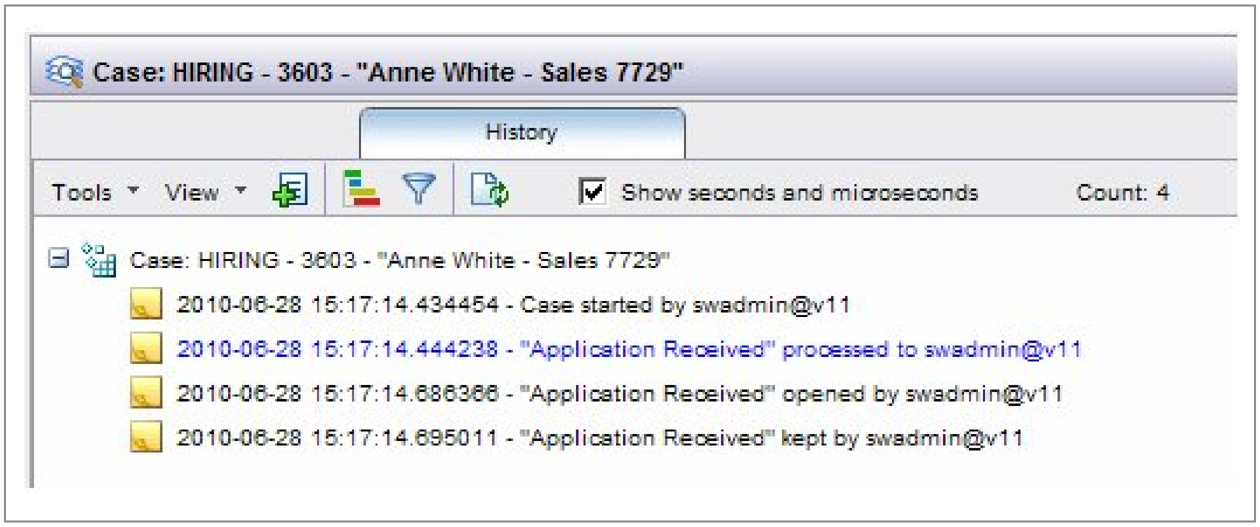
Procedure: HIRING v1.0 ""
Host Node: v11

For more information about the functions on each of the tabs displayed by the **CaseDetail** component, see the subsection for that particular tab/component.

Note: The case detail tabs are also displayed when a case is “opened” from the case list. The **CaseDetail** component is available to allow you to display the case detail tabs separately from opening a case from the case list.

You can control, using the **CaseDetail** properties, which of the four tabbed components you would like to display. For example, you can specify that only the **Summary** and **History** tabs display by setting the appropriate properties.

Note: If you display the **CaseDetail** component, but the users' access privileges only allow them to view a single tab (case history, for example), the tab for that single component is displayed.


Case: HIRING - 3603 - "Anne White - Sales 7729"
History
Tools
View
Show seconds and microseconds
Count: 4

Case: HIRING - 3603 - "Anne White - Sales 7729"
2010-06-28 15:17:14.434454 - Case started by swadmin@v11
2010-06-28 15:17:14.444238 - "Application Received" processed to swadmin@v11
2010-06-28 15:17:14.686366 - "Application Received" opened by swadmin@v11
2010-06-28 15:17:14.695011 - "Application Received" kept by swadmin@v11

Required Data

The **CaseDetail** component requires a **case tag** to be able to be displayed.

Subscribe to: **Cases** component:

- List Item Select (single click) event
- List Item Execute (double click) event

CaseDetail Component Properties

The **CaseDetail** component contains the following properties, which are used to control access to the functions on the tabs that are displayed by the **CaseDetail** component.

The properties are shown in the table in the same order in which they are listed in the **Component Hierarchy**.

Property	Applies to...	Description
Case	All tabs	Enables/disables access to the case detail tabs. This property controls rendering of the case detail tabs. This would be used to control access from an external application, rather than from an event from another component (such as the case list).

Property	Applies to...	Description
Activate Case	Summary tab	Enables/disables the ability to activate a suspended case. This property controls access to the **Activate Case(s)** function on the **Summary** tab.
Close Case	Summary tab	Enables/disables the ability to close a case. This property controls access to the **Close Case(s)** function on the **Summary** tab.
Process Jump	Summary tab	Enables/disables the ability to jump to a new outstanding step in the case. This property controls access to the **Process Jump** function on the **Summary** tab.
Process Jump - View Fields	Summary tab	Enables/disables read-only access to the **Case Data** dialog available through the **Process Jump** dialog box. If the **Process Jump** / **Update Fields** property (see below) is enabled, this property is ignored and view access is always granted. If both this and the **Process Jump** / **Update Fields** property are disabled, the **Data** button is not displayed on the **Process Jump** dialog box.
Process Jump - Update Fields	Summary tab	Enables/disables update access to the **Case Data** dialog available through the **Process Jump** dialog box. This property overrides the **Process Jump** / **View Fields** property (see above) if it is enabled. If both this and the **Process Jump** / **View Fields** property are disabled, the **Data** button is not displayed on the **Process Jump** dialog box.
Process Jump - Select Outstanding	Summary tab	Enables/disables the ability to specify which columns to display in the outstanding items list on the **Process Jump** dialog box.

Property	Applies to...	Description
Items Columns		This property controls access to the **Select Columns** selection on the **View** menu for the outstanding items list on the **Process Jump** dialog box.
Suspend Case	Summary tab	Enables/disables the ability to suspend a case. This property controls access to the **Suspend Case(s)** function on the **Summary** tab.
Event Trigger	Summary tab	Enables/disables the ability to trigger an event in a case. This property controls access to the **Trigger Event** function on the **Summary** tab.
Event Trigger - View Fields	Summary tab	Enables/disables read-only access to the **Case Data** dialog available through the **Events** dialog box. If the **Trigger Event** / **Update Fields** property (see below) is enabled, this property is ignored and view access is always granted. If both this and the **Trigger Event** / **Update Fields** property are disabled, the **Data** button is not displayed on the **Events** dialog box.
Event Trigger - Update Fields	Summary tab	Enables/disables update access to the **Case Data** dialog available through the **Events** dialog box. This property overrides the **Trigger Event** / **View Fields** property (see above) if it is enabled. If both this and the **Trigger Event** / **View Fields** property are disabled, the **Data** button is not displayed on the **Events** dialog box.
Event Trigger - Resurrect Case	Summary tab	Enables/disables the ability to resurrect (reactivate) a closed case. This property controls access to the **Trigger Event** function on the **Summary** tab when a closed case is

Property	Applies to...	Description
		selected.
Event Trigger - Recalculate Deadlines	Summary tab	Enables/disables the ability to recalculate deadlines in the case. This property controls access to the **Recalculate Deadlines** radio buttons on the **Events** dialog box.
Purge Case	Summary tab	Enables/disables the ability to purge (permanently delete) cases. This property controls access to the **Purge Case(s)** function on the **Summary** tab.
Open Case	All tabs	Enables/disables the ability to open the case detail window. This property controls access to all of the tabs on the case detail window, i.e., if this property is disabled, none of the case detail tabs are displayed.
Open Case - Case Summary	Summary tab	Enables/disables the ability to access the **Summary** tab. If this property is disabled, the **Summary** tab is not displayed in the case details. Also note that if this property is disabled, it causes all of the other **Summary** tab-related properties shown above in this table to also be disabled (although the other **Summary** tab-related properties don’t show a red ‘x’ when the **Case Summary** property is disabled).
Open Case - Case History	History tab	Enables/disables the ability to access the **History** tab. If this property is disabled, the **History** tab is not displayed in the case details.
Open Case - Case History	History tab	Enables/disables the ability to manually add an entry to the case history.

Property	Applies to...	Description
Add History Entry		This property controls access to the **Add Entry** function on the **History** tab.
Open Case - Case History - Case Predict	History tab	Enables/disables the ability to perform a case prediction function. This property controls access to the **Predict Case** function on the **History** tab.
Open Case - Case History - Graphical Case History	History tab	Enables/disables the ability to view the case history in a graphical format. This property controls access to the **Graphical History** function on the **History** tab.
Open Case - Outstanding Items	Outstanding tab	Enables/disables the ability to access the **Outstanding** tab. If this property is disabled, the **Outstanding** tab is not displayed in the case details.
Open Case - Outstanding Items - Select Columns	Outstanding tab	Enables/disables the ability to specify which columns to display in the outstanding items list on the **Outstanding** tab. This property controls access to the **Select Columns** selection on the **Outstanding** tab **View** menu.
Open Case - View Fields	Data tab	Enables/disables read-only access to the **Data** tab. If the **Open Case / Update Fields** property (see below) is enabled, this property is ignored and view access is always granted. If both this and the **Open Case / Update Fields** property are disabled, the case **Data** tab is not displayed.
Open Case	Data tab	Enables/disables update access to the **Data** tab.

Property	Applies to...	Description
Update Fields		This property overrides the **Open Case** / **View Fields** property (see above) if it is enabled. If both this and the **Open Case** / **View Fields** property are disabled, the case **Data** tab is not displayed.

CaseDetail Component Events

The CaseDetail component publishes the following events, which fire when the action described by the event occurs.

Applies to...	Event	Description
Summary tab	Process Jump	Indicates the user is performing a process jump function. Fires when the user clicks the **Process Jump** button on the **Summary** tab toolbar, or chooses the **Process Jump** selection on the **Summary** tab **Tools** menu.
	Trigger Event	Indicates the user is performing a trigger event function. Fires when the user clicks the **Trigger Events** button on the **Summary** tab toolbar, or chooses the **Trigger Events** selection on the **Summary** tab **Tools** menu.
	Purge Case	Indicates the user is purging (permanently deleting) the case. Fires when the user clicks the **Purge Case** button on the **Summary** tab toolbar, or chooses the **Purge Case** selection on the **Summary** tab **Tools** menu.
	Close Case	Indicates the user is closing the case. Fires when the user clicks the **Close Case** button on the **Summary** tab toolbar, or chooses the **Close Case** selection on the **Summary** tab **Tools** menu.
	Suspend Case	Indicates the user is suspending the case.

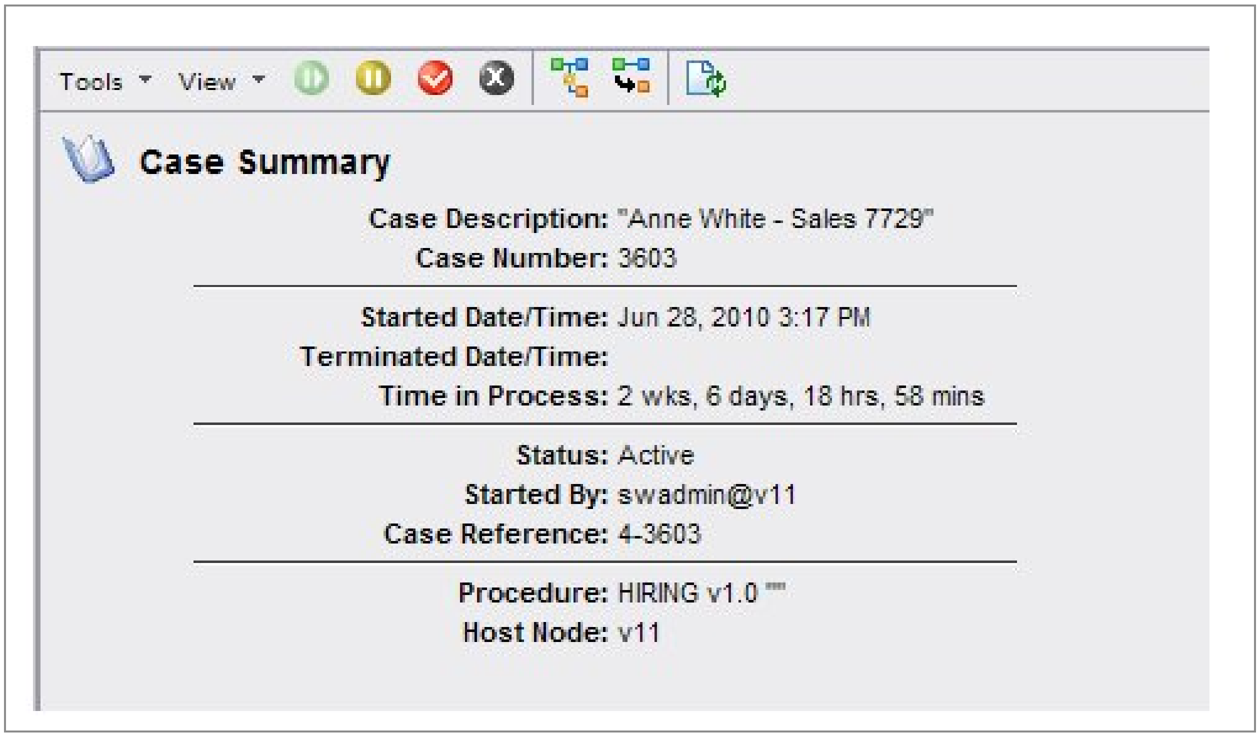
Applies to...	Event	Description
		Fires when the user clicks the **Suspend Case** button on the **Summary** tab toolbar, or chooses the **Suspend Case** selection on the **Summary** tab **Tools** menu.
	Activate Case	Indicates the user is activating the suspended case. Fires when the user clicks the **Activate Case** button on the **Summary** tab toolbar, or chooses the **Activate Case** selection on the **Summary** tab **Tools** menu. (Only applicable when the case is suspended.)
History tab	View Graphical Case History	Indicates the user is viewing the case's history in a graphical format. Fires when the user clicks the **Graphical History** button on the **History** tab toolbar, or chooses the **Graphical History** selection on the **History** tab **View** menu.
	Case Prediction	Indicates the user is performing a case prediction operation. Fires when the user clicks the **Predict Case** button on the **History** tab toolbar, or chooses the **Predict Case** selection on the **History** tab **Tools** menu.
	Add History	Indicates the user is manually adding an entry to the case history. Fires when the user clicks the **Add Entry** button on the **History** tab toolbar, or chooses the **Add Entry** selection on the **History** tab **Tools** menu.
	Select Case History (single click)	Indicates the user has selected an entry in the case history. Fires when the user single-clicks a case history entry on the **History** tab.
	Execute Case	Indicates the user has executed an entry in the case

Applies to...	Event	Description
	History (double click)	history. Fires when the user double-clicks a case history entry on the **History** tab. This event also fires when the user presses **Enter** when a case history entry is already selected (highlighted).
Outstanding tab	Select Case Outstanding (single click)	Indicates the user has selected an outstanding work item. Fires when the user single-clicks an outstanding work item on the **Outstanding** tab.
	Execute Case Outstanding (double click)	Indicates the user has executed an outstanding work item. Fires when the user double-clicks an outstanding work item on the **Outstanding** tab. This event also fires when the user presses **Enter** when an outstanding work item is already selected (highlighted).
	Open Case Outstanding Work Item(s)	Indicates the user has opened an outstanding work item. Fires when the user clicks the **Open Selected Work Item (s)** button on the **Outstanding** tab toolbar, or chooses the **Open Selected Work Item(s)** selection on the **Outstanding** tab **Tools** menu.
Data tab	Case Data Update	Indicates the user has updated case data. Fires when the user clicks on the **Apply** button on the **Data** tab.

CaseSummary

The **CaseSummary** component displays summary information about a specific case. It also provides access to functions to perform operations on the case.

An example case summary window is illustrated below.



Case Summary

Case Description: "Anne White - Sales 7729"
Case Number: 3603

Started Date/Time: Jun 28, 2010 3:17 PM
Terminated Date/Time:
Time in Process: 2 wks, 6 days, 18 hrs, 58 mins

Status: Active
Started By: swadmin@v11
Case Reference: 4-3603

Procedure: HIRING v1.0 ""
Host Node: v11

The following describes the functions available from the case summary window:

- **Activate Case** - Reactivates the current case; the case must be suspended for this function to be active.
- **Suspend Case** - Suspends activity in the current case and all of its sub-cases (if applicable). A suspended case can be reactivated using the **Activate Case** button.
- **Close Case** - Stops the process flow for the current case.
- **Purge Case** - Permanently deletes the current case from the system.
- **Trigger Event** - Starts the process flow from an Event step in the current case.
- **Process Jump** - Used to withdraw currently outstanding steps from the case, and jump to new outstanding steps.
- **Refresh Case** - Requests that new data be retrieved from the server, refreshing the case summary. (Note - Access to this function is not controlled by a property, nor does it fire an event when selected.)

If the currently signed on user does not have access privileges to a particular function, the icon/menu selection for that function does not display. Access to each of the functions available on the case summary window is controlled by properties on the **CaseSummary** component. For more information about setting these properties, see CaseSummary Component Properties.

The **CaseSummary** component also publishes events for many of the functions available from the case summary window. For more information about these events, see CaseSummary Component Events.

For more information about using the functions listed above (e.g., activating a case, closing a case, etc.), see the *TIBCO iProcess Workspace (Browser) User Guide.*

Required Data

The **CaseSummary** component requires a **case tag** to be able to be displayed.

Subscribe to: **Cases** component:

- List Item Select (single click) event
- List Item Execute (double click) event

CaseSummary Component Properties

The **CaseSummary** component contains the following properties, which are used to control access to each of the functions available from the case summary window.

Property	Description
Case	Enables/disables access to the case summary window. This property controls rendering of the case summary window. This would be used to control access from an external application, rather than from an event from another component (such as the case list).
Activate Case	Enables/disables the ability to activate the suspended case. This property controls access to the **Activate Case** function on the case summary window.
Close Case	Enables/disables the ability to close the case. This property controls access to the **Close Case** function on the case summary window.
Process Jump	Enables/disables the ability to jump to a new outstanding step in the case.

Property	Description
	This property controls access to the **Process Jump** function on the case summary window.
Process Jump - View Fields	Enables/disables read-only access to the **Case Data** dialog available through the **Process Jump** dialog box. If the **Process Jump** / **Update Fields** property (see below) is also enabled, it overrides this property, giving the user update access to case data. If both this and the **Process Jump** / **Update Fields** property are disabled, the **Data** button is not displayed on the **Process Jump** dialog box.
Process Jump - Update Fields	Enables/disables update access to the **Case Data** dialog available through the **Process Jump** dialog box. This property overrides the **Process Jump** / **View Fields** property (see above) if it is enabled. If both this and the **Process Jump** / **View Fields** property are disabled, the **Data** button is not displayed on the **Process Jump** dialog box.
Process Jump - Select Outstanding Items Columns	Enables/disables the ability to specify which columns to display in the outstanding items list on the **Process Jump** dialog box. This property controls access to the **Select Columns** selection on the **View** menu for the outstanding items list on the **Process Jump** dialog box.
Suspend Case	Enables/disables the ability to suspend a case. This property controls access to the **Suspend Case** function on the case summary window.
Event Trigger	Enables/disables the ability to trigger an event in the case. This property controls access to the **Trigger Event** function on the case summary window.
Event Trigger - View Fields	Enables/disables read-only access to the **Case Data** dialog available through the **Events** dialog box.

Property	Description
	If the **Trigger Event** / **Update Fields** property (see below) is also enabled, it overrides this property, giving the user update access to case data. If both this and the **Trigger Event** / **Update Fields** property are disabled, the **Data** button is not displayed on the **Events** dialog box.
Event Trigger - Update Fields	Enables/disables update access to the **Case Data** dialog available through the **Events** dialog box. This property overrides the **Trigger Event** / **View Fields** property (see above) if it is enabled. If both this and the **Trigger Event** / **View Fields** property are disabled, the **Data** button is not displayed on the **Events** dialog box.
Event Trigger - Resurrect Case	Enables/disables the ability to resurrect (reactivate) the closed case. This property controls access to the **Trigger Event** function on the case summary window. (Only applicable when the case is closed.)
Event Trigger - Recalculate Deadlines	Enables/disables the ability to recalculate deadlines in the case. This property controls access to the **Recalculate Deadlines** radio buttons on the **Events** dialog box.
Purge Case	Enables/disables the ability to purge (permanently delete) the case. This property controls access to the **Purge Case** function on the case summary window.
Open Case	Enables/disables the ability to open the case summary window. **Note:** There is no “open case” function on the case summary window — this property is a result of the component properties mirroring the user access profiles. This property performs the same function as the **Case Summary** property (see below).
Case Summary	Enables/disables the ability to access the case summary window.

Property	Description
	Also note that if this property is disabled, it causes all of the other case summary window-related properties shown above in this table to also be disabled (however the other case summary window-related properties don’t show a red ‘x’ when the **Case Summary** property is disabled).

CaseSummary Component Events

The **CaseSummary** component publishes the following events, which fire when the action described by the event occurs.

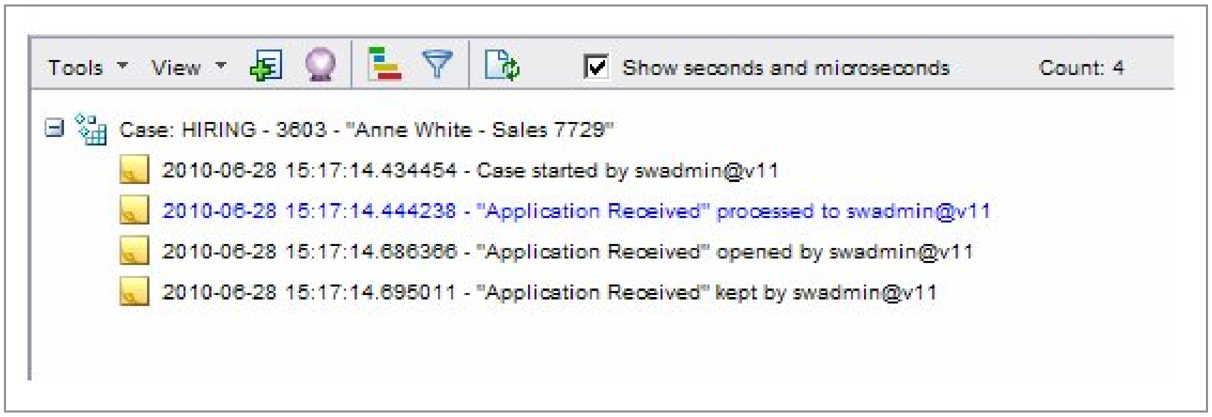
Event	Description
Process Jump	Indicates the user is performing a process jump function. Fires when the user clicks the **Process Jump** button on the case summary window toolbar, or chooses the **Process Jump** selection on the case summary window **Tools** menu.
Trigger Event	Indicates the user is performing a trigger event function. Fires when the user clicks the **Trigger Event** button on the case summary window toolbar, or chooses the **Trigger Event** selection on the case summary window **Tools** menu.
Purge Case	Indicates the user is purging (permanently deleting) the case. Fires when the user clicks the **Purge Case** button on the case summary window toolbar, or chooses the **Purge Case** selection on the case summary window **Tools** menu.
Close Case	Indicates the user is closing the case. Fires when the user clicks the **Close Case** button on the case summary window toolbar, or chooses the **Purge Case** selection on the case summary window **Tools** menu.
Suspend Case	Indicates the user is suspending the case. Fires when the user clicks the **Suspend Case** button on the case summary

Event	Description
	window toolbar, or chooses the **Suspend Case** selection on the case summary window **Tools** menu.
Activate Case	Indicates the user is activating the suspended case. Fires when the user clicks the **Activate Case** button on the case summary window toolbar, or chooses the **Activate Case** selection on the case summary window **Tools** menu. (Only applicable when the case is suspended.)

CaseHistory

The **CaseHistory** component displays information about the case's chronological progress through the procedure, that is, it shows you which steps (work items) have been processed and who processed them. This is sometimes referred to as the case "audit trail."

An example case history is illustrated below.



Case: HIRING - 3603 - "Anne White - Sales 7729"
2010-06-28 15:17:14.434454 - Case started by swadmin@v11
2010-06-28 15:17:14.444238 - "Application Received" processed to swadmin@v11
2010-06-28 15:17:14.686366 - "Application Received" opened by swadmin@v11
2010-06-28 15:17:14.695011 - "Application Received" kept by swadmin@v11

The following describes the functions available from the case history window:

- **Add Entry** - Allows the user to make a manual entry to the case's history.
- **Predict Case** - Provides the means for predicting the expected outcome of a live case. Running the case prediction function causes a list of "predicted work items" to be returned that represent the work items that are currently due (outstanding work items), as well as the work items that are expected to be due in the future.
- **Graphical History** - Displays the case history in a graphical time-line chart.

- **Filter Case History** - Allows the user to filter the case history entries. This is useful when the number of case history entries become very large — it allows you to more easily find the desired entry.
- **Refresh History** - Requests that new data be retrieved from the server, refreshing the case history. (Note - Access to this function is not controlled by a property, nor does it fire an event when selected.)

If the currently signed on user does not have access privileges to a particular function, the icon/menu selection for that function does not display. Access to each of the functions available on the case history window is controlled by properties on the **CaseHistory** component. For more information about setting these properties, see CaseHistory Component Properties.

The **CaseHistory** component also publishes events for many of the functions available from the case history window. For more information about these events, see CaseHistory Component Events.

For more information about using the functions listed above (e.g., viewing graphical case history, predicting a case, etc.), see the *TIBCO iProcess Workspace (Browser) User Guide.*

Required Data

The **CaseHistory** component requires a **case tag** to be able to be displayed.

Subscribe to: **Cases** component:

— List Item Select (single click) event

— List Item Execute (double click) event

CaseHistory Component Properties

The **CaseHistory** component contains the following properties, which are used to control access to each of the functions available from the case history window.

Property	Description
Open Case	Enables/disables the ability to access the case history window.

Property	Description
	Note: There is no “open case” function on the case history window — this property is a result of the component properties mirroring the user access profiles. This property performs the same function as the **Case History** property (see below).
Case History	Enables/disables the ability to access the case history window.
Add History Entry	Enables/disables the ability to manually add an entry to the case history. This property controls access to the **Add Entry** function on the case history window.
Case Predict	Enables/disables the ability to perform a case prediction function. This property controls access to the **Predict Case** function on the case history window.
Graphical Case History	Enables/disables the ability to view the case history in a graphical format. This property controls access to the **Graphical History** function on the case history window.
Filter Case History	Enables/disables the ability to filter the list of case history entries. This property controls access to the **Filter Case History** function on the case history window.

CaseHistory Component Events

The **CaseHistory** component publishes the following events, which fire when the action described by the event occurs.

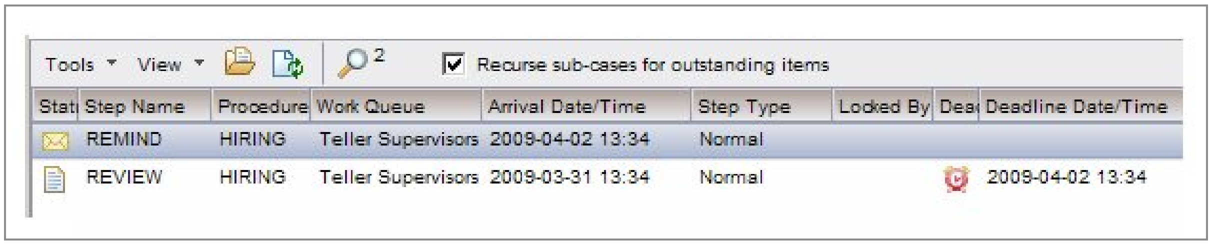
Event	Description
View Graphical Case History	Indicates the user is viewing the case’s history in a graphical format. Fires when the user clicks the **Graphical History** button on the case history

Event	Description
	window toolbar, or selects the **Graphical History** selection on the case history window **View** menu.
Case Prediction	Indicates the user is performing a case prediction operation. Fires when the user clicks the **Predict Case** button on the case history window toolbar, or chooses the **Predict Case** selection on the case history window **Tools** menu.
Add History	Indicates the user is manually adding an entry to the case history. Fires when the user clicks the **Add Entry** button on the case history window toolbar, or chooses the **Add Entry** selection on the case history window **Tools** menu.
Select Case History (single click)	Indicates the user has selected an entry in the case history. Fires when the user single-clicks a case history entry on the case history window.
Execute Case History (double click)	Indicates the user has executed an entry in the case history. Fires when the user double-clicks a case history entry on the case history window. This event also fires when the user presses **Enter** when a case history entry is already selected (highlighted).

CaseOutstanding

The **CaseOutstanding** component displays a list of all of the “outstanding” work items in the case. Outstanding work items represent the steps at which the process flow is currently sitting.

An example outstanding work item list is shown below:



☑ Recurse sub-cases for outstanding items

Stat	Step Name	Procedure	Work Queue	Arrival Date/Time	Step Type	Locked By	Dea	Deadline Date/Time
	REMIND	HIRING	Teller Supervisors	2009-04-02 13:34	Normal			
	REVIEW	HIRING	Teller Supervisors	2009-03-31 13:34	Normal			2009-04-02 13:34

Required Data

The **CaseOutstanding** component requires a **case tag** to be able to be displayed.

Subscribe to: **Cases** component:

- List Item Select (single click) event
- List Item Execute (double click) event

CaseOutstanding Component Properties

The **CaseOutstanding** component contains the following properties, which are used to control access to the functions available from the outstanding work item list.

Property	Description
Open Case	Enables/disables access to the outstanding work item list. **Note:** There is no “open case” function on the outstanding work item list — this property is a result of the component properties mirroring the user access profiles. This property performs the same function as the **Outstanding Items** property.
Outstanding Items	Enables/disables access to the outstanding work item list.
Select Columns	Enables/disables the ability to specify which columns to display in the outstanding work item list. This property controls access to the **Select Columns** selection on the outstanding work item list **View** menu.

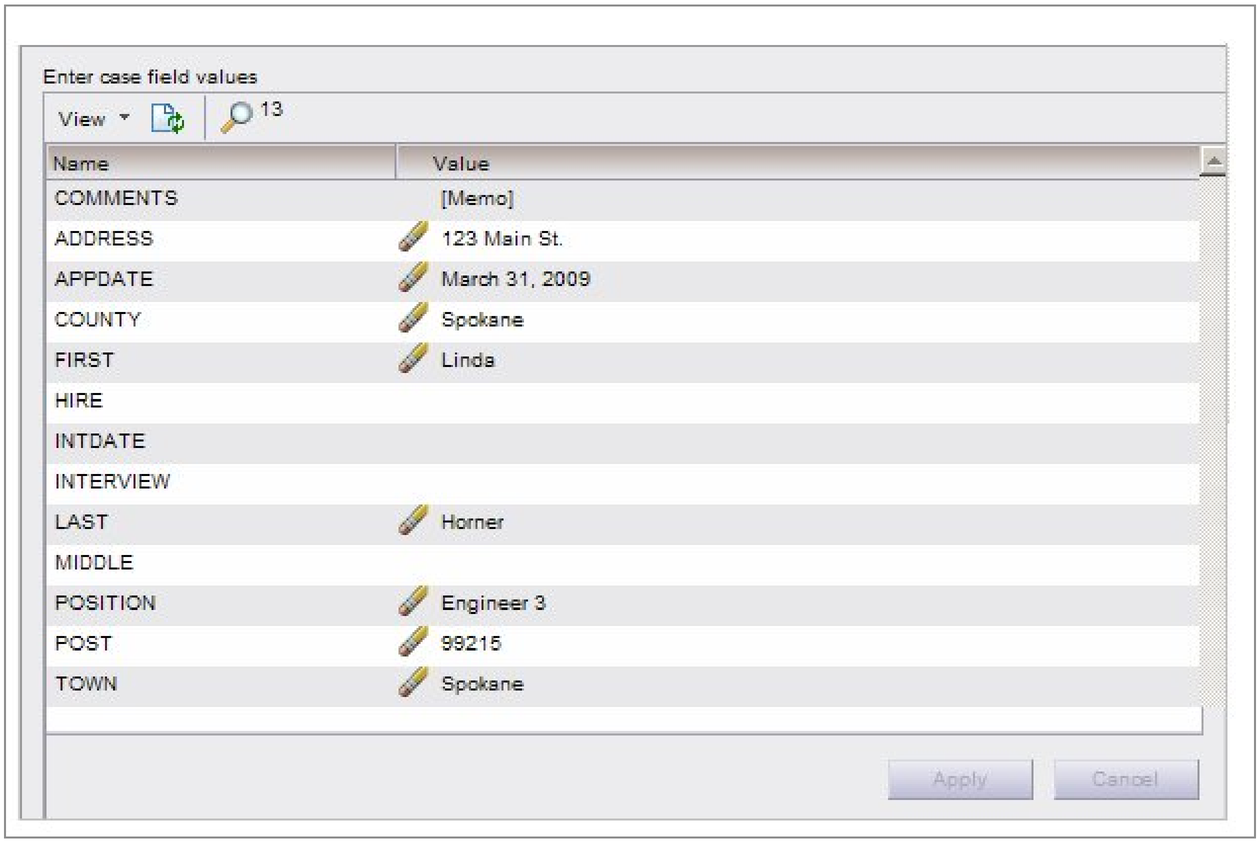
CaseOutstanding Component Events

The **CaseOutstanding** component publishes the following events, which fire when the action described by the event occurs.

Event	Description
Select Case Outstanding (single click)	Indicates the user has selected an outstanding work item. Fires when the user single-clicks a work item on the outstanding work item list.
Execute Case Outstanding (double click)	Indicates the user has executed an outstanding work item. Fires when the user double-clicks a work item on the outstanding work item list. This event also fires when the user presses **Enter** when an outstanding work item is already selected (highlighted).
Open Case Outstanding Work Item(s)	Indicates the user has opened an outstanding work item. Fires when the user clicks the **Open Selected Work Item(s)** button, or chooses the **Open Selected Work Item(s)** selection on the **Tools** menu.

CaseData

The **CaseData** component displays a window that shows the current case data values for the fields in the selected case. For example:



Name	Value
COMMENTS	[Memo]
ADDRESS	123 Main St.
APPDATE	March 31, 2009
COUNTY	Spokane
FIRST	Linda
HIRE	
INTDATE	
INTERVIEW	
LAST	Horner
MIDDLE	
POSITION	Engineer 3
POST	99215
TOWN	Spokane

From this window, the user can view and/or update the case data, depending on the user’s access privileges.

Required Data

The **CaseData** component requires a **case tag** to be able to be displayed.

Subscribe to: **Cases** component:

- List Item Select (single click) event
- List Item Execute (double click) event

CaseData Component Properties

The **CaseData** component contains the following properties, which are used to control access to viewing/updating the values in the fields on the case data window.

Property	Description
Open Case	Enables/disables access to the case data window. **Note:** There is no “open case” function on the case data window — this property is a result of the component properties mirroring the user access profiles. Disabling this property prevents the user from opening the case data window.
View Fields	Enables/disables read-only access to the case data window. If the **Update Fields** property (see below) is also enabled, it overrides this property, giving the user update access to case data. If both this and the **Update Fields** property are disabled, the case data window is not displayed.
Update Fields	Enables/disables update access to the case data window. This property overrides the **View Fields** property (see above) if it is enabled. If both this and the **View Fields** property are disabled, the case data window is not displayed.

CaseData Component Events

The **CaseData** component publishes the following events, which fire when the action described by the event occurs.

Event	Description
Case Data Update	Indicates the user is updating the value of the case data. Fires when the user clicks the **Apply** button after making a change on the case data window.

Composites

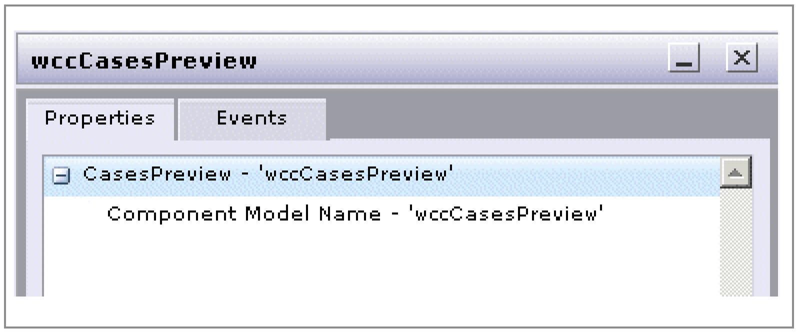
The composite components combine more than one component into a single component to provide an easy way to display multiple lists or elements without the need to add multiple components to your application, or to set up the events between those components — the events between the components provided in the composites are already configured.

The following are the composite components available:

- CasesPreview - composite of the case list component (see Cases) and the preview pane
- ProcsCases - composite of the procedure list component (**StartHistoryProcs**) — see Procedures) and the case list component (see Cases)
- WorkItemsPreview - composite of the work item list component (see WorkItems) and the preview pane
- WorkQsItems - composite of the work queue list component (see UserWorkQs) and the work item list component (see WorkItems)

Composite Properties

The composite components do not have properties that can be configured in the Properties Editor. Displaying the Properties Editor for a composite shows that it does not have its own properties:



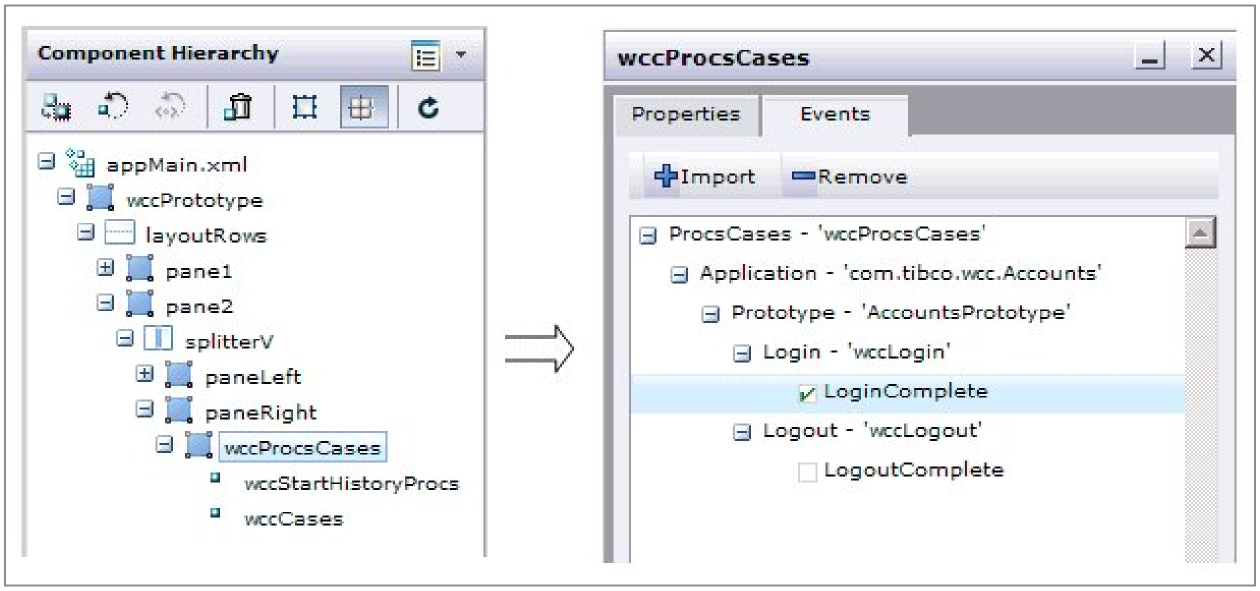
CasesPreview - 'wccCasesPreview'
Component Model Name - 'wccCasesPreview'

To configure access to the functions available from each component in a composite, you must drill down to the desired component under the composite in the **Component Hierarchy**, then configure the appropriate property.

Composite Events

When configuring events for a composite component, the composite component itself should subscribe to events, not the components that are included in the composite.

For example, the **ProcsCases** composite component needs to subscribe to an event to cause it to display, as follows:

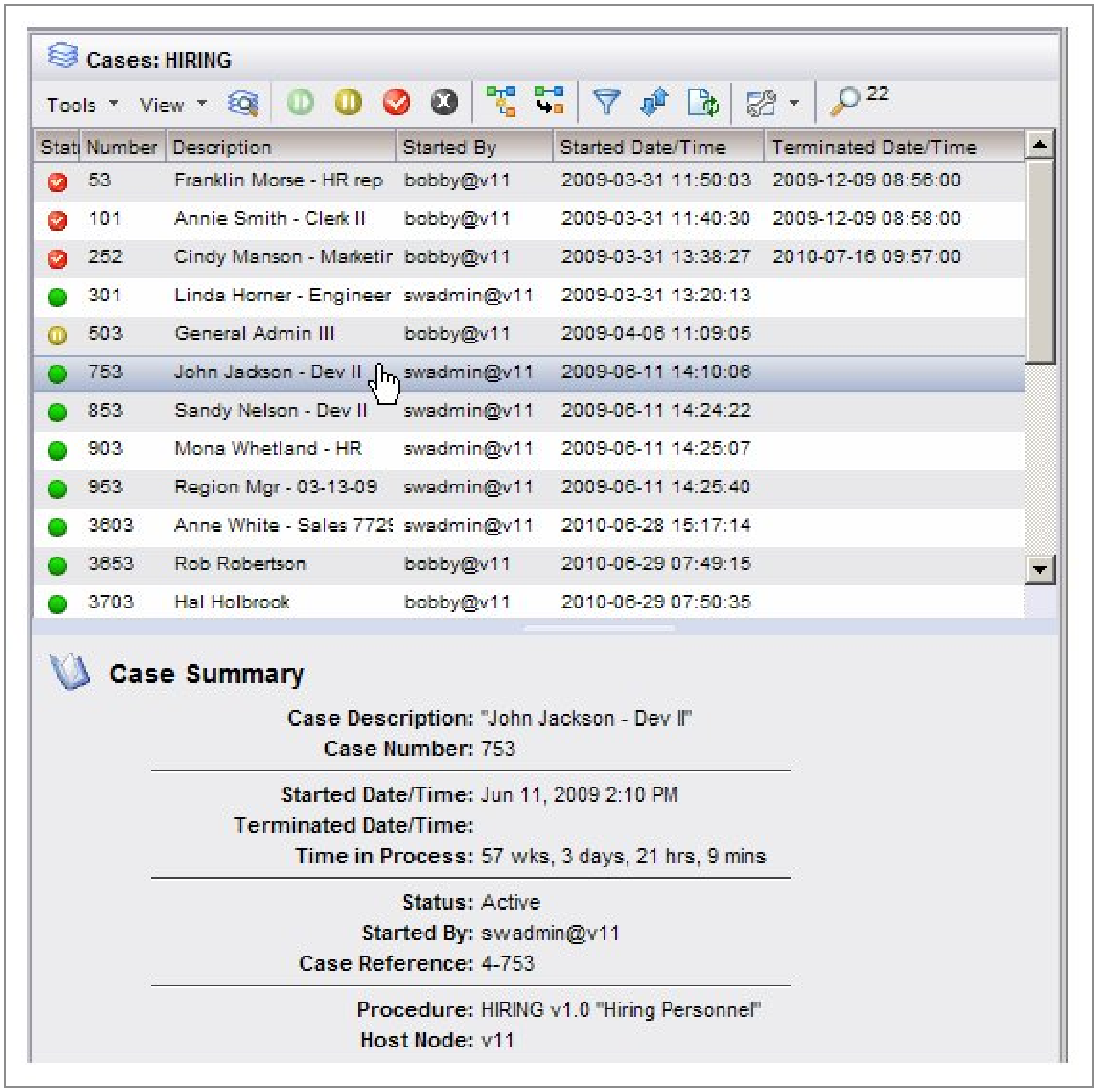


Note: You cannot drill down to the composite's underlying components in the **Component Hierarchy** and set up event subscriptions for those components because the **Event** tab for the underlying components is disabled (although other components may want to subscribe to the events published by the underlying components).

CasesPreview

The **CasesPreview** component is a composite of the case list component (Cases) and the preview pane. (This is considered a composite, although technically it does not comprise two components that you can see in the **Component Hierarchy**; in this context, the preview pane is considered a component.)

When a case is selected (single click) in the case list displayed by this component, the case summary is displayed in the preview pane. When a case is opened (double click, or select the **Open Case** function), the case detail tabs are displayed in the preview pane. Example:



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 08:56:00
✓	101	Annie Smith - Clerk II	bobby@v11	2009-03-31 11:40:30	2009-12-09 08:58:00
✓	252	Cindy Manson - Marketir	bobby@v11	2009-03-31 13:38:27	2010-07-16 09:57:00
●	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 772	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	

Case Summary

Case Description: "John Jackson - Dev II"
Case Number: 753

Started Date/Time: Jun 11, 2009 2:10 PM
Terminated Date/Time:
Time in Process: 57 wks, 3 days, 21 hrs, 9 mins

Status: Active
Started By: swadmin@v11
Case Reference: 4-753

Procedure: HIRING v1.0 "Hiring Personnel"
Host Node: v11

The case detail tabs displayed are the same tabs that can be displayed using the **CaseDetail** component (see CaseDetail). The example illustrated above shows all four available tabs, although they may not all display, depending on the user's access privileges.

The case list that is displayed using the **CasesPreview** component also contains the **Preview** icon, as well as the **Preview** menu selections on the **View** menu to allow you to display the case details in a floating window. (Note that the **Cases** component, which also displays the case list, always opens the case details in a floating window.)

Required Data

The **CasesPreview** component requires a **procedure tag** to be able to be displayed.

Subscribe to: **HistoryProcs**, **NodeProcs**, **StartHistoryProcs**, or **StartProcs** component:

— List Item Select (single click) event

— List Item Execute (double click) event

CasesPreview Component Properties

None — composite components do not have their own properties. To set the properties for the individual components that are part of the composite, click on the underlying component in the **Component Hierarchy**, then use the Properties Editor to set access for functions provided by that component.

For more information about the properties available for the components included with **CasesPreview**, see:

- Cases Component Properties

Note: The **Cases** component is the only underlying component in **CasesPreview**. The case details that appear in the preview pane are not shown as a component in the **Component Hierarchy**. Therefore, you cannot separately configure access to the functions on the case details alone. When you configure access to functions using the properties on the **Cases** component under the **CasesPreview** component, you are configuring access to the functions on both the case list and the case details.

CasesPreview Component Events

None — composite components do not publish events themselves, although the individual components that comprise the composite publish events. Therefore, other components cannot subscribe to events of a composite, but they can subscribe to events of the underlying components.

For more information about the events published by the components included with **CasesPreview**, see:

- Cases Component Events

Note: The **Cases** component events include the events published by case detail tabs displayed when a case is opened from the case list.

ProcsCases

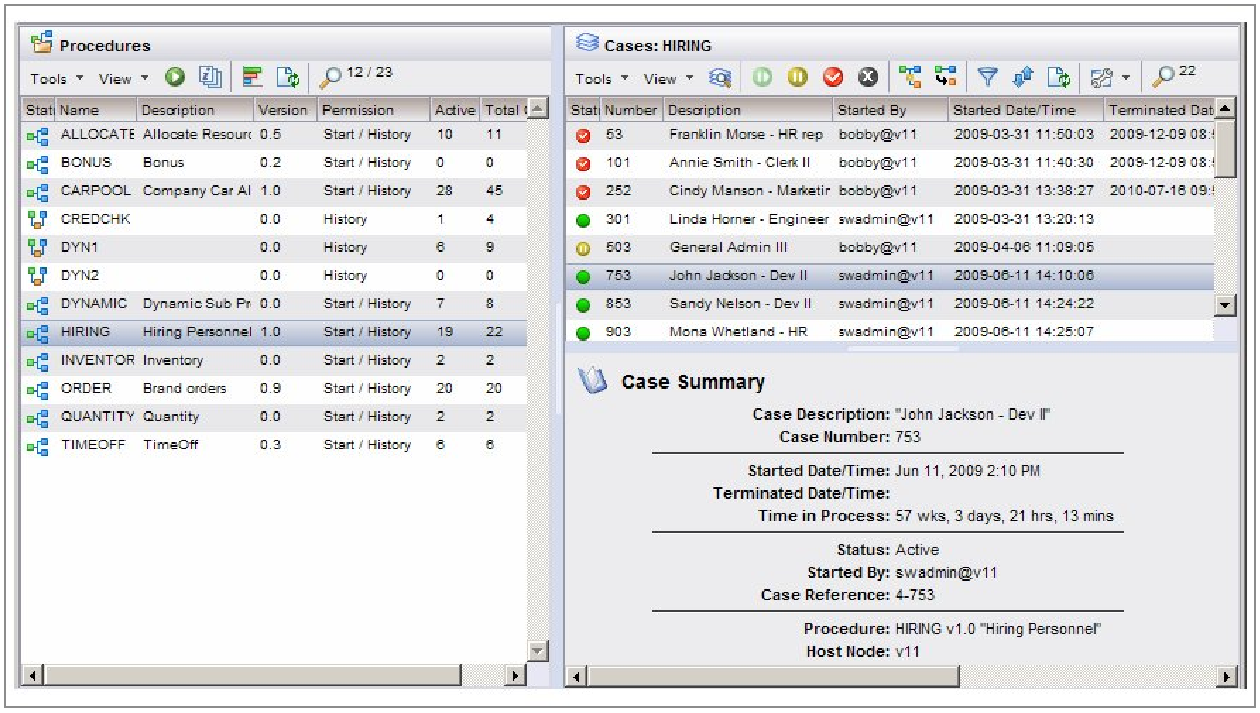
The **ProcsCases** component is a composite of the following two components:

- **StartHistoryProcs** (see Procedures)
- **Cases** (see Cases) (note that when using the **ProcsCases** component, the case list is always displayed with the preview pane; it appears the same as the **CasesPreview** component, but is actually the **Cases** component)

This composite component:

- displays the procedure list — by default, the procedure list contains all procedures for which the user has start and history access (it uses the **StartHistoryProcs** component)
- displays the case list when the user clicks on a procedure in the procedure list — by default, the case list subscribes to the “List Item Select (single click)” event on the **StartHistoryProcs** component, causing the case list to appear upon a single click of a procedure
- displays the case summary in the preview pane when a case is selected (single click) from the case list
- displays the case details in the preview pane when a case is opened (double click) from the case list

The **ProcsCases** component appears as follows:



Procedures

Tools View 12 / 23

Stat	Name	Description	Version	Permission	Active	Total (
	ALLOCATE	Allocate Resour	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

Cases: HIRING

Tools View 22

Stat	Number	Description	Started By	Started Date/Time	Terminated Dat
✓	53	Franklin Morse - HR rep	bobby@v11	2009-03-31 11:50:03	2009-12-09 08:
✓	101	Annie Smith - Clerk II	bobby@v11	2009-03-31 11:40:30	2009-12-09 08:
✓	252	Cindy Manson - Marketir	bobby@v11	2009-03-31 13:38:27	2010-07-16 09:
●	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	

Case Summary

Case Description: "John Jackson - Dev II"
Case Number: 753

Started Date/Time: Jun 11, 2009 2:10 PM
Terminated Date/Time:
Time in Process: 57 wks, 3 days, 21 hrs, 13 mins

Status: Active
Started By: swadmin@v11
Case Reference: 4-753

Procedure: HIRING v1.0 "Hiring Personnel"
Host Node: v11

Required Data

The **ProcsCases** component does not need data to render, although there must have been a previous login because it needs a valid user and server name.

ProcsCases Component Properties

None — composite components do not have their own properties. To set the properties for the individual components that are part of the composite, click on the appropriate component in the **Component Hierarchy**, then use the Properties Editor to set access for functions provided by that component.

For more information about the properties available for the components included with **ProcsCases**, see:

- Procs Component Properties
- Cases Component Properties

ProcsCases Component Events

None — composite components do not publish events themselves, although the individual components that comprise the composite publish events. Therefore, other components cannot subscribe to events of a composite, but they can subscribe to events of the underlying components.

For more information about the events published by the components included with **ProcsCases**, see:

- Procs Component Events [note that the events published by the **StartHistoryProcs** component are described in the **Procs** component section]
- Cases Component Events

WorkItemsPreview

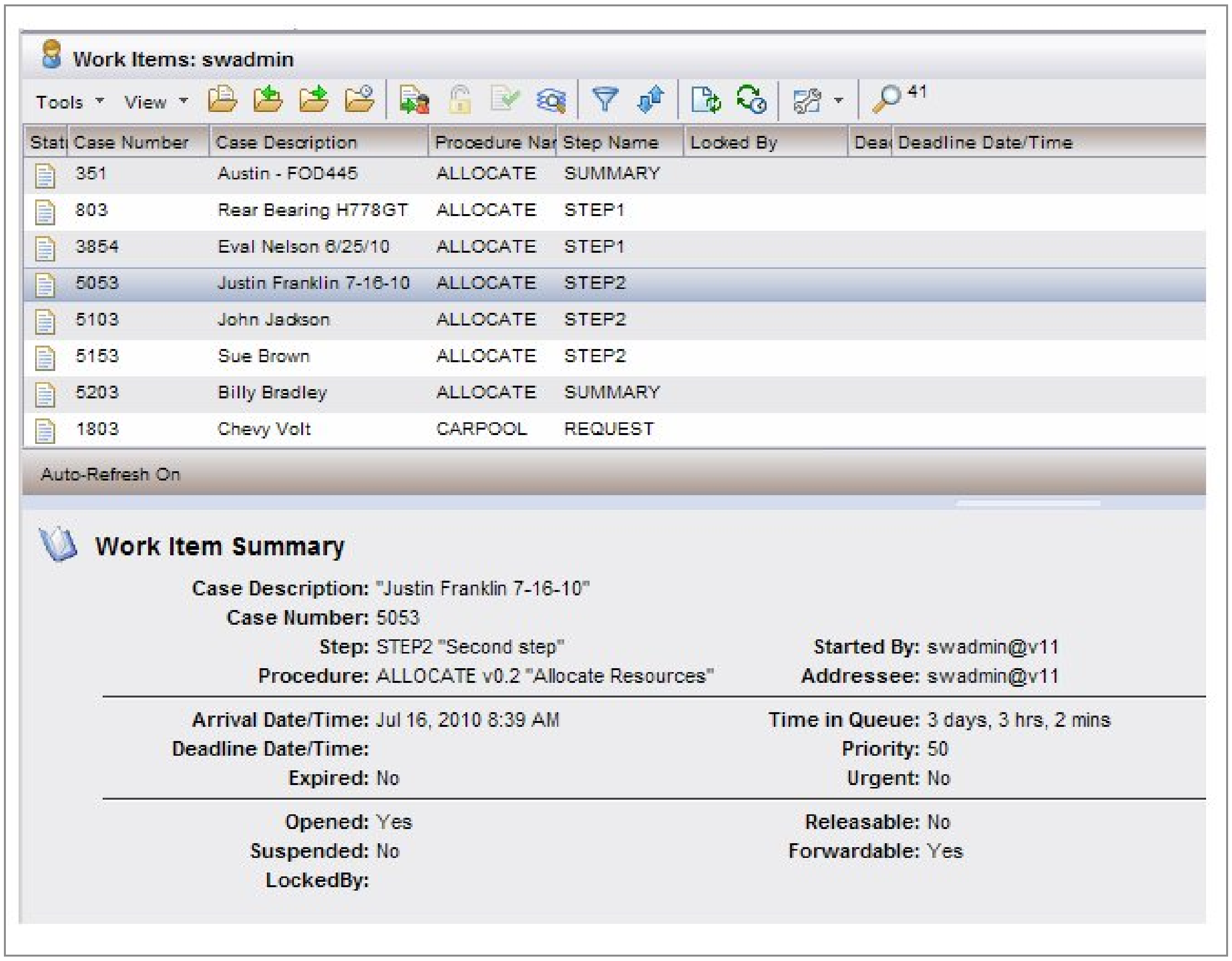
The **WorkItemsPreview** component is a composite of the work item list component (WorkItems) and the preview pane. (This is considered a composite, although technically it does not comprise two components that you can see in the **Component Hierarchy**.)

When a work item is selected (by single clicking) in the work item list displayed by this component, the work item summary is displayed in the preview pane. When a work item is opened (by double clicking, or selecting the **Open Selected Work Items** function), the work item form is displayed in the preview pane.

This differs from using just the WorkItems component, in that with the WorkItems component a work item summary is not displayed, and the work item form is always displayed in a floating window.

Note: Work item forms appear in the preview pane only if they were developed using TIBCO® General Interface. Non-General Interface forms always appear in a floating window.

The **WorkItemsPreview** component displays the work item list, followed by the preview pane:



Stat	Case Number	Case Description	Procedure Nar	Step Name	Locked By	Dea	Deadline Date/Time
	351	Austin - FOD445	ALLOCATE	SUMMARY			
	803	Rear Bearing H778GT	ALLOCATE	STEP1			
	3854	Eval Nelson 6/25/10	ALLOCATE	STEP1			
	5053	Justin Franklin 7-16-10	ALLOCATE	STEP2			
	5103	John Jackson	ALLOCATE	STEP2			
	5153	Sue Brown	ALLOCATE	STEP2			
	5203	Billy Bradley	ALLOCATE	SUMMARY			
	1803	Chevy Volt	CARPOOL	REQUEST			

Work Item Summary

Case Description: "Justin Franklin 7-16-10"
Case Number: 5053
Step: STEP2 "Second step"
Procedure: ALLOCATE v0.2 "Allocate Resources"
Started By: swadmin@v11
Addressee: swadmin@v11

Arrival Date/Time: Jul 16, 2010 8:39 AM
Deadline Date/Time:
Expired: No
Time in Queue: 3 days, 3 hrs, 2 mins
Priority: 50
Urgent: No

Opened: Yes
Suspended: No
LockedBy:
Releasable: No
Forwardable: Yes

Note: The work item list **Filter** dialog might be displayed instead of the work item list if the number of work items in the work queue exceeds a specified threshold. For more information, see WorkItems.

The work item list that is displayed using the **WorkItemsPreview** component also contains the **Preview** icon, as well as the Preview menu selections on the **View** menu to allow you to display the work item form in a floating window. (Note that the **WorkItems** component, which also displays the work item list, only opens the work item form in a floating window.)

Required Data

The **WorkItemsPreview** component requires a **work queue tag** to be able to be displayed.

Subscribe to: **UserWorkQs** component:

— List Item Select (single click) event

— List Item Execute (double click) event

WorkItemsPreview Component Properties

None — composite components do not have their own properties. To set the properties for the individual components that are part of the composite, click on the underlying component in the **Component Hierarchy**, then use the Properties Editor to set access for functions provided by that component.

For more information about the properties available for the components included with **WorkItemsPreview**, see:

- WorkItems Component Properties

WorkItemsPreview Component Events

None — composite components do not publish events themselves, although the individual components that comprise the composite publish events. Therefore, other components cannot subscribe to events of a composite, but they can subscribe to events of the underlying components. For more information, see *TIBCO iProcess Workspace (Browser) Components Concepts* guide.

For more information about the events published by the components included with **WorkItemsPreview**, see:

- WorkItems Component Events

WorkQsItems

The **WorkQsItems** component is a composite of the following two components:

— **UserWorkQs** (see UserWorkQs)

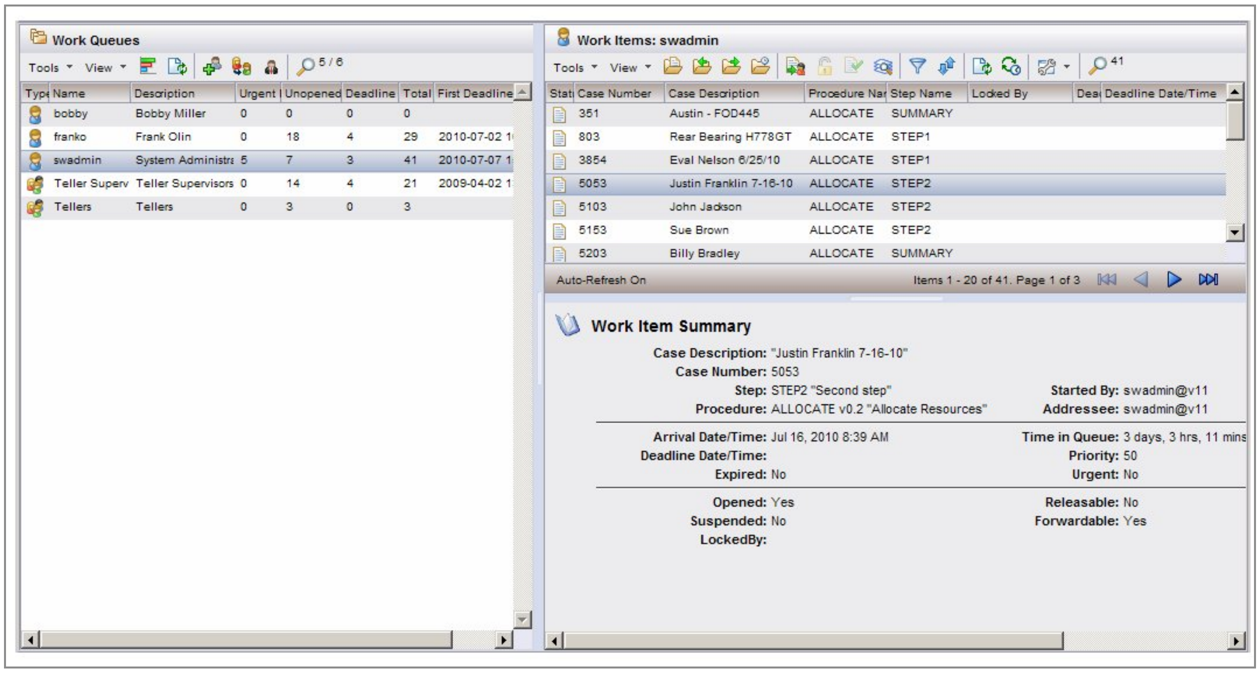
— **WorkItems** (see WorkItems) (note that when using the **WorkQsItems** component, the work item list is always displayed with the preview pane; it appears the same as the **WorkItemsPreview** component, but is actually the **WorkItems** component)

This composite component:

- displays the work queue list in a pane to the left of a vertical slider

- displays the work item list in the upper right part of the window — by default, the **WorkItems** component subscribes to the “List Item Select (single click)” event on the **UserWorkQs** component, causing the work item list to appear upon a single click of a work queue.
- displays the work item summary in the preview pane when a work item is selected from the work item list
- displays the work item form in the preview pane when a work item is opened from the work item list

The **WorkQsItems** component appears as follows:



Type	Name	Description	Urgent	Unopened	Deadline	Total	First Deadline
	bobby	Bobby Miller	0	0	0	0	
	franko	Frank Olin	0	18	4	29	2010-07-02 1
	swadmin	System Administr	5	7	3	41	2010-07-07 1
	Teller Superv	Teller Supervisors	0	14	4	21	2009-04-02 1
	Tellers	Tellers	0	3	0	3	

Stat	Case Number	Case Description	Procedure Nar	Step Name	Locked By	Dea	Deadline Date/Time
	351	Austin - FOD445	ALLOCATE	SUMMARY			
	803	Rear Bearing H778GT	ALLOCATE	STEP1			
	3854	Eval Nelson 6/25/10	ALLOCATE	STEP1			
	5053	Justin Franklin 7-16-10	ALLOCATE	STEP2			
	5103	John Jackson	ALLOCATE	STEP2			
	5153	Sue Brown	ALLOCATE	STEP2			
	5203	Billy Bradley	ALLOCATE	SUMMARY			

Auto-Refresh On   Items 1 - 20 of 41. Page 1 of 3

Work Item Summary

Case Description: "Justin Franklin 7-16-10"
Case Number: 5053
Step: STEP2 "Second step"   **Started By:** swadmin@v11
Procedure: ALLOCATE v0.2 "Allocate Resources"   **Addressee:** swadmin@v11

Arrival Date/Time: Jul 16, 2010 8:39 AM   **Time in Queue:** 3 days, 3 hrs, 11 mins
Deadline Date/Time: **Priority:** 50
Expired: No   **Urgent:** No

Opened: Yes   **Releasable:** No
Suspended: No   **Forwardable:** Yes
LockedBy:

Note: The work item list **Filter** dialog might be displayed instead of the work item list if the number of work items in the selected work queue exceeds a specified threshold. For more information, see WorkItems.

Required Data

The **WorkQsItems** component does not need data to render, although there must have been a previous login because it needs a valid user and server name.

WorkQsItems Component Properties

None — composite components do not have their own properties. To set the properties for the underlying components that are part of the composite, click on the appropriate component in the **Component Hierarchy**, then use the Properties Editor to set access for functions provided by that component.

For more information about the properties available for the components included with **WorkQsItems**, see:

- UserWorkQs Component Properties
- WorkItems Component Properties

WorkQsItems Component Events

None — composite components do not publish events themselves, although the individual components that comprise the composite publish events. Therefore, other components cannot subscribe to events of a composite, but they can subscribe to events of the underlying components.

For more information about the events published by the components included with **WorkQsItems**, see:

- UserWorkQs Component Events
- WorkItems Component Events

Lists

The List components are subdivided into the following categories. These correspond to the type of information displayed in the list:

- Cases
- Procedures
- Work Items
- Work Queues

The following subsections provide details of the list components.

Cases

The following component is available for displaying a case list:

- **Cases** - Displays a list of all cases for a specific procedure.

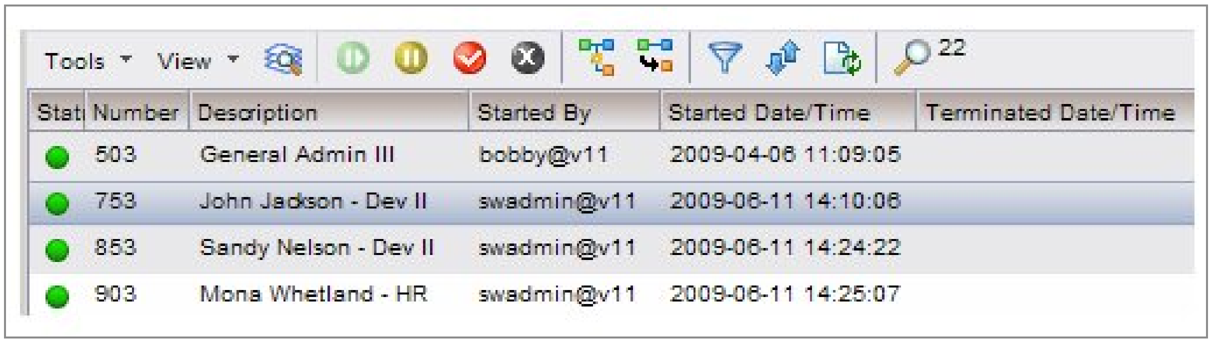
The following subsections provide details of the list/cases component.

Cases

When the **Cases** component is activated, one of two screens is displayed:

- The case list Filter dialog - This dialog is displayed if the number of cases that will appear in the case list exceeds a specified threshold number. For more information about the filter threshold and filtering the case list, see Case List Filtering.
- The case list - This is displayed if the number of cases in the list does not exceed a specified threshold number.

An example case list is illustrated below.



Stat	Number	Description	Started By	Started Date/Time	Terminated Date/Time
●	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	

The following describes the functions available from the case list:

- **Open Case(s)** - This *opens* the case, which causes the case details to be displayed in a floating window. (You can also display the case details using the **CaseDetails** component (see CaseDetail). You can also display the case list and open the case details in the preview pane using the **CasesPreview** component (see CasesPreview).
- **Activate Case(s)** - Reactivates the selected suspended cases.
- **Suspend Case(s)** - Suspends activity in the selected cases and all of their sub-cases. Suspended cases can be reactivated using the **Activate Case(s)** button.
- **Close Case(s)** - Stops the process flow for the selected cases.
- **Purge Case(s)** - Permanently deletes the selected cases from the system.

- **Trigger Event** - Starts the process flow from an Event step in the procedure.
- **Process Jump** - Used to withdraw currently outstanding steps from the case, and jump to new outstanding steps.
- **Filter** - Used to reduce the number of cases in the case list by filtering out the unneeded cases. For more information, see Case List Filtering.
- **Sort** - Used to list the cases in the case list in the desired order.

  If you had applied sort criteria the last time you viewed the case list, that same sort criteria will be applied when you open the list again — in other words, the most recently applied sort criteria will remain persistent until you either change it or remove it. If previously applied sort criteria is still active when the case list is displayed, a red check mark will be shown on the **Sort** icon.
- **Refresh Cases** - Requests that new data be retrieved from the server, refreshing the case list. (Note - Access to this function is not controlled by a property, nor does it fire an event when selected.)
- **Find** - Used to find the desired case(s) by entering search criteria. (Note - Access to this function is not controlled by a property, nor does it fire an event when selected.)
- **Select Columns** - Used to display the desired columns in the case list. (This function is only available from the case list **View** menu — there is no **Select Column** icon on the tool bar. Also, this function does not fire an event when selected.)

If the currently signed on user does not have access privileges to a particular function, the icon/menu selection for that function does not display. Access to each of the functions available on the case list is controlled by properties on the **Cases** component. Note that the **Cases** component also has properties available that control access to the functions available on the case detail tabs that are displayed when you open the case. For more information about setting these properties, see Cases Component Properties.

The **Cases** component also publishes events for many of the functions available from the case list. For more information about these events, see Cases Component Events.

For more information about using the functions listed above (e.g., activating a case, closing a case, etc.), see the *TIBCO iProcess Workspace (Browser) User Guide.*

Required Data

The **Cases** component requires a **procedure tag** to be able to be displayed.

Subscribe to: **HistoryProcs**, **NodeProcs**, **StartHistoryProcs**, or **StartProcs** component:

- — List Item Select (single click) event
- — List Item Execute (double click) event

Cases Component Properties

The **Cases** component contains the following properties, which are used to control access to each of the functions available from the case list.

Property	Description
Case	Enables/disables access to the case list. This property controls rendering of the case list. This would be used to control access from an external application, rather than from an event from another component (such as the procedure list).
Activate Case	Enables/disables the ability to activate a suspended case. This property controls access to the **Activate Case(s)** function on the case list, and on the case **Summary** tab that is displayed after the case is opened.
Close Case	Enables/disables the ability to close a case. This property controls access to the **Close Case(s)** function on the case list, and on the case **Summary** tab that is displayed after the case is opened.
Process Jump	Enables/disables the ability to jump to a new outstanding step in the case. This property controls access to the **Process Jump** function on the case list, and on the case **Summary** tab that is displayed after the case is opened.
Process Jump - View Fields	Enables/disables read-only access to the **Case Data** dialog available through the **Process Jump** dialog. If the **Process Jump** / **Update Fields** property (see below) is also enabled, it overrides this property, giving the user update access to case data. If both this and the **Process Jump** / **Update Fields** property are disabled, the **Data** button is not displayed on the **Process Jump** dialog box.
Process Jump - Update	Enables/disables update access to the **Case Data** dialog available through the **Process Jump** dialog.

Property	Description
Fields	This property overrides the **Process Jump** / **View Fields** property (see above) if it is enabled. If both this and the **Process Jump** / **View Fields** property are disabled, the **Data** button is not displayed on the **Process Jump** dialog box.
Process Jump - Select Outstanding Items Columns	Enables/disables the ability to specify which columns to display in the outstanding items list on the **Process Jump** dialog box. This property controls access to the **Select Columns** selection on the **View** menu for the outstanding items list on the **Process Jump** dialog box.
Suspend Case	Enables/disables the ability to suspend a case. This property controls access to the **Suspend Case(s)** function on the case list, and on the case **Summary** tab that is displayed after the case is opened.
Event Trigger	Enables/disables the ability to trigger an event in a case. This property controls access to the **Trigger Event** function on the case list, and on the case **Summary** tab that is displayed after the case is opened.
Event Trigger - View Fields	Enables/disables read-only access to the **Case Data** dialog available through the **Events** dialog box. If the **Trigger Event** / **Update Fields** property (see below) is also enabled, it overrides this property, giving the user update access to case data. If both this and the **Trigger Event** / **Update Fields** property are disabled, the **Data** button is not displayed on the **Events** dialog box.
Event Trigger - Update Fields	Enables/disables update access to the **Case Data** dialog available through the **Events** dialog. This property overrides the **Trigger Event** / **View Fields** property (see above) if it is enabled. If both this and the **Trigger Event** / **View Fields** property are disabled, the **Data** button is not displayed on the **Events** dialog box.

Property	Description
Event Trigger - Resurrect Case	Enables/disables the ability to resurrect (reactivate) a closed case. This property controls access to the **Trigger Event** function on the case list when a closed case is selected, and on the case **Summary** tab that is displayed when a closed case is opened.
Event Trigger - Recalculate Deadlines	Enables/disables the ability to recalculate deadlines in the case. This property controls access to the **Recalculate Deadlines** radio buttons on the **Events** dialog box.
Purge Case	Enables/disables the ability to purge (permanently delete) cases. This property controls access to the **Purge Case(s)** function on the case list, and on the case **Summary** tab that is displayed after the case is opened.
Open Case	Enables/disables the ability to open cases. This property controls access to the **Open Case(s)** function on the case list.
Open Case - Case Summary	Enables/disables the ability to access the **Summary** tab after opening a case.
Open Case - Case History	Enables/disables the ability to access the **History** tab after opening a case.
Open Case - Case History - Add History Entry	Enables/disables the ability to manually add an entry to the case history. This property controls access to the **Add Entry** function on the **History** tab.
Open Case - Case History - Case	Enables/disables the ability to perform a case prediction function. This property controls access to the **Predict Case** function on the **History** tab.

Property	Description
Predict	
Open Case - Case History - Graphical Case History	Enables/disables the ability to view the case history in a graphical form. This property controls access to the **Graphical History** function on the **History** tab.
Open Case - Outstanding Items	Enables/disables the ability to access the **Outstanding** tab after opening a case.
Open Case - Outstanding Items - Select Columns	Enables/disables the ability to specify which columns to display in the outstanding items list on the **Outstanding** tab. This property controls access to the **Select Columns** selection on the **Outstanding** tab **View** menu.
Open Case - View Fields	Enables/disables read-only access to the **Data** tab. If the **Open Case / Update Fields** property (see below) is also enabled, it overrides this element, giving the user update access to case data. If both this and the **Open Case / Update Fields** property are disabled, the case **Data** tab is hidden.
Open Case - Update Fields	Enables/disables update access to the **Data** tab. This property overrides the **Open Case** / **View Fields** property (see above) if it is enabled. If both this and the **Open Case** / **View Fields** property are disabled, the case **Data** tab is hidden.
Select Columns	Enables/disables the ability to specify which columns to display in the case list.

Property	Description
	This property controls access to the **Select Columns** selection on the case list **View** menu.
Filter	Enables/disables the ability to filter the cases in the case list. This property controls access to the **Filter** function on the case list. Note that this property setting does not prevent the filter dialog from being displayed if the number of cases in the list exceeds the specified threshold — for more information, see Case List Filtering.
Sort	Enables/disables the ability to sort the cases in the case list. This property controls access to the **Sort** function on the case list.

Cases Component Events

The **Cases** component publishes the following events, which fire when the action described by the event occurs.

Event	Description
List Item Select (single click)	Indicates the user has “selected” a case in the case list. Fires when the user single-clicks a case in the case list, or when the user moves the highlight bar on the case list using the keyboard arrow keys.
List Item Execute (double click)	Indicates the user has “executed” a case in the case list. Fires when the user double-clicks a case in the case list, or single-clicks a case, then presses the **Enter** key.
Process Jump	Indicates the user is performing a process jump function. Fires when the user clicks the **Process Jump** button on the case list toolbar, or chooses the **Process Jump** menu selection on either the case list or the case **Summary** tab.
Trigger Event	Indicates the user is performing a trigger event function.

Event	Description
	Fires when the user clicks the **Trigger Event** button on the case list toolbar, or chooses the **Trigger Event** menu selection on either the case list or the case **Summary** tab.
Purge Case	Indicates the user is purging (permanently deleting) one or more cases. Fires when the user clicks the **Purge Case(s)** button on the case list toolbar, or chooses the **Purge Case(s)** menu selection on either the case list or the case **Summary** tab.
Close Case	Indicates the user is closing one or more cases. Fires when the user clicks the **Close Case(s)** button on the case list toolbar, or chooses the **Purge Case(s)** menu selection on either the case list or the case **Summary** tab.
Suspend Case	Indicates the user is suspending one or more cases. Fires when the user clicks the **Suspend Case(s)** button on the case list toolbar, or chooses the **Suspend Case(s)** menu selection on either the case list or the case **Summary** tab.
Activate Case	Indicates the user is activating one or more suspended cases. Fires when the user clicks the **Activate Case(s)** button on the case list toolbar, or chooses the **Activate Case(s)** menu selection on either the case list or the case **Summary** tab.
Open Case	Indicates the user is initiating the **Open Case(s)** function. Fires when the user clicks the **Open Case(s)** button on the case list toolbar, or chooses the **Open Case(s)** menu selection on the case list.
View Graphical Case History	Indicates the user is viewing the case’s history in a graphical format. Fires when the user clicks the **Graphical History** button on the **History** tab toolbar, or chooses the **Graphical History** selection on the **History** tab **View** menu.
Case	Indicates the user is performing a case prediction operation.

Event	Description
Prediction	Fires when the user clicks the **Predict Case** button on the **History** tab toolbar, or chooses the **Predict Case** selection on the **History** tab **Tools** menu.
Add History	Indicates the user is manually adding an entry to the case history. Fires when the user clicks the **Add Entry** button on the **History** tab toolbar, or chooses the **Add Entry** selection on the **History** tab **Tools** menu.
Select Case History (single click)	Indicates the user has selected an entry in the case history. Fires when the user single-clicks a case history entry on the **History** tab.
Execute Case History (double click)	Indicates the user has executed an entry in the case history. Fires when the user double-clicks a case history entry on the **History** tab. This event also fires when the user presses **Enter** when a case history entry is already selected (highlighted).
Select Case Outstanding (single click)	Indicates the user has selected an outstanding work item. Fires when the user single-clicks an outstanding work item on the **Outstanding** tab.
Execute Case Outstanding (double click)	Indicates the user has executed an outstanding work item. Fires when the user double-clicks an outstanding work item on the **Outstanding** tab. This event also fires when the user presses **Enter** when an outstanding work item is already selected (highlighted).
Open Case Outstanding Work Item(s)	Indicates the user has opened an outstanding work item. Fires when the user clicks the **Open Selected Work Item(s)** button on the **Outstanding** tab toolbar, or chooses the **Open Selected Work Item(s)** selection on the **Outstanding** tab **Tools** menu.
Case Data Update	Indicates the user has updated case data. Fires when the user clicks on the **Apply** button on the **Data** tab.

Case List Filtering

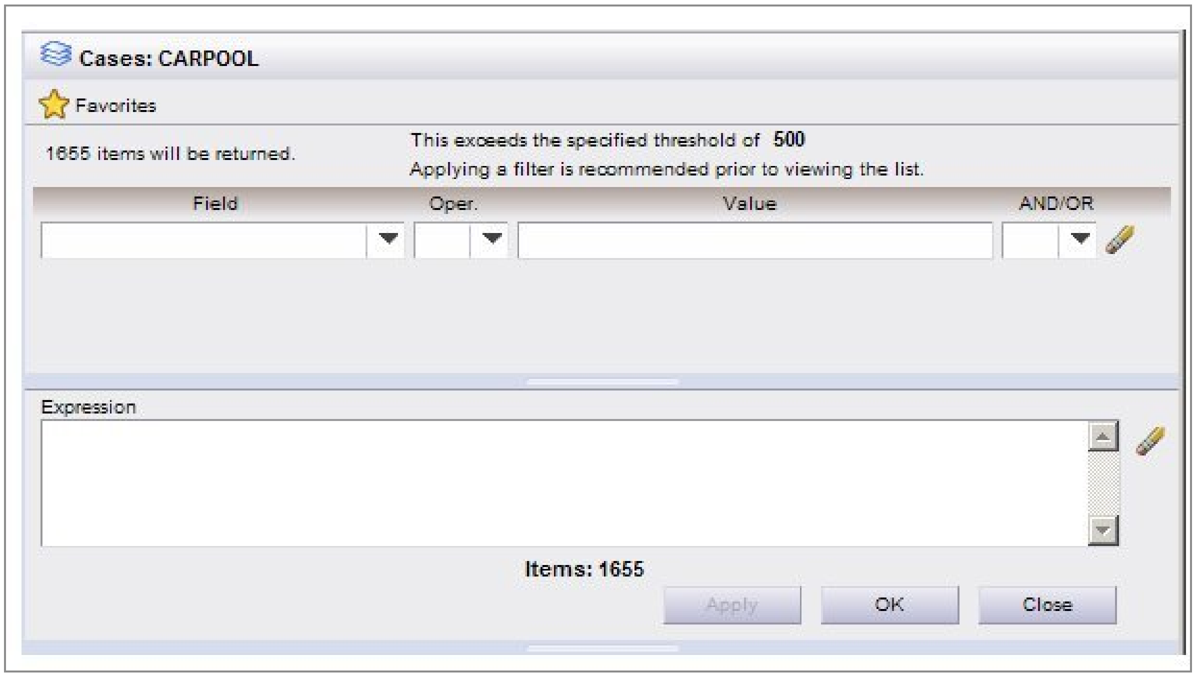
Filtering a list of cases involves entering filter criteria so that only some of the cases are shown in the list, rather than all of them. Filtering a list of cases allows you to display only the cases you are interested in. For example, you may only be interested in cases that were started earlier than a certain date. You can filter the list so only those cases are displayed.

The filtering function allows you to build a “filter expression” that is applied to all cases for the procedure that was selected. If the case satisfies the filter expression (e.g., the start date is earlier than a specified date), it is shown in the case list; if it does not satisfy the filter expression (e.g., the start date is later than a specified date), it is not shown in the list (and is not downloaded from the iProcess Engine).

You can filter the case list at any time by clicking the **Filter** icon, or by selecting **Filter** from the **View** menu on the case list. This displays the filter dialog box, on which you can specify a filter expression.

The filter dialog may also be automatically displayed when you attempt to display a list of cases by selecting a procedure from the procedure list. This occurs if the number of cases of the procedure exceeds a threshold that has been specified in the TIBCO iProcess Workspace (Browser) component application configuration file, `config.xml`. This threshold defaults to 500. (For more information about setting the case list filter threshold, see the *TIBCO iProcess Workspace (Browser) Configuration and Customization* guide).

The following image is an example of filter dialog box:


Cases: CARPOOL

1655 items will be returned.

This exceeds the specified threshold of 500

Applying a filter is recommended prior to viewing the list.

Note:
If you had applied a filter expression the last time you viewed the case list, that same filter expression will be applied when you open the list again — in other words, the most recently applied filter expression will remain persistent until you either change it or remove it (to remove an expression, click the “eraser” icon to the right of the **Expression** section).

If a filter expression is still active, the **Filter** icon is shown with a green check mark, and if the filter dialog is displayed, the previously entered filter expression is shown in the **Expression** section.

For more information about building a filter expression and how to apply the expression, see the *TIBCO iProcess Workspace (Browser) User Guide.*

Procedures

The following components are available for displaying a procedure list:

— **HistoryProcs** - lists only the procedures for which the user is permitted to view cases

— **StartProcs** - lists only the procedures for which the user is permitted to start new cases

— **StartHistoryProcs** - lists the procedures for which the user is permitted to view cases and/or start cases

— **NodeProcs** - list of all procedures defined on a server node

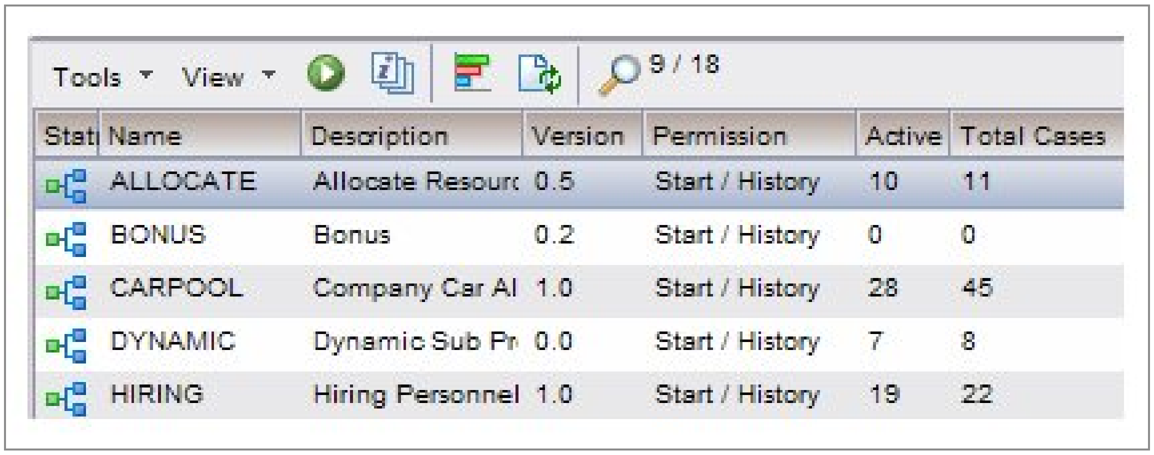
All four of the components listed above cause the procedure list to be displayed. The difference is each component displays procedures in the list based on different user permissions — one displays only the procedures for which the user is authorized to view case lists, another displays only the procedures for which the user can start cases, etc.

Note: “History” in this context does not mean you have “case history” permission; rather it means you have “procedure history” permission — you can view the cases that have been started for that procedure. In other words, you are authorized to view the case list for the procedure.

Use the appropriate component to display the procedure list, based on which procedures you want to appear in the list.

Note: Also, see the Composites components, which can be used to display the procedure list in conjunction with other components.

An example procedure list is illustrated below.



Stat	Name	Description	Version	Permission	Active	Total Cases
	ALLOCATE	Allocate Resour	0.5	Start / History	10	11
	BONUS	Bonus	0.2	Start / History	0	0
	CARPOOL	Company Car Al	1.0	Start / History	28	45
	DYNAMIC	Dynamic Sub Pr	0.0	Start / History	7	8
	HIRING	Hiring Personnel	1.0	Start / History	19	22

The following describes the functions available from the procedure list:

- **Start Case** - Starts a new case of the selected procedure.
- **Procedure Versions** - Displays version history for the selected procedure.
- **Procedure Loading Chart** - Displays a graphical summary of the cases for all procedures on the system.
- **Refresh** - Requests that new data be retrieved from the server, refreshing the procedure list. (Note - Access to this function is not controlled by a property, nor does it fire an event when selected.)
- **Find** - Used to find the desired procedure(s) by entering search criteria. (Note - Access to this function is not controlled by a property, nor does it fire an event when selected.)
- **Select Columns** - Used to display the desired columns in the procedure list. (This is available from the procedure list **View** menu. Also, this function does not fire an event when selected.)

If the currently signed-on user does not have access privileges to a particular function, the icon/menu selection for that function does not display. Access to each of the functions available on the procedure list is controlled by properties on the **HistoryProcs**, **StartProcs**,

StartHistoryProcs, and **NodeProcs** components. For more information about setting these properties, see Procs Component Properties.

The procedure components also publish events for some of the functions on the procedure list. For more information about these events, see Procs Component Events.

For more information about using the functions listed above (e.g., starting a case, viewing procedure versions, etc.), see the *TIBCO iProcess Workspace (Browser) User Guide.*

Required Data

The **Procs** components do not need data to render, although there must have been a previous login because they need a valid user and server name.

Procs Component Properties

The **HistoryProcs**, **StartProcs**, **StartHistoryProcs**, and **NodeProcs** components contain the following properties, which are used to control access to each of the functions available from the procedure list.

Property	Description
Procedure View	Enables/disables access to the procedure list. This property controls rendering of the procedure list. This would be used to control access from an external application, rather than from an event from another component (such as the Login component).
Procedure Versions	Enables/disables access to the Procedure Versions function, which allows the user to view version history for the selected procedure. This property controls access to the **Procedure Versions** button on the toolbar, as well as the **Versions** selection on the procedure list **Tools** menu.
Case Start	Enables/disables the ability for the user to start cases of procedures. This property controls access to the **Start Case** button on the toolbar, as well as the **Start New Case** selection on the procedure list **Tools** menu.
Loading Chart	Enables/disables the ability for the user to display the procedure loading chart.

Property	Description
	This property controls access to the **Procedure Loading Chart** button on the toolbar, as well as the **Procedure Loading Chart** selection on the procedure list **Tools** menu.
Select Columns	Enables/disables the ability for the user to specify which columns to display in the procedure list. This property controls access to the **Select Columns** selection on the procedure list **View** menu.

Procs Component Events

The **HistoryProcs**, **StartProcs, StartHistoryProcs**, and **NodeProcs** components contain the following events, which fire when the action described by the event occurs:

Event	Description
List Item Select (single click)	Indicates the user has selected a procedure in the procedure list. Fires when the user single-clicks a procedure in the procedure list, or when the user moves the highlight bar on the procedure list using the keyboard arrow keys.
List Item Execute (double click)	Indicates the user has executed a procedure in the procedure list. Fires when the user double-clicks a procedure in the procedure list. This event also fires when the user presses **Enter** when a procedure is already selected (highlighted).
Start Case	Indicates the user has started a case of a procedure. Fires when the user clicks on the **Start Case** toolbar button, or chooses the **Start New Case** menu selection on the procedure list.
Procedure Versions	Indicates the user has requested to view the procedure version history. Fires when the user clicks on the **Procedure Versions** toolbar button, or chooses the **Versions** menu selection on the procedure list.

Event	Description
Loading Chart	Indicates the user has requested to view the procedure loading chart. Fires when the user clicks on the **Procedure Loading Chart** toolbar button, or chooses the **Procedure Loading Chart** menu selection on the procedure list.

Work Items

The following component is available for displaying work item information in lists:

— WorkItems - list of all work items for a specific work queue

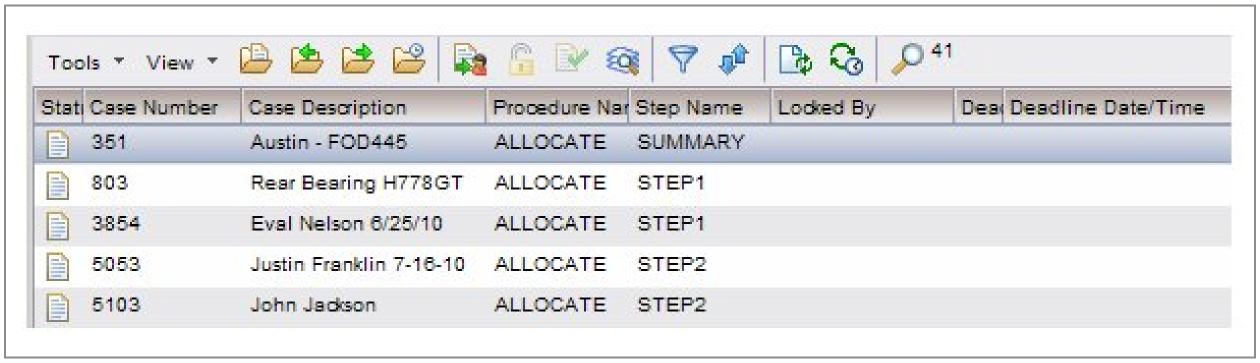
The following subsection provides details of this component.

WorkItems

When the **WorkItems** component is activated, one of two screens is displayed:

- The work item list Filter dialog - This dialog is displayed if the number of work items that appear in the work items list exceeds a specified threshold number. For more information about the filter threshold and filtering the case list, see Work Item List Filtering.
- **The work item list** - This is displayed if the number of work items in the list does not exceed the specified threshold number.

An example work item list is illustrated below.



Stat	Case Number	Case Description	Procedure Nar	Step Name	Locked By	Dea	Deadline Date/Time
	351	Austin - FOD445	ALLOCATE	SUMMARY			
	803	Rear Bearing H778GT	ALLOCATE	STEP1			
	3854	Eval Nelson 6/25/10	ALLOCATE	STEP1			
	5053	Justin Franklin 7-16-10	ALLOCATE	STEP2			
	5103	John Jackson	ALLOCATE	STEP2			

The following describes the functions available from the work item list:

- **Open Selected Work Item(s)** - Opens (and locks) the selected work items.

- **Open First Work Item** - Causes the first available work item in the list to be opened, where “available” means a work item that is not locked nor suspended.
- **Open Next Work Item** - Causes the next available work item in the list to be opened, starting from the currently selected (highlighted) work item, where “available” means a work item that is not locked nor suspended. If there is no work item currently selected, the first available work item in the list is opened.
- **Auto-Repeat Open Work Item** - Causes the next available work item to be automatically opened after you’ve released a work item, where “available” means a work item that is not locked nor suspended. This allows you to process work items in succession without manually opening each one.

  The auto-repeat feature works as a toggle — clicking the **Auto-Repeat Open Work Item** button causes it to remain highlighted, and selecting **Auto-Repeat Open Work Item** from the **Tools** menu causes a check mark to appear next to the selection, indicating the feature is active.
- **Forward Work Item(s)** - Allows you to manually forward one or more work items to another work queue.
- **Unlock Work Item(s)** - Unlocks the selected work items.

Note: Work items are automatically unlocked when you keep or release them; normally, you do not need to explicitly unlock work items. This function is for those rare occasions when a work item was left open for some reason (e.g., a system crash).

- **Release Work Item(s)** - Causes any information that has been entered into the work item form (if applicable) to be written to the database (considered “case data”). The work item is removed from the work queue, then the case advances to the next step in the procedure, possibly resulting in another work item appearing in someone’s work queue.
- **Open Case** - Opens the case details (Summary, History, Outstanding, and Data tabs) for the case associated with the currently selected work item. For more information about case details, see CaseDetail.
- **Filter** - Used to reduce the number of work items in the work item list by filtering out the unneeded work items. For more information, see Work Item List Filtering.
- **Sort** - Used to list the work items in the work item list in the desired order.

  If you had applied sort criteria the last time you viewed the work item list, that same sort criteria will be applied when you open the list again — in other words, the most recently applied sort criteria will remain persistent until you either change it or remove it. If previously applied sort criteria is still active when the work item list is displayed, a red check mark will be shown on the **Sort** icon.

- **Refresh Work Items** - Requests that new data be retrieved from the server, refreshing the work item list. (Note - Access to this function is not controlled by a property, nor does it fire an event when selected.)
- **Auto-Refresh** - Displays the **Auto-Refresh** dialog, which allows you to enable or disable the auto-refresh feature. When enabled, the auto-refresh feature causes the work item list to be automatically refreshed at specified intervals.
- **Find** - Used to find the desired work item(s) by entering search criteria. (Note: Access to this function is not controlled by a property, nor does it fire an event when selected.)
- **Page Size** - Allows you to change the number of work items in a *page*.

  When a list of work items is displayed on your screen, only a *page* of work items is displayed at one time. This is done to speed up the display, especially if there are a very large number of work items in the queue.

  (This function is only available from the work item list **View** menu — there is no **Page Size** icon on the tool bar. Also, access to this function is not controlled by a property, nor does it fire an event when selected.
- **Select Columns** - Used to display the desired columns in the work item list.

  (This function is only available from the work item list **View** menu — there is no **Select Columns** icon on the tool bar. Also, an event does not fire when this function is selected.)

If the currently signed on user does not have access privileges to a particular function, the icon/menu selection for that function does not display. Access to each of the functions available on the work item list is controlled by properties on the **WorkItems** component. For more information about setting these properties, see WorkItems Component Properties.

The **WorkItems** component also publishes events for many of the functions available from the work item list. For more information about these events, see WorkItems Component Events.

For more information about using the functions listed above (e.g., forwarding work items, releasing work items, etc.), see the *TIBCO iProcess Workspace (Browser) User Guide.*

Required Data

The **WorkItems** component requires a **work queue tag** to be able to be displayed.

Subscribe to: **UserWorkQs** component:

- List Item Select (single click) event
- List Item Execute (double click) event

WorkItems Component Properties

The **WorkItems** component contains the following properties, which are used to control access to each of the functions available from the work item list.

Property	Description
WorkItem	Enables/disables access to the work item list. This property controls rendering of the work item list. This would be used to control access from an external application, rather than from an event from another component (such as the work queue list).
Auto-Refresh	Enables/disables the ability to automatically refresh the work item list. This property controls access to the **Refresh Work Items** function on the work item list.
Open	Enables/disables the ability to open (lock) a work item. This property controls access to the **Open Selected Work Item(s)** function on the work item list.
Open First Item	Enables/disables the ability to open (lock) the first work item in the list. This property controls access to the **Open First Work Item** function on the work item list.
Open Next Item	Enables/disables the ability to open (lock) the next work item in the list. This property controls access to the **Open Next Work Item** function on the work item list.

Property	Description
Open Auto-Repeat	Enables/disables the ability to automatically open (lock) a work item. This property controls access to the **Auto-Repeat Open Work Items** function on the work item list. Note that if access to both OpenFirst Item and OpenNext Item (see above) are prohibited, the **Auto-Repeat Open Work Items** tool is automatically disabled, as it requires OpenFirst Item and OpenNext Item.
Forward	Enables/disables the ability to forward a work item to another work queue. This property controls access to the **Forward Work Item(s)** function on the work item list.
Forward - Forward to any queue	Enables/disables the ability to forward work items to any work queue. This is applicable only if the user has access to the **Forward Work Item (s)** function (provided by the **Forward** property — see above). This causes the list of work queues on the **Forward Selected Work Items** dialog to contain all work queues on the system. If disabled, the list of work queues on the **Forward Selected Work Items** dialog contain only the work queues of which the user is a member.
Release	Enables/disables the ability to release a work item via the **Release Work Item(s)** function. This property controls access to the **Release Work Item(s)** button on the work item list, as well as the **Release Work Item(s)** selection on the work item list **Tools** menu. **Note:** If this property is disabled, it does not prevent the user from releasing a work item via a work item form.
Unlock	Enables/disables the ability to unlock a work item locked by another user. This property controls access to the **Unlock Work Item(s)** function on the work item list.

Property	Description
Select Columns	Enables/disables the ability to specify which columns to display in the work item list. This property controls access to the **Select Columns** selection on the work item list **View** menu.
Open Case	Enables/disables the ability to open a case from the work item list. This property controls access to the **Open Case** function on the work item list.
Open Case - Case Summary	Enables/disables the ability to access the **Summary** tab after opening a case from the work item list.
Open Case - Case Summary - Activate Case	Enables/disables the ability to activate a suspended case. This property controls access to the **Activate Case** function on the **Summary** tab when a case is opened from the work item list.
Open Case - Case Summary - Close Case	Enables/disables the ability to close a case. This property controls access to the **Close Case** function on the **Summary** tab when a case is opened from the work item list.
Open Case - Case Summary - Process Jump	Enables/disables the ability to jump to a new outstanding step in the case. This property controls access to the **Process Jump** function on the **Summary** tab when a case is opened from the work item list.
Open Case - Case Summary - Process Jump - View Fields	Enables/disables read-only access to the **Case Data** dialog available through the **Process Jump** dialog box. If the **Process Jump** / **Update Fields** property (see below) is also enabled, it overrides this property, giving the user update access to case data.

Property	Description
	If both this and the **Process Jump** / **Update Fields** property are disabled, the **Data** button is not displayed on the **Process Jump** dialog box.
Open Case - Case Summary - Process Jump - Update Fields	Enables/disables update access to the **Case Data** dialog available through the **Process Jump** dialog box. This property overrides the **Process Jump** / **View Fields** property (see above) if it is enabled. If both this and the **Process Jump** / **View Fields** property are disabled, the **Data** button is not displayed on the **Process Jump** dialog box.
Open Case - Case Summary - Process Jump - Select Outstanding Items Columns	Enables/disables the ability to specify which columns to display in the outstanding items list on the **Process Jump** dialog box. This property controls access to the **Select Columns** selection on the **View** menu for the outstanding items list on the **Process Jump** dialog box.
Open Case - Case Summary - Suspend Case	Enables/disables the ability to suspend a case. This property controls access to the **Suspend Case** function on the **Summary** tab that is displayed after the case is opened from the work item list.
Open Case - Case Summary - Trigger Event	Enables/disables the ability to trigger an event in a case. This property controls access to the **Trigger Event** function on the **Summary** tab that is displayed after the case is opened from the work item list.
Open Case - Case Summary - Trigger	Enables/disables read-only access to the **Case Data** dialog available through the **Events** dialog box. If the **Trigger Event** / **Update Fields** property (see below) is also enabled,

Property	Description
View Fields	it overrides this property, giving the user update access to case data. If both this and the **Trigger Event** / **Update Fields** property are disabled, the **Data** button is not displayed on the **Events** dialog box.
Open Case - Case Summary - Trigger Event - Update Fields	Enables/disables update access to the **Case Data** dialog available through the **Events** dialog. This property overrides the **Trigger Event** / **View Fields** property (see above) if it is enabled. If both this and the **Trigger Event** / **View Fields** property are disabled, the **Data** button is not displayed on the **Events** dialog box.
Open Case - Case Summary - Trigger Event - Resurrect Case	Enables/disables the ability to resurrect (reactivate) a closed case. This property controls access to the **Trigger Event** function on the **Summary** tab that is displayed when a closed case is opened from the work item list.
Open Case - Case Summary - Trigger Event - Recalculate Deadlines	Enables/disables the ability to recalculate deadlines in the case. This property controls access to the **Recalculate Deadlines** radio buttons on the **Events** dialog when the case is opened from the work item list.
Open Case - Case Summary - Purge Case	Enables/disables the ability to purge (permanently delete) cases. This property controls access to the **Purge Case** function on the **Summary** tab that is displayed after the case is opened from the work item list.
Open Case - Case History	Enables/disables the ability to access the **History** tab after opening a case from the work item list.

Property	Description
Open Case - Case History - Add History Entry	Enables/disables the ability to manually add an entry to the case history. This property controls access to the **Add Entry** function on the **History** tab after opening a case from the work item list.
Open Case - Case History - Case Predict	Enables/disables the ability to perform a case prediction function. This property controls access to the **Predict Case** function on the **History** tab after opening a case from the work item list.
Open Case - Case History - Graphical Case History	Enables/disables the ability to view the case history in a graphical form. This property controls access to the **Graphical History** function on the **History** tab after opening a case from the work item list.
Open Case - Outstanding Items	Enables/disables the ability to access the **Outstanding** tab after opening a case after opening a case from the work item list.
Open Case - Outstanding Items - Select Columns	Enables/disables the ability to specify which columns to display in the outstanding items list on the **Outstanding** tab. This property controls access to the **Select Columns** selection on the **Outstanding** tab **View** menu after opening a case from the work item list.
Filter	Enables/disables the ability to filter the work items in the work item list. This property controls access to the **Filter** function on the work item list. **Note:** This property setting does not prevent the filter dialog from being displayed if the number of items in the list exceeds the specified threshold — for more information, see Work Item List Filtering.
Sort	Enables/disables the ability to sort the work items in the work item list. This property controls access to the **Sort** function on the work item list.

WorkItems Component Events

The **WorkItems** component publishes the following events, which fire when the action described by the event occurs.

Event	Description
List Item Select (single click)	Indicates the user has selected a work item in the work item list. Fires when the user single-clicks a work item in the work item list, or when the user moves the highlight bar on the work item list using the keyboard arrow keys.
List Item Execute (double click)	Indicates the user has executed a work item in the work item list. Fires when the user double-clicks a work item in the work item list. This event also fires when the user presses **Enter** when a work item is already selected (highlighted).
Open Work Item(s)	Indicates the user has opened a work item. Fires when the user clicks the **Open Selected Work Item(s)** button on the work item list toolbar, or chooses the **Open Selected Work Item(s)** menu selection on the work item list.
Forward Work Item(s)	Indicates the user is forwarding a work item to another work queue. Fires when the user clicks the **Forward Work Item(s)** button on the work item list toolbar, or chooses the **Forward Work Item(s)** menu selection on the work item list.
Unlock Work Item(s)	Indicates the user is unlocking a work item that was locked by another user. Fires when the user clicks the **Unlock Work Item(s)** button on the work item list toolbar, or chooses the **Unlock Work Item(s)** menu selection on the work item list.
Release Work Item(s)	Indicates the user is releasing a work item via the work item list. Fires when the user clicks the **Release Work Item(s)** button on the work item list toolbar, or chooses the **Release Work Item(s)** menu selection on the work item list.

Event	Description
Open Case	Indicates the user is opening a case via the work item list. Fires when the user clicks the **Open Case** button on the work item list toolbar, or chooses the **Open Case** menu selection on the work item list.
Process Jump	Indicates the user is performing a process jump function. Fires when the user opens the case via the work item list, then clicks the **Process Jump** button, or chooses the **Process Jump** menu selection, on the case **Summary** tab.
Trigger Event	Indicates the user is performing a trigger event function. Fires when the user opens the case via the work item list, then clicks the **Trigger Event** button, or chooses the **Trigger Event** menu selection, on the case **Summary** tab.
Purge Case	Indicates the user is purging (permanently deleting) one or more cases. Fires when the user opens the case via the work item list, then clicks the **Purge Case** button, or chooses the **Purge Case** menu selection, on the case **Summary** tab.
Close Case	Indicates the user is closing one or more cases. Fires when the user opens the case via the work item list, then clicks the **Close Case** button, or chooses the **Purge Case** menu selection, on the case **Summary** tab.
Suspend Case	Indicates the user is suspending one or more cases. Fires when the user opens the case via the work item list, then clicks the **Suspend Case** button, or chooses the **Suspend Case** menu selection, on the case **Summary** tab.
Activate Case	Indicates the user is activating one or more suspended cases. Fires when the user opens the case via the work item list, then clicks the **Activate Case**, or chooses the **Activate Case** menu selection, on the case **Summary** tab.

Event	Description
View Graphical Case History	Indicates the user is viewing the case's history in a graphical format. Fires when the user opens the case via the work item list, then clicks the **Graphical History** button on the **History** tab toolbar, or chooses the **Graphical History** selection on the **History** tab **View** menu.
Case Prediction	Indicates the user is performing a case prediction operation. Fires when the user opens the case via the work item list, then clicks the **Predict Case** button on the **History** tab toolbar, or chooses the **Predict Case** selection on the **History** tab **Tools** menu.
Add History	Indicates the user is manually adding an entry to the case history. Fires when the user opens the case via the work item list, then clicks the **Add Entry** button on the **History** tab toolbar, or chooses the **Add Entry** selection on the **History** tab **Tools** menu.
Select Case History (single click)	Indicates the user has selected an entry in the case history. Fires when the user opens the case via the work item list, then single-clicks a case history entry on the **History** tab.
Execute Case History (double click)	Indicates the user has executed an entry in the case history. Fires when the user opens the case via the work item list, then double-clicks a case history entry on the **History** tab. This event also fires when the user presses **Enter** when a case history entry is already selected (highlighted).
Select Case Outstanding (single click)	Indicates the user has selected an outstanding work item. Fires when the user opens the case via the work item list, then single-clicks an outstanding work item on the **Outstanding** tab.
Execute Case Outstanding (double click)	Indicates the user has executed an outstanding work item. Fires when the user opens the case via the work item list, then double-clicks an outstanding work item on the **Outstanding** tab. This event also fires when the user presses **Enter** when an outstanding work item is already selected (highlighted).

Event	Description
Open Case Outstanding Work Item(s)	Indicates the user has opened an outstanding work item. Fires when the user clicks the **Open Selected Work Item(s)** button on the **Outstanding** tab toolbar, or chooses the **Open Selected Work Item(s)** selection on the **Outstanding** tab **Tools** menu.
Case Data Update	Indicates the user has updated case data. Fires when the user opens the case via the work item list, then clicks the **Apply** button on the **Data** tab.

Work Item List Filtering

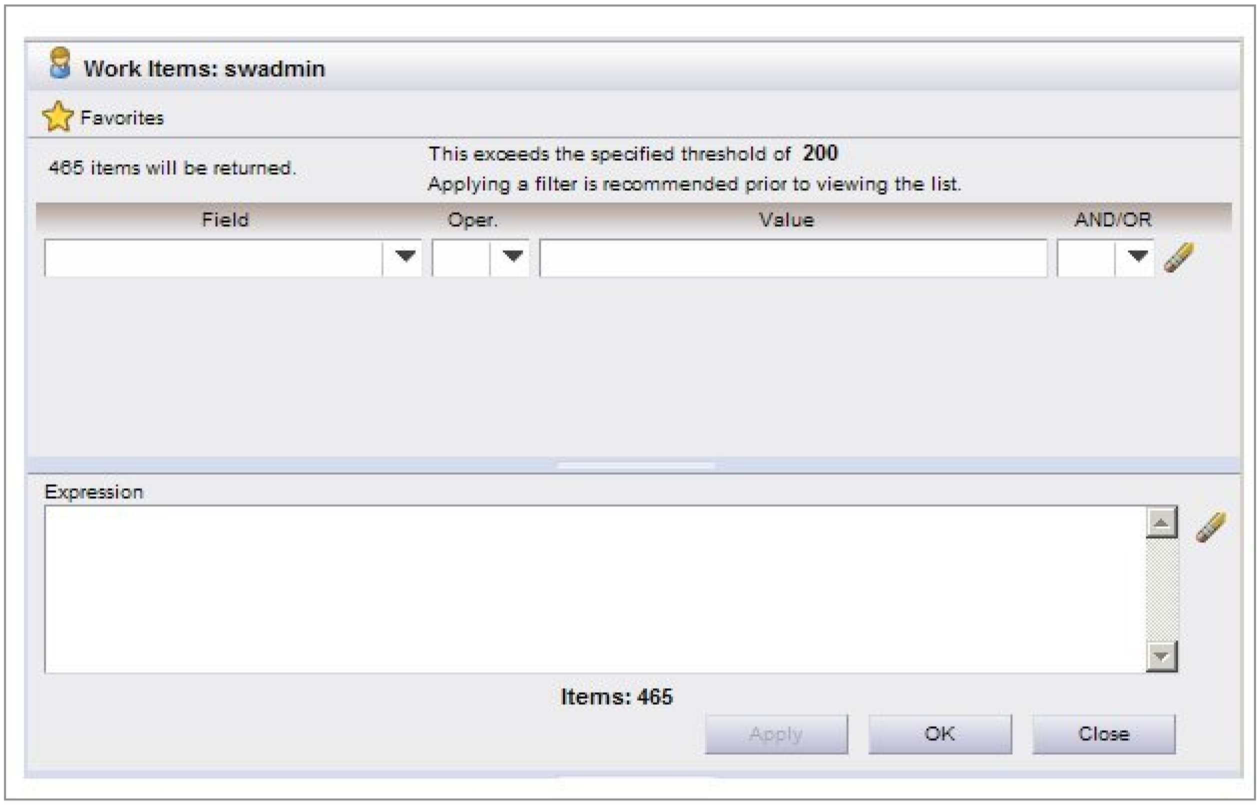
Filtering a list of work items involves entering filter criteria so that only some of the work items are shown in the list, rather than all of them. Filtering a list of work items allows you to display only the work items you are interested in. For example, you may only be interested in work items that arrived in the work queue after Dec. 15, 2004. You can filter the list so that only those work items are shown.

The filtering function allows you to build a “filter expression” that is applied to all work items for the work queue that was selected. If the work item satisfies the filter expression (e.g., it arrived after Dec. 15, 2004), it is shown in the work item list; if it does not satisfy the filter expression, it is not shown in the list (and is not downloaded from the iProcess Engine).

You can filter the work item list at any time by clicking on the **Filter** icon, or by selecting **Filter** from the **View** menu on the work item list. This displays the filter dialog box, on which you can specify a filter expression.

The filter dialog may also be automatically displayed when you attempt to display a list of work items by selecting a work queue from the work queue list. This occurs if the number of work items in the work queue exceeds a threshold that has been specified in the TIBCO iProcess Workspace (Browser) component application configuration file, `config.xml`. This threshold defaults to 500. (For more information about setting the work item list filter threshold, see *TIBCO iProcess Workspace (Browser) Configuration and Customization* guide.)

The following is an example filter dialog box:



465 items will be returned.

This exceeds the specified threshold of 200
Applying a filter is recommended prior to viewing the list.

Items: 465

Note:
If you had applied a filter expression the last time you viewed the work item list, that same filter expression will be applied when you open the list again — in other words, the most recently applied filter expression will remain persistent until you either change it or remove it (to remove an expression, click the “eraser” icon to the right of the **Expression** section).

If a filter expression is still active, the **Filter** icon is shown with a red check mark, and if the filter dialog is displayed, the previously entered filter expression is shown in the **Expression** section.

For more information about building a filter expression and how to apply the expression, see the *TIBCO iProcess Workspace (Browser) User Guide.*

Work Queues

The following component is available for displaying work queue information in lists:

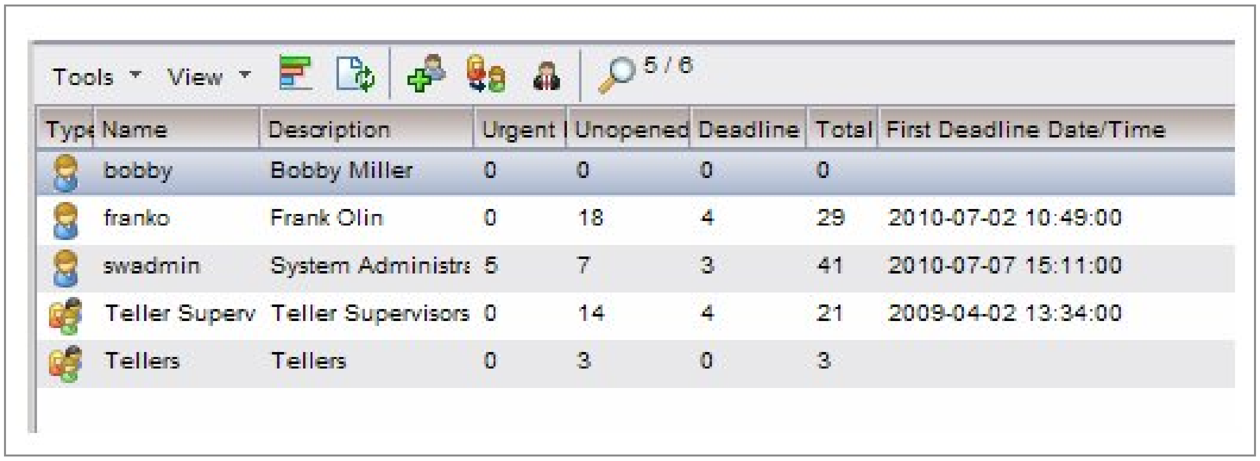
— **UserWorkQs** - list of all work queues to which the logged-in user has access

The following subsections provide details of this component.

UserWorkQs

The **UserWorkQs** component lists all work queues (user and group queues) to which the logged-in user has access.

An example work queue list is illustrated below, including callouts that show the functions available from the work queue list toolbar. Note that these functions are also available from the **Tools** and **View** menus on the work queue list:



Type	Name	Description	Urgent I	Unopened	Deadline	Total	First Deadline Date/Time
	bobby	Bobby Miller	0	0	0	0	
	franko	Frank Olin	0	18	4	29	2010-07-02 10:49:00
	swadmin	System Administr	5	7	3	41	2010-07-07 15:11:00
	Teller Superv	Teller Supervisors	0	14	4	21	2009-04-02 13:34:00
	Tellers	Tellers	0	3	0	3	

The following describes the functions available from the work queue list:

- **Work Queue Loading** - Displays a graphical summary of the work queues.
- **Refresh Work Queues** - Requests that new data be retrieved from the server, refreshing the work queue list.
- **Manage Participation** - Allows the user to manage participation schedules, which specify dates and times that "participant" users will have access to another user's work queue.
- **Manage Redirection** - Allows the user to manage redirection schedules, which specify dates and times that work items destined for a particular user's work queue will be redirected to another user's work queue.
- **Manage Supervisors** - Allows the user to specify who the "work queue supervisors" are for a particular work queue. A user must be designated as a work queue supervisor to manage participation or redirection schedules for a work queue.
- **Find** - Used to find the desired work queue(s) by entering search criteria.

- **Select Columns** - Used to display the desired columns in the procedure list. (This is available from the procedure list **View** menu.)

If the currently signed-on user does not have access privileges to a particular function, the icon/menu selection for that function does not display. Access to each of the functions available on the procedure list is controlled by properties on the **UserWorkQs** component. For more information about setting these properties, see UserWorkQs Component Properties.

The **UserWorkQs** component also publishes events for some of the functions on the work queue list. For more information about these events, see UserWorkQs Component Events.

For more information about using the functions listed above (e.g., managing participation, managing supervisors, etc.), see the *TIBCO iProcess Workspace (Browser) User Guide.*

Required Data

The **UserWorkQs** component does not need data to render, although there must have been a previous login because it needs a valid user and server name.

UserWorkQs Component Properties

The **UserWorkQs** components contain the following properties, which are used to control access to each of the functions available from the work queue list.

Property	Description
Work Queue View	Enables/disables access to the work queue list. This property controls rendering of the work queue list. This would be used to control access from an external application, rather than from an event from another component (such as the Login component).
Loading Chart	Enables/disables the ability for the user to display the work queue loading chart. This property controls access to the **Work Queue Loading Chart** button on the toolbar, as well as the **Work Queue Loading Chart** selection on the work queue list **Tools** menu.
Participation	Enables/disables the ability for the user to manage participation schedules.

Property	Description
	This property controls access to the **Manage Participation** button on the toolbar, as well as the **Manage Work Queue Participation** selection on the work queue list **Tools** menu.
Redirection	Enables/disables the ability for the user to manage redirection schedules. This property controls access to the **Manage Redirection** button on the toolbar, as well as the **Manage Work Queue Redirection** selection on the work queue list **Tools** menu.
Supervisors	Enables/disables the ability for the user to manage supervisors. This property controls access to the **Manage Supervisors** button on the toolbar, as well as the **Manage Work Queue Supervisors** selection on the work queue list **Tools** menu.
Select Columns	Enables/disables the ability for the user to specify which columns to display in the work queue list. This property controls access to the **Select Columns** selection on the work queue list **View** menu.

UserWorkQs Component Events

The **UserWorkQs** component contains the following events, which fire when the action described by the event occurs:

Event	Description
List Item Select (single click)	Indicates the user has selected a work queue in the work queue list. Fires when the user single-clicks a work queue in the work queue list, or when the user moves the highlight bar on the work queue list using the keyboard arrow keys.
List Item Execute (double click)	Indicates the user has executed a work queue in the work queue list. Fires when the user double-clicks a work queue in the work queue list. This event also fires when the user presses **Enter** when a work queue is already

Event	Description
	selected (highlighted).
Loading Chart	Indicates the user has requested to view the work queue loading chart. Fires when the user clicks on the **Work Queue Loading Chart** toolbar button, or chooses the **Work Queue Loading Chart** menu selection on the work queue list.
Participation	Indicates the user has requested to manage participation schedules. Fires when the user clicks on the **Manage Participation** toolbar button, or chooses the **Manage Work Queue Participation** menu selection on the work queue list.
Redirection	Indicates the user has requested to manage redirection schedules. Fires when the user clicks on the **Manage Redirection** toolbar button, or chooses the **Manage Work Queue Redirection** menu selection on the work queue list.
Supervisors	Indicates the user has requested to manage work queue supervisors. Fires when the user clicks on the **Manage Supervisors** toolbar button, or chooses the **Manage Work Queue Supervisors** menu selection on the work queue list.

WCC Tools Methods

This section describes the WCC tools methods available in TIBCO iProcess Workspace (Browser).

Introduction

The WCC tools methods allow you to perform application functions through method calls when using a customized WCC application. (Note that there are also equivalent methods called the "IPC" tools methods that can be used with the iProcess Client application. Those methods are described in the *TIBCO iProcess Workspace (Browser) Configuration and Customization* guide.)

These static methods provide the same functionality that is available through the client application (as well as the iProcess Workspace (Browser) components), such as starting a case and opening a work item.

Most of the WCC tools methods expect a tag of some sort (for example, 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 components, save it somewhere, then use it at a later time in a method call in your application to perform a function (like opening a work item).

  In this scenario, you would typically disable the default handler for an action on a component from which you acquired the tag so that you could perform, when desired, the function that the handler would have performed by default. For example, you may want to perform some custom business logic when the user double-clicks on a work item in a work item list, then have the work item open. In this case, you would disable the default handler for the **Open** property on the **WorkItems** component, then subscribe to the **List Item Execute (double click)** event on the **WorkItems** component. When the user double-clicks on a work item, you can perform your custom logic, then call the **openWorkItem** WCC tools method to open the work item.

For more information about disabling the default handler, see the *Using the Properties Editor* section in the *TIBCO iProcess Workspace (Browser) Components Concepts* guide.

- 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*.

Note: The WCC tools methods are available only if the components portion of the iProcess Workspace (Browser) is installed. For more information, see the *TIBCO iProcess Workspace (Browser) Installation* guide.

Application Class

The **com.tibco.bpm.***WCCProjectName***.Application** class is the main application-level class for custom iProcess applications running in the TIBCO® General Interface environment. This class contains entry points for custom code, including code that calls the static methods described in this section.

The **Application** class is available in the following location:

```
GIWorkspaceDir\JSXAPPS\WCCProjectName\application\js\Application.js
```

where *GIWorkspaceDir* is the TIBCO® General Interface workspace directory, and *WCCProjectName* is the name you gave your project/application in TIBCO® General Interface Builder.

Note:
Package-level custom code can be added to the following file:

```
GIWorkspaceDir\JSXAPPS\WCCProjectName\components\js\APP.js
```

where *GIWorkspaceDir* is the TIBCO® General Interface workspace directory, and *WCCProjectName* is the name you gave your project/application in TIBCO® General Interface Builder.

WCC Tools Methods

This section describes the WCC tools methods available in the iProcess Workspace (Browser).

To use the WCC tools methods, you must provide the full path to the method. For example, to call the **startCase** method, you call:

```
com.tibco.bpm.wcc.Tools.startCase(this.getApp(), procTag)
```

Note that the path is case sensitive, i.e., “Tools” must be capitalized.

The following are the methods available:

- Case Functions
  - startCase - Starts a case of a procedure.
  - closeCases - Closes one or more cases.
  - purgeCases - Purges (permanently deletes) one or more cases.
  - suspendCases - Suspends one or more cases.
  - activateCases - Reactivates one or more suspended cases.
  - viewGraphicalCaseHistory - Displays the case history in a graphical format.
  - addCaseHistoryEntry - Adds an entry to the case history.
  - viewCasePrediction - Predicts the outcome of the case.
  - triggerEvent - Starts process flow from an event step.
  - processJump - Changes the process flow in a case.
- Work Item Functions
  - openWorkItem - Opens (locks) a work item by passing in a work item tag.
  - openWorkItemEx - Opens (locks) a work item by passing in parameters that identify the work item.
  - unlockWorkItem - Unlocks a work item.
  - forwardWorkItem - Forwards a work item to a different work queue.
  - releaseWorkItem - Releases a work item.
- Work Queue Functions

  - configureSupervisors - Allows set up of work queue supervisors.
  - configureParticipation - Allows set up of participation schedules.
  - configureRedirection - Allows set up of redirection schedules.
  - viewWorkQLoadingChart - Displays a graphical work queue summary.

- Procedure Functions
  - getStartProcs - Returns the procedures for which the logged-in user has permission to start cases.
  - getAuditProcs - Returns the procedures for which the logged-in user has permission to view or add entries to case history.
  - viewProcLoadingChart - Displays a graphical procedure summary.
  - loadProcVersion - Displays version history information for the procedure.
- Other Functions
  - loadServerInfo - Displays information about the iProcess Objects Server.
  - workspaceOptions - Displays interface for establishing default settings.
  - workItemTag2CaseTag - Returns a case tag that is extracted from the work item tag passed as an argument.
  - workItemTag2WorkQTag - Returns a work queue tag that is extracted from the work item tag passed as an argument.
  - getUserAttributes - Returns an array of objects that represent iProcess attributes assigned to a specific user.
  - getGroupAttributes - Returns an array of objects that represent iProcess attributes assigned to a specific group.
  - manageMER - Displays the interface for managing Monitor Event Request (MER) messages on the server.

Namespace Reference

All of the WCC tools methods require a *namespaceRef* parameter. This parameter specifies a reference to the application. It can be one of the following:

  - String - If the namespace reference is a string, it must be the **namespace** property value assigned in your application's `config.xml` file. This is usually, but not always, the name of your TIBCO® General Interface project.

— jsx3.gui.* object - If the namespace reference is a jsx3.gui.* object, it must be a GUI object that is in your application. It can be a WCC component, but it can also be any other component in your application. The method can acquire the application reference through this GUI object.

— 'this.getApp()' - You can include '`this.getApp()`', which references the TIBCO® General Interface Server, **jsx3.app.Server**. For example:

```
com.tibco.bpm.wcc.Tools.startCase(this.getApp(),
"myserver|CARPOOL|1|2")
```

If the *namespaceRef* lookup by the method does not result in an application object, an exception is thrown.

Login Required

All of the WCC tools methods require that the user be logged in prior to the method being called. A login can be accomplished in a number of ways, for example:

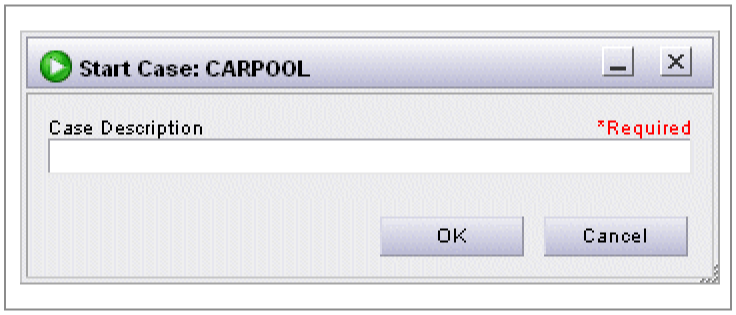
- using the WCC **Login** component — see Login
- using *single authentication*, which allows the user to be authenticated using credentials the user has already entered in another application — see the *TIBCO iProcess Workspace (Browser) Configuration and Customization* guide
- using the Action Processor **Login** request — see the *TIBCO iProcess Workspace (Browser) Action Processor Reference*

startCase

This static method starts a case of the specified procedure.

The **startCase** method displays the following dialog box, which allows the user to enter a case description[1], then start the case by clicking the **OK** button:

[1]The case description may or may not be required, depending on how the procedure was configured. If the case description is required, "*Required" is shown on the **Start Case** dialog box.



This method is equivalent to selecting **Start New Case** from the **Tools** menu on the procedure list in the iProcess Workspace (Browser) client application.

When the user clicks **OK**, a form is opened if the addressee of the first step in the procedure is SW_STARTER. The context in which the form is displayed depends on how the form was created, as follows:

- iProcess Modeler Forms - These forms are always displayed in a new browser window.
- TIBCO® General Interface Builder Forms - These forms (also known as "GI Forms) are always displayed in a separate dialog box.
- TIBCO Forms - These forms are always displayed in a separate dialog box.

To start a case, the user must have permission to start cases of the particular procedure. To determine the procedures the user has permission to start, use the getStartProcs method.

Syntax

```
com.tibco.bpm.wcc.Tools.startCase(nameSpaceRef,
                                  procTag,
                                  parent);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For more information, see Namespace Reference.

procTag - (String) Identifies the procedure for which you want to start a case. For more information about tags, see Introduction.

parent - This identifies a TIBCO® General Interface component (object) in which the work item form can be placed. This component must implement the TIBCO® General Interface **setChild()** method; if it doesn't, an exception is thrown. Note that this parameter is only applicable if the form is displayed in a separate dialog box, i.e., you are using GI Forms and TIBCO Forms. For all other form types (e.g., iProcess Modeler-produced forms), this parameter is ignored.

Example

```
* Event handler for startCase
*/
wccClass.prototype.startCaseEventHandler = function(objEvent) {
   com.tibco.bpm.ipc.log('startCaseEventHandler');
   var procTag = 'myserver|CARPOOL|1|2';
   var parentPane = this.getAppBlock().getDescendantOfName
('parentPane');
   try {
   com.tibco.bpm.wcc.Tools.startCase(this.getApp(), procTag,
     parentPane);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

closeCases

This static method closes the specified active cases of a procedure. This stops the process flow for the cases.

You must have system administrator authority to close cases.

An optional confirmation message can be displayed.

This method is equivalent to selecting **Close Case(s)** from the **Tools** menu on the case list in the iProcess Workspace (Browser) client application.

Syntax

```
com.tibco.bpm.wcc.Tools.closeCases(namespaceRef,
                                   caseTags,
                                   suppressConfirm,
                                   doEvents);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

caseTags - (String or Array of Strings) Identifies the case(s) to close. For more information about tags, see Introduction.

suppressConfirm - (Boolean - Optional) Specifies whether or not to suppress the confirmation message. False (default) causes a confirmation message to be displayed; True suppresses the confirmation message.

doEvents - (Boolean - Optional) When true, a procedure-level event is triggered when the cases are closed. The procedure can be defined to catch the event, then perform business logic either before or after the case is closed. This parameter is only applicable when the *supressConfirm* parameter is true (when it is false, the user specifies this value on the confirmation dialog). This parameter is only relevant when an iProcess Engine version 11.4.0 or newer is being used. For earlier versions, this parameter is ignored. The default is true if not passed (and the engine version is 11.4.0 or newer).

Example

```
* Event handler for closeCases
*/
wccClass.prototype.closeCasesEventHandler = function(objEvent) {
   com.tibco.bpm.ipc.log('closeCasesEventHandler');
   var caseTags = 'myserver|HIRING|1|2|405';
   try {
   com.tibco.bpm.wcc.Tools.closeCases(this.getApp(), caseTags,
   true, true);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

purgeCases

This static method purges the specified cases of a procedure. Purging cases permanently deletes them from the system. You can purge both active and closed cases.

The user must have system administrator authority to purge cases.

An optional confirmation message can be displayed.

This method is equivalent to selecting **Purge Case(s)** from the **Tools** menu on the case list in the iProcess Workspace (Browser) client application.

Syntax

```
com.tibco.bpm.wcc.Tools.purgeCases(namespaceRef,
                                   caseTags,
                                   suppressConfirm)
                                   doEvents);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For more information, see Namespace Reference.

caseTags - (String or Array of Strings) Identifies the case(s) to purge. For more information about tags, see Introduction.

suppressConfirm - (Boolean - Optional) Specifies whether or not to suppress the confirmation message. False (default) causes a confirmation message to be displayed; True suppresses the confirmation message.

doEvents - (Boolean - Optional) When true, a procedure-level event is triggered when the cases are purged. The procedure can be defined to catch the event, then perform business logic before the cases are purged. This parameter is only applicable when the *supressConfirm* parameter is true (when it is false, the user specifies this value on the confirmation dialog). This parameter is only relevant when an iProcess Engine version 11.4.0 or newer is being used. For earlier versions, this parameter is ignored. The default is true if not passed (and the engine version is 11.4.0 or newer).

Example

```
* Event handler for purgeCases
*/
wccClass.prototype.purgeCasesEventHandler = function(objEvent) {
   com.tibco.bpm.ipc.log('purgeCasesEventHandler');
   var caseTags = 'myserver|LOAN|0|1|2057';
   try {
   com.tibco.bpm.wcc.Tools.purgeCases(this.getApp(), caseTags,
   true, true);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

suspendCases

This static method suspends one or more cases. Note that when you suspend a case, you are suspending the entire *case family*, which includes the main case and all of its sub-cases, if any.

When a case is suspended, current work items from that case can no longer be opened.

If a work item is already open when the case is suspended, the work item can still be kept, which causes the work item to become immediately suspended, and it cannot be opened again until the case is reactivated (see activateCases). The opened work item can also be released; this causes any new work items as a result of the release to become immediately suspended (unless they are flagged to ignore suspensions).

For more information about case suspensions, see the *TIBCO iProcess Workspace (Browser) User Guide*.

An optional confirmation message can be displayed.

This method is equivalent to selecting **Suspend Case(s)** from the **Tools** menu on the case list in the iProcess Workspace (Browser) client application.

Syntax

```
com.tibco.bpm.wcc.Tools.suspendCases(namespaceRef,
                                     caseTags,
                                        suppressConfirm);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For more information, see Namespace Reference.

caseTags - (String or Array of Strings) Identifies the case(s) to suspend. For more information about tags, see Introduction.

suppressConfirm - (Boolean - Optional) Specifies whether or not to suppress the confirmation message. False (default) causes a confirmation message to be displayed; True suppresses the confirmation message.

Example

```
* Event handler for suspendCases
*/
wccClass.prototype.suspendCasesEventHandler = function(objEvent) {
   com.tibco.bpm.ipc.log('suspendCasesEventHandler');
   var caseTags = 'myserver|LOAN|0|1|2057';
   try {
   com.tibco.bpm.wcc.Tools.suspendCases(this.getApp(), caseTags);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

activateCases

This static method reactivates one or more suspended cases (see suspendCases), which causes the process to flow as usual. Work items that were suspended because the case they are a part of was suspended can now be opened and processed normally.

An optional confirmation message can be displayed.

This method is equivalent to selecting **Activate Case(s)** from the **Tools** menu on the case list in the iProcess Workspace (Browser) client application.

Syntax

```
com.tibco.bpm.wcc.Tools.activateCases(namespaceRef,
                                      caseTags,
                                      suppressConfirm);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For more information, see Namespace Reference.

caseTags - (String or Array of Strings) Identifies the suspended case(s) to activate. For more information about tags, see Introduction.

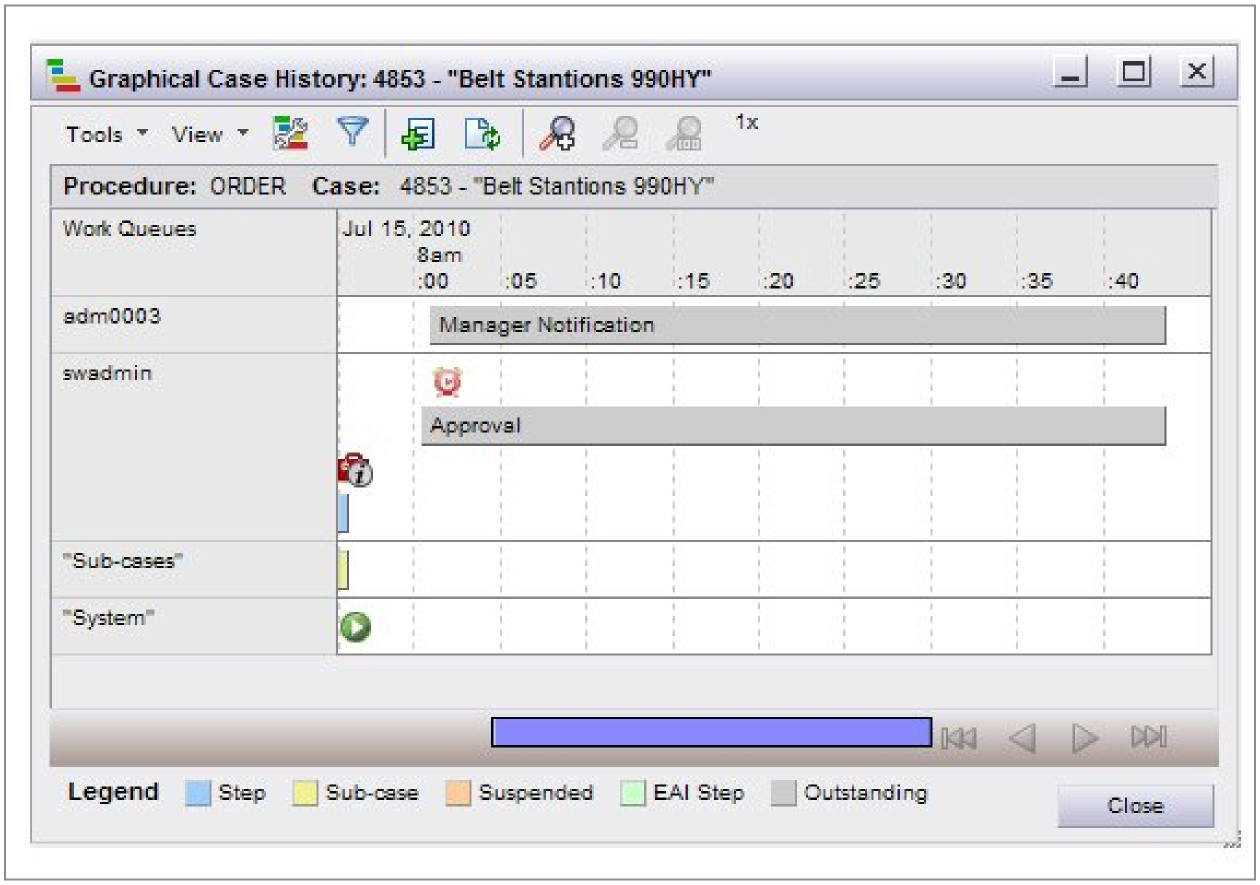
suppressConfirm - (Boolean - Optional) Specifies whether or not to suppress the confirmation message. False (default) causes a confirmation message to be displayed; True suppresses the confirmation message.

Example

```
* Event handler for activateCases
*/
wccClass.prototype.activateCasesEventHandler = function(objEvent) {
   com.tibco.bpm.ipc.log('activateCasesEventHandler');
   var caseTags = 'myserver|LOAN|0|1|2057';
   try {
   com.tibco.bpm.wcc.Tools.activateCases(this.getApp(), caseTags);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

viewGraphicalCaseHistory

This static method displays the case history for the specified case in a graphical format. Example:



This method is equivalent to selecting **Graphical History** from the **View** menu on the case's **Summary** tab.

To view the graphical case history, the user must have permission to audit the procedure (of which the case is an instance). To determine the procedures the user has permission to audit, use the getAuditProcs method.

For more information about using the graphical case history, see the *TIBCO iProcess Workspace (Browser) User Guide*.

Syntax

```
com.tibco.bpm.wcc.Tools.viewGraphicalCaseHistory(namespaceRef,
                                                 caseTag);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For more information, see Namespace Reference.

caseTag - (String) Identifies the case whose history to display in a graphical format. For more information about tags, see Introduction.

Example

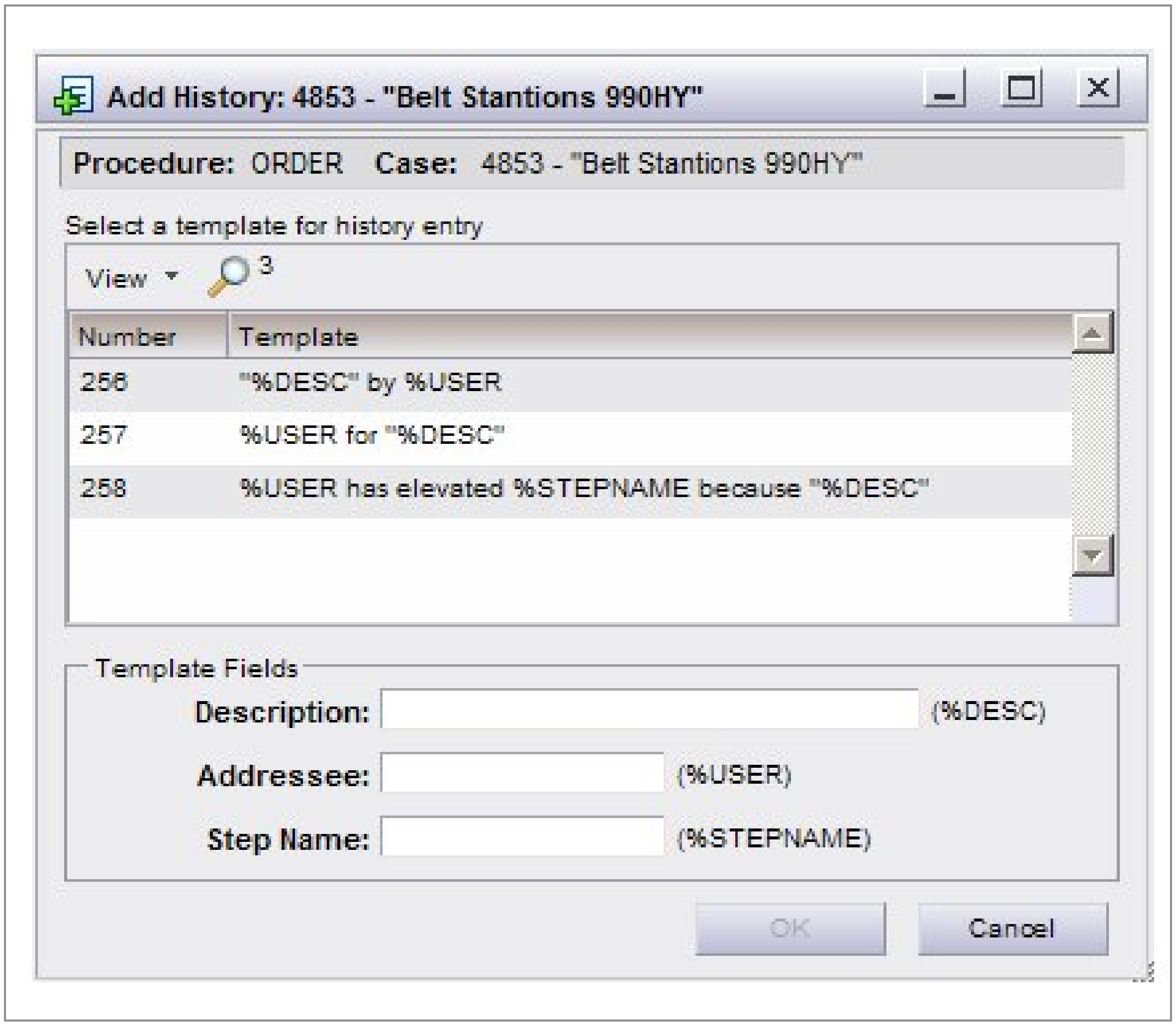
```
* Event handler for viewGraphicalCaseHistory
*/
wccClass.prototype.viewGraphicalCaseHistoryEventHandler = function
(objEvent) {
   com.tibco.bpm.ipc.log('viewGraphicalCaseHistoryEventHandler');
   var caseTag = 'myserver|LOAN|0|1|2057';
   try {
   com.tibco.bpm.wcc.Tools.viewGraphicalCaseHistory(this.getApp(),
     caseTag);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

addCaseHistoryEntry

This static method allows the user to manually add an entry to a case history.

This requires that a file (**auditusr.mes** file) be set up on the system that contains predefined messages that you can add to the case history.

This method causes a dialog similar to the following to be displayed:



Procedure: ORDER Case: 4853 - "Belt Stantions 990HY"

Select a template for history entry

Number	Template
256	"%DESC" by %USER
257	%USER for "%DESC"
258	%USER has elevated %STEPNAME because "%DESC"

Template Fields

Description: (%DESC)

Addressee: (%USER)

Step Name: (%STEPNAME)

This dialog presents the messages that have been added to the **auditusr.mes** file. It allows the user to select which message to add to the case history.

Note: If your iProcess Engine does not support Add Case History “Templates”, the **Add History** dialog contains a **Message Number** field instead of the list of available message numbers. If your system does not support templates, enter the message number in the **Message Number** field.

This method is equivalent to selecting **Add Entry** from the **Tools** menu on the case’s **History** tab.

To add an entry to case history, the user must have permission to audit the procedure (of which the case is an instance). To determine the procedures the user has permission to audit, use the getAuditProcs method.

For more information about adding case history entries, see the *TIBCO iProcess Workspace (Browser) User Guide*.

Syntax

```
com.tibco.bpm.wcc.Tools.addCaseHistoryEntry(namespaceRef,
                                            caseTag);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For more information, see Namespace Reference.

caseTag - (String) Identifies the case to which you are adding an entry to case history. For more information about tags, see Introduction.

Example

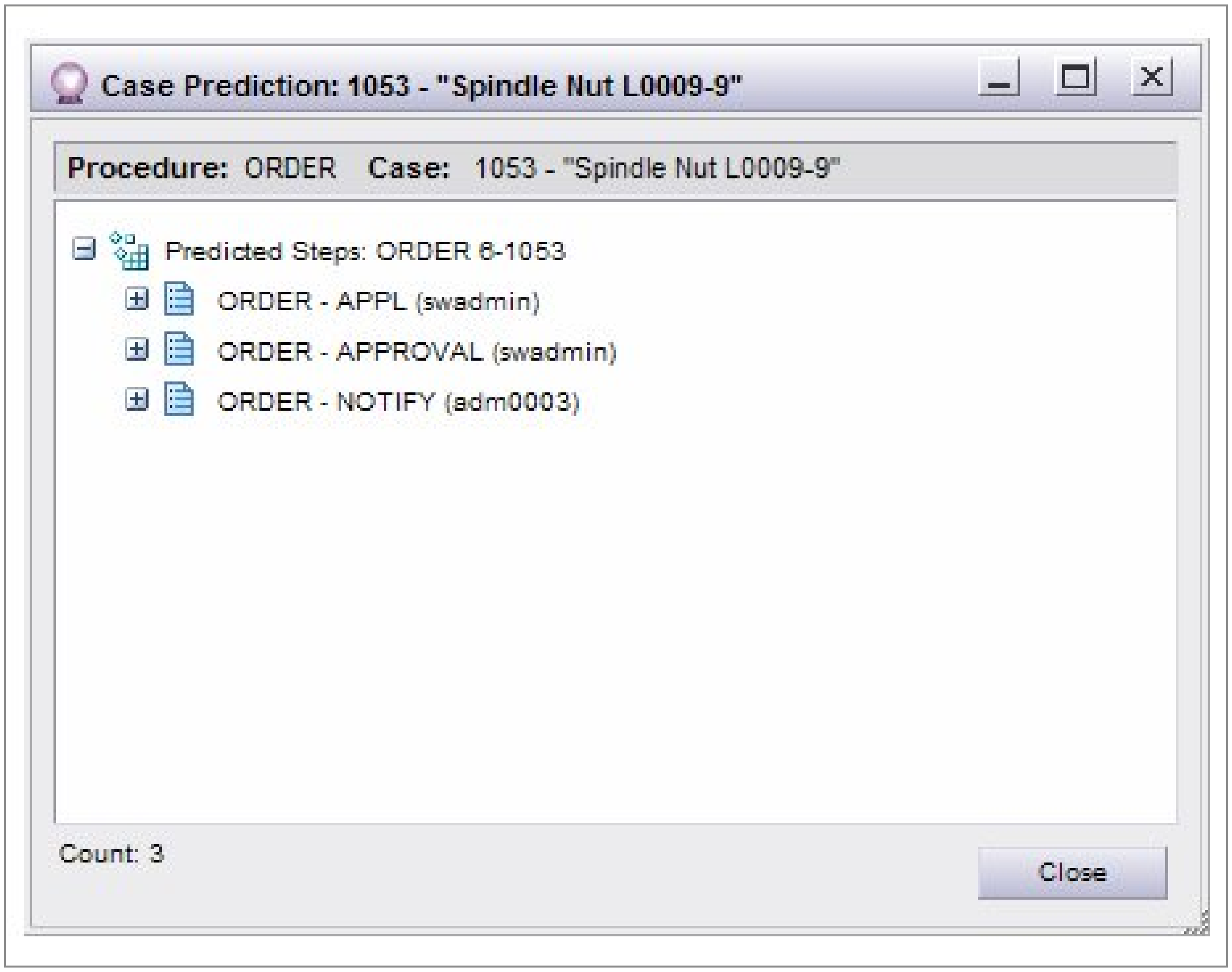
```
* Event handler for addCaseHistoryEntry
*/
wccClass.prototype.addCaseHistoryEntryEventHandler = function(objEvent)
{
   com.tibco.bpm.ipc.log('addCaseHistoryEntryEventHandler');
   var caseTag = 'myserver|LOAN|0|1|2057';
   try {
   com.tibco.bpm.wcc.Tools.addCaseHistoryEntry(this.getApp(),
     caseTag);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

viewCasePrediction

This static method is used to predict the expected outcome of the specified live case. Running the case prediction function causes a list of “predicted work items” to be returned that represent the work items that are currently due (outstanding work items), as well as the work items that are expected to be due in the future.

Included with the predicted work items returned is information about the expected times the work items are predicted to start and end, providing information that can be used to predict the outcome of the case. This can be used to improve work forecasting and estimate the expected completion of cases.

This method causes a dialog similar to the following to be displayed:


Case Prediction: 1053 - "Spindle Nut L0009-9"
Procedure: ORDER Case: 1053 - "Spindle Nut L0009-9"
Predicted Steps: ORDER 6-1053
ORDER - APPL (swadmin)
ORDER - APPROVAL (swadmin)
ORDER - NOTIFY (adm0003)
Count: 3
Close

This provides a list of the *predicted steps* — the currently outstanding steps, and steps predicted to be outstanding as the case is processed to completion. For each step, it also indicates in parentheses the addressee of the step.

This method is equivalent to selecting **Predict Case** from the **Tools** menu on the case's **History** tab.

For more details about using case prediction, see the *TIBCO iProcess Workspace (Browser) User Guide*.

Syntax

```
com.tibco.bpm.wcc.Tools.viewCasePrediction(namespaceRef,
                                           caseTag);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For more information, see Namespace Reference.

caseTag - (String) Identifies the case on which you want to perform a case prediction function. For more information about tags, see Introduction.

Example

```
* Event handler for viewCasePrediction
*/
wccClass.prototype.viewCasePredictionEventHandler = function(objEvent) {
   com.tibco.bpm.ipc.log('viewCasePredictionEventHandler');
   var caseTag = 'myserver|LOAN|0|1|2057';
   try {
   com.tibco.bpm.wcc.Tools.viewCasePrediction(this.getApp(),
     caseTag);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

triggerEvent

This static method is used to start the process flow from an event step in the procedure.

An event step is a step in a procedure that allows you to control the process flow in various ways, depending on how your procedure was designed. It can be used to perform actions such as:

- Suspending the flow of a case until an external action takes place.
- Starting a parallel branch in a case.
- Pausing a case for a specific period of time.

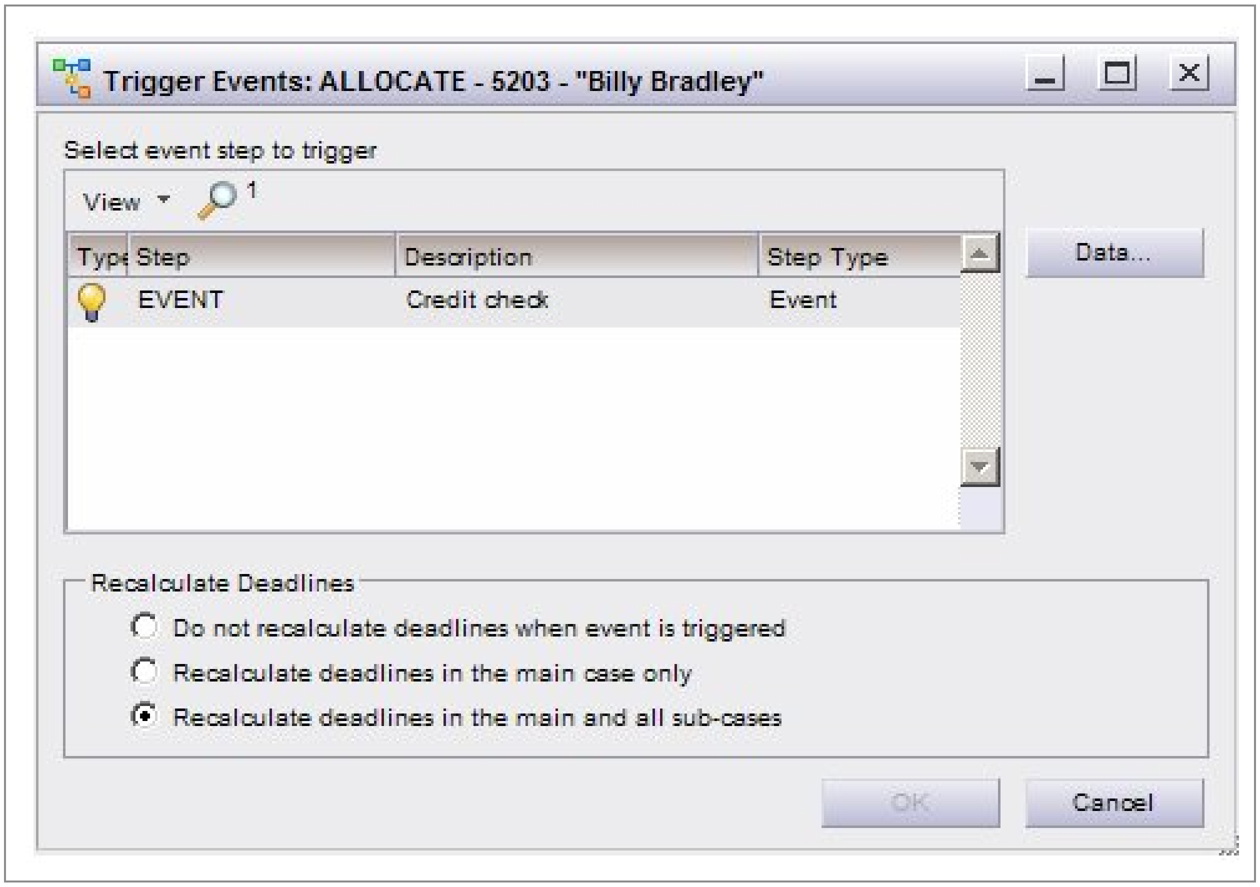
When the process flow reaches an event step, process flow is halted, and remains halted, until the user triggers the event with the **triggerEvent** method. When the event is triggered, the process flow will continue again.

However, an event step does not need to be outstanding (i.e., process flow has reached the step) to be triggered. You can trigger an event step at any time, such as:

- before the process flow has reached the event step,
- after the process flow has been halted at the event step, or
- after the event step has been triggered — one event step can be triggered multiple times. This allows you to run a segment of the procedure at any time, as many times as necessary.

Also note that other actions can be performed when an even step is triggered. These include resurrecting a closed case, as well as recalculating deadlines in the case. These actions and other details about triggering events are described in the *TIBCO iProcess Workspace (Browser) User Guide*.

The **triggerEvent** method causes a dialog similar to the following to be displayed:


Trigger Events: ALLOCATE - 5203 - "Billy Bradley"

Select event step to trigger

Type	Step	Description	Step Type
💡	EVENT	Credit check	Event

Recalculate Deadlines

- ( ) Do not recalculate deadlines when event is triggered
- ( ) Recalculate deadlines in the main case only
- (●) Recalculate deadlines in the main and all sub-cases

This dialog allows the user to select the event step from which the process flow should begin.

This method is equivalent to selecting **Trigger Event** from the **Tools** menu on the case's **Summary** tab.

Syntax

```
com.tibco.bpm.wcc.Tools.triggerEvent(namespaceRef,
                                     caseTag);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

caseTag - (String) Identifies the case on which you want to trigger the event. For more information about tags, see Introduction.

Example

```
* Event handler for triggerEvent
*/
wccClass.prototype.triggerEventEventHandler = function(objEvent) {
   com.tibco.bpm.ipc.log('triggerEventEventHandler');
   var caseTag = 'myserver|LOAN|0|1|1234';
   try {
   com.tibco.bpm.wcc.Tools.triggerEvent(this.getApp(), caseTag);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

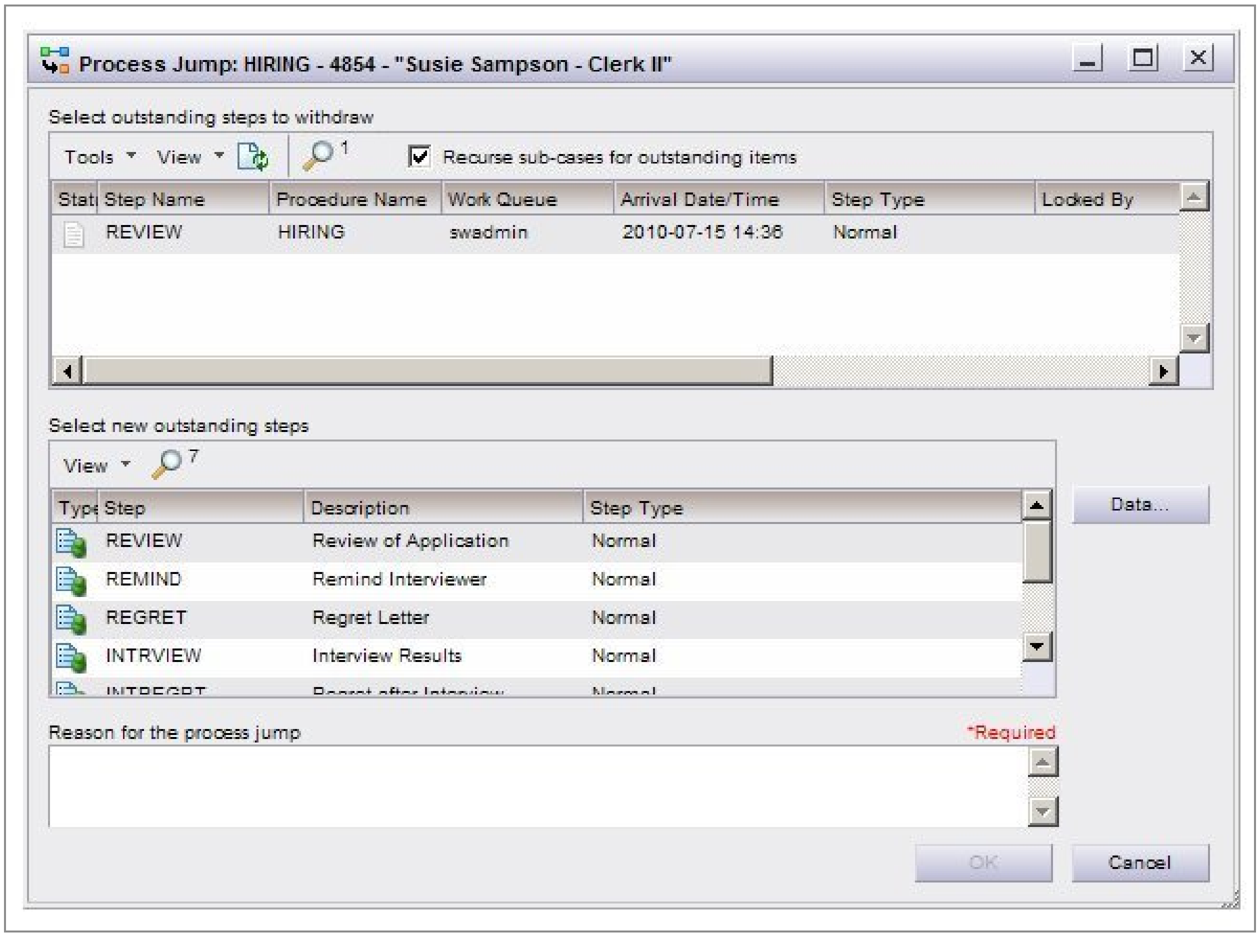
processJump

This static method is used to change the process flow in the following ways:

- You can select currently outstanding steps you would like to "withdraw," i.e., make them no longer outstanding.

- You can specify a set of steps to “jump to,” making those steps the new outstanding items.

Calling this method causes the following dialog to be displayed:


Process Jump: HIRING - 4854 - "Susie Sampson - Clerk II"
Select outstanding steps to withdraw
Tools
View
1
Recurse sub-cases for outstanding items
Select new outstanding steps
View
7
Data...
Reason for the process jump
*Required
OK
Cancel

Stat	Step Name	Procedure Name	Work Queue	Arrival Date/Time	Step Type	Locked By
	REVIEW	HIRING	swadmin	2010-07-15 14:36	Normal	

Type	Step	Description	Step Type
	REVIEW	Review of Application	Normal
	REMIND	Remind Interviewer	Normal
	REGRET	Regret Letter	Normal
	INTRVIEW	Interview Results	Normal
	INTREGRT	Regret after Interview	Normal

This dialog allows the user to select the outstanding steps to withdraw, as well as the new steps to make outstanding.

For more details about using the process jump function, see the *TIBCO iProcess Workspace (Browser) User Guide*.

Syntax

```
com.tibco.bpm.wcc.Tools.processJump(namespaceRef,
                                    caseTag);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

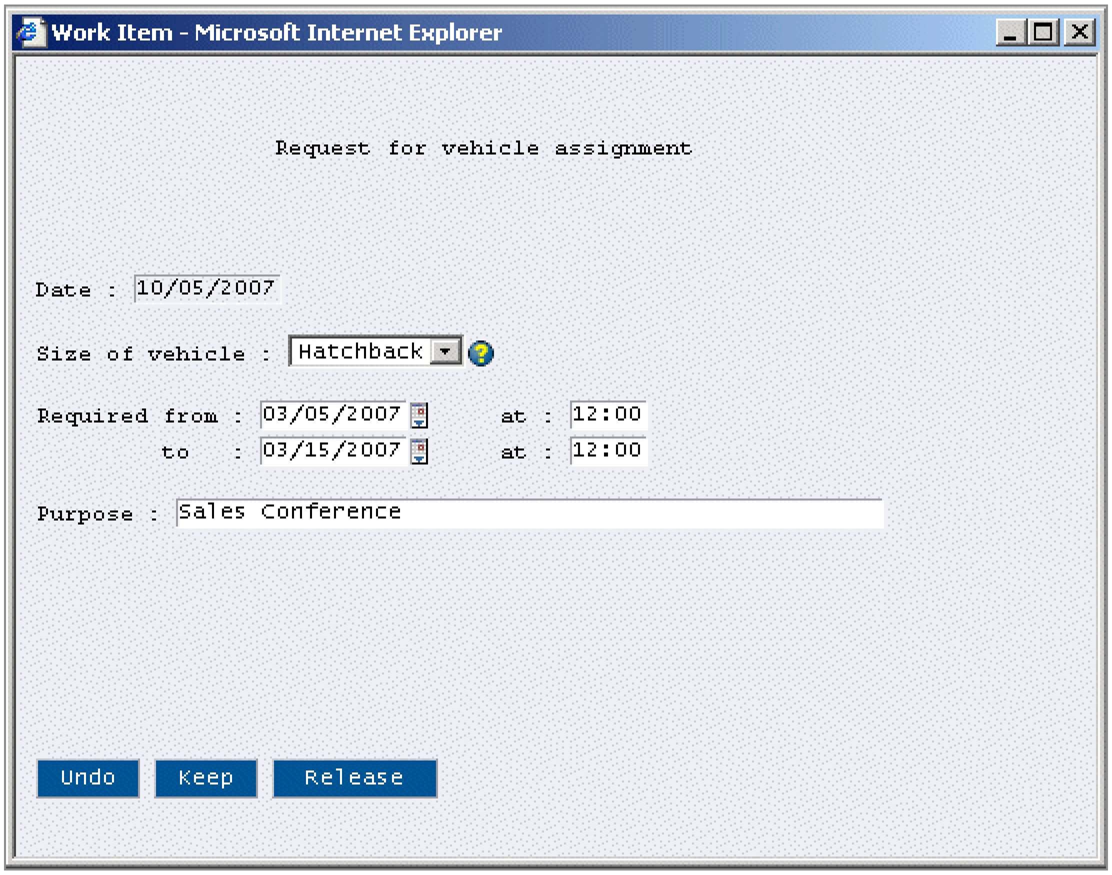
caseTag - (String) Identifies the case on which you want to change the process flow. For more information about tags, see Introduction.

Example

```
* Event handler for processJump
*/
wccClass.prototype.processJumpEventHandler = function(objEvent) {
   com.tibco.bpm.ipc.log('processJumpEventHandler');
   var caseTag = 'myserver|LOAN|0|1|1234';
   try {
   com.tibco.bpm.wcc.Tools.processJump(this.getApp(), caseTag);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

openWorkItem

This static method opens (and locks) the specified work item and displays the form associated with that work item. For example:



Request for vehicle assignment

Date : 10/05/2007

Size of vehicle : Hatchback

Required from : 03/05/2007 at : 12:00
to : 03/15/2007 at : 12:00

Purpose : Sales Conference

This method is equivalent to selecting **Open Selected Work Item(s)** from the **Tools** menu on the work item list in the iProcess Workspace (Browser) client application.

The context in which the form is displayed depends on how the form was created, as follows:

- iProcess Modeler Forms - These forms are always displayed in a new browser window (as in the example shown above).
- TIBCO® General Interface Builder Forms - These forms (also known as “GI forms”) are always displayed in a separate dialog.
- TIBCO Forms - These forms are always displayed in a separate dialog.

Also see the **openWorkItemEx** method on openWorkItemEx.

Syntax

```
com.tibco.bpm.wcc.Tools.openWorkItem(nameSpaceRef,
                                     workQTag,
                                     workItemTag
                                     [parent]);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

workQTag - (String) Identifies the work queue in which the work item resides. For more information about tags, see Introduction.

workItemTag - (String) Identifies the work item to open. For more information about tags, see Introduction.

parent - (optional) This identifies a TIBCO® General Interface component (object) in which the work item form can be placed. This component must implement the TIBCO® General Interface **setChild()** method; if it doesn't, an exception is thrown. If this parameter is not defined, external forms (ASP Forms, JSP Forms, BusinessWorks FormBuilder Forms, and iProcess Modeler Forms) always load in a separate browser window, whereas GI Forms (custom TIBCO® General Interface Forms and TIBCO Forms) load in a dialog or browser window, based on settings in `config.xml`. A custom GI Form will load in a browser or dialog based on the value of the **floatWorkItems** attribute in the form definition (<**Form**> element in `config.xml`); if it's not specified in the form definition, the value of the **floatWorkItems** attribute in the <**options**> record is used. For TIBCO Forms, the value of the **floatWorkItems** attribute in the <**options**> record is used.

Example

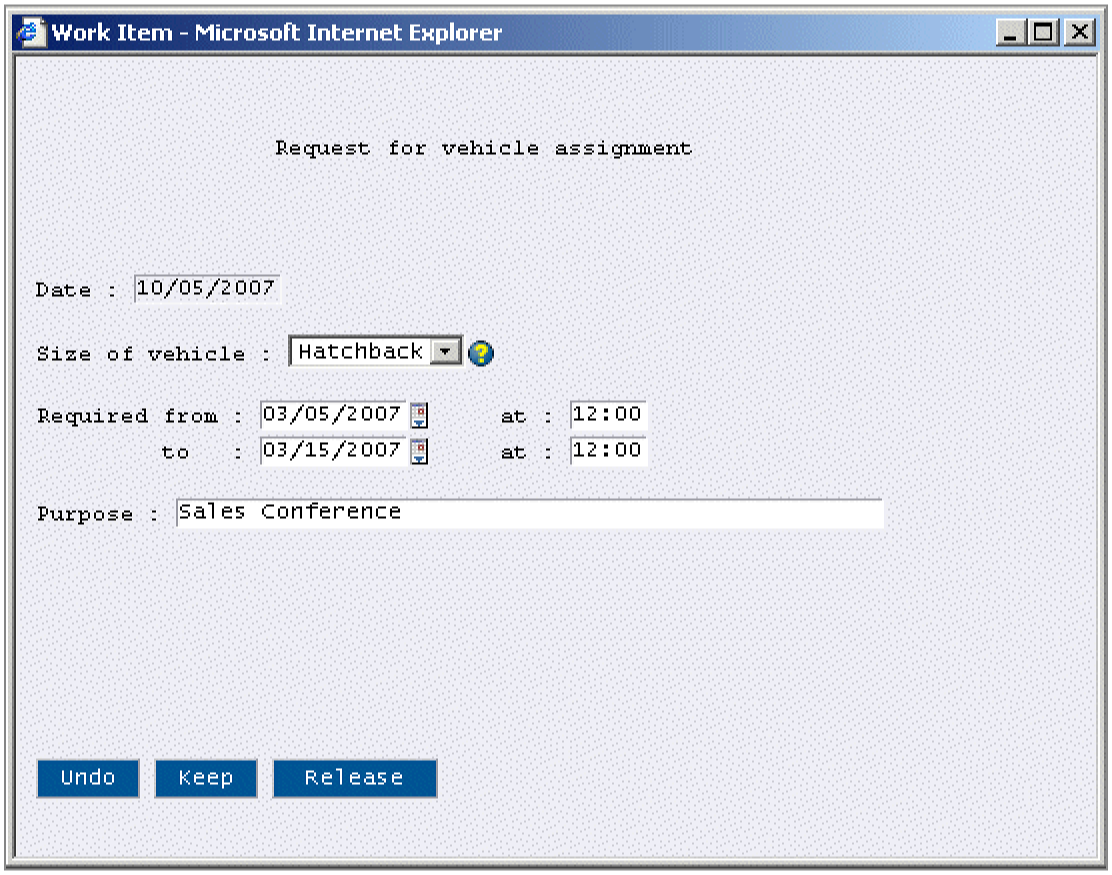
```
* Event handler for openWorkItem
*/
wccClass.prototype.openWorkItemEventHandler = function(objEvent) {
   com.tibco.bpm.ipc.log('openWorkItemEventHandler');
   var workQTag = 'myserver|swadmin|R';
   var wiTag =
'myserver|ALLOCATE|swadmin|R|8334|432045|myserver|STEP1|0|2';
   var parentPane = this.getAppBlock().getDescendantOfName
('parentPane');
   try {
```

```
   com.tibco.bpm.wcc.Tools.openWorkItem(this.getApp(), workQTag, wiTag,
parentPane);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

openWorkItemEx

This static method opens (and locks) the specified work item and displays the form associated with that work item. Unlike the **openWorkItem** method (see openWorkItem), however, you do not pass in a work item tag with this method — instead, the work item tag is obtained from a list of outstanding work items using the information passed in the parameters.

An example work item form is shown below:



Request for vehicle assignment

Date : 10/05/2007

Size of vehicle : Hatchback

Required from : 03/05/2007 at : 12:00

to : 03/15/2007 at : 12:00

Purpose : Sales Conference

This method is equivalent to selecting **Open Selected Work Item(s)** from the **Tools** menu on the work item list in the iProcess Workspace (Browser) client application.

The context in which the form is displayed depends on how the form was created, as follows:

- iProcess Modeler Forms - These forms are always displayed in a new browser window (as in the example shown above).
- TIBCO® General Interface Builder Forms - These forms (also known as “GI forms”) are always displayed in a separate dialog.
- TIBCO Forms - These forms are always displayed in a separate dialog.

Also see the **openWorkItemViaURL** application for an example of the usage of this method — see the *openWorkItemViaURL* section in the *TIBCO iProcess Workspace (Browser) Components Concepts* guide.

Syntax

```
com.tibco.bpm.wcc.Tools.openWorkItemEx(nameSpaceRef,
                                       caseNumber,
                                       procName
                                       stepName
                                       queuName
                                       queueReleased
                                       [parent]);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

caseNumber - (String) Identifies the case in which the work item was created.

procName - (String) Identifies the procedure.

stepName - (String) Identifies the step in the procedure that corresponds to the work item.

queueName - (String) Identifies the work queue in which the work item resides.

queueReleased - (String) "Y" or "N" indicating whether or not the work queue is released.

parent - (optional) This identifies a TIBCO® General Interface component (object) in which the work item form can be placed. This component must implement the TIBCO® General

Interface **setChild()** method; if it doesn't, an exception is thrown. If this parameter is not defined, external forms (ASP Forms, JSP Forms, BusinessWorks FormBuilder Forms, and iProcess Modeler Forms) always load in a separate browser window, whereas GI Forms (custom TIBCO® General Interface Forms and TIBCO Forms) load in a dialog or browser window, based on settings in `config.xml`. A custom GI Form will load in a browser or dialog based on the value of the **floatWorkItems** attribute in the form definition (<**Form**> element in `config.xml`); if it's not specified in the form definition, the value of the **floatWorkItems** attribute in the <**options**> record is used. For TIBCO Forms, the value of the **floatWorkItems** attribute in the <**options**> record is used.

Example

```
* Event handler for openWorkItemEx
*/
wccClass.prototype.openWorkItemExEventHandler = function(objEvent) {
   com.tibco.bpm.ipc.log('openWorkItemExEventHandler');
   var caseNumber = '8378';
   var procName = 'ALLOCATE';
   var stepName = 'STEP1';
   var queueName = 'swadmin';
   var queueReleased = 'Y';
   var parentPane = this.getAppBlock().getDescendantOfName
('parentPane');
   try {
   com.tibco.bpm.wcc.Tools.openWorkItemEx(this.getApp(), caseNumber,
procName,
     stepName, queueName, queueReleased, parentPane);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

unlockWorkItem

This static method unlocks the specified work item.

This method is equivalent to selecting **Unlock Work Item(s)** from the **Tools** menu on the work item list in the iProcess Workspace (Browser) client application.

Note that work items are automatically unlocked when you keep or release them; normally, you do not need to explicitly unlock work items. This function is for those rare occasions when a work item was left open for some reason (e.g., a system crash).

Unlocking a work item using this method causes any changes that were made on the form while the work item was open to be discarded.

Any user can unlock a work item that they have opened. To unlock a work item that was opened by another user, you must have system administrator authority.

If you attempt to unlock a work item that is not locked, the method call returns silently with no error.

An optional confirmation message can be displayed.

Syntax

```
com.tibco.bpm.wcc.Tools.unlockWorkItem(namespaceRef,
                                       workQTag,
                                       workItemTag,
                                       suppressConfirm);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

workQTag - (String) Identifies the work queue in which the work item resides. For more information about tags, see Introduction.

workItemTag - (String) Identifies the work item to unlock. For more information about tags, see Introduction.

suppressConfirm - (Boolean - Optional) Specifies whether or not to suppress the confirmation message. False (default) causes a confirmation message to be displayed; True suppresses the confirmation message.

Example

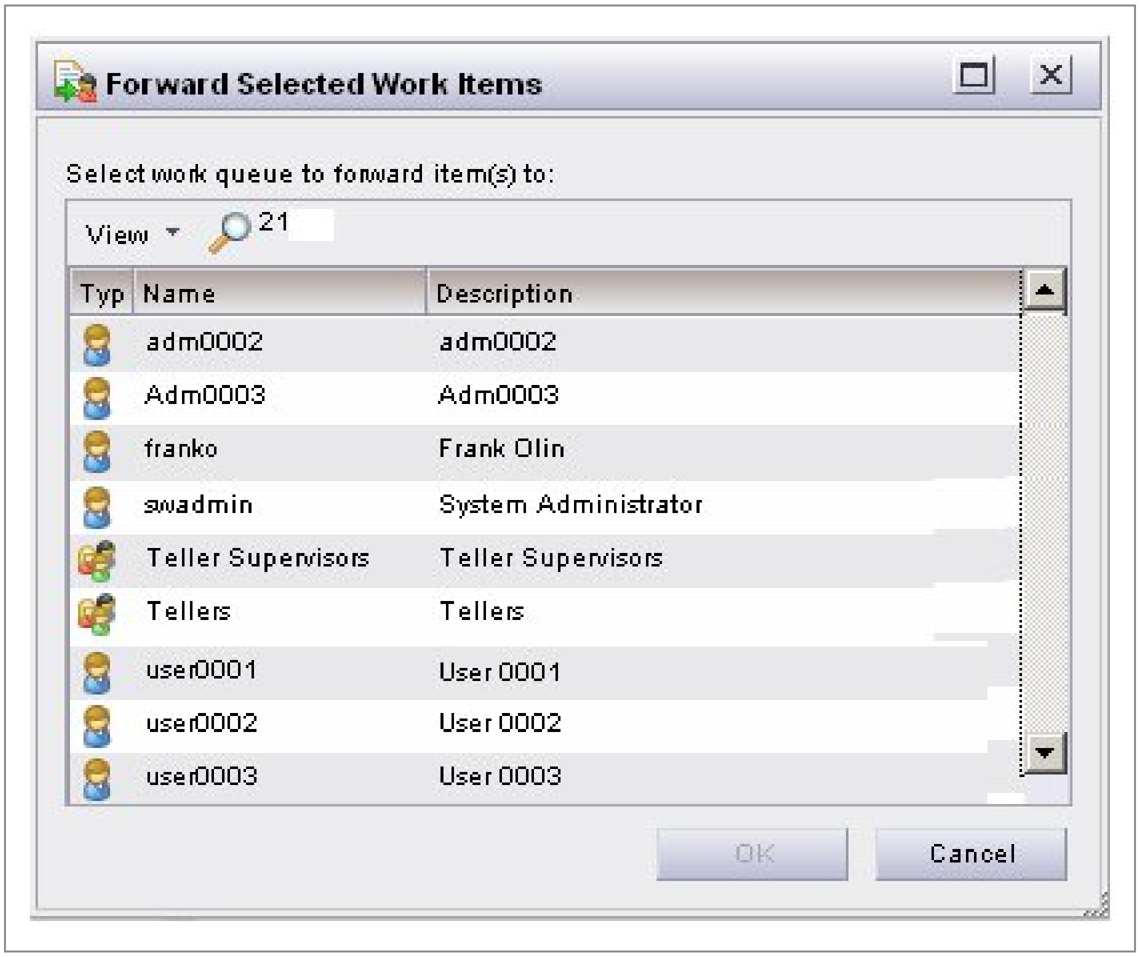
```
* Event handler for unlockWorkItem
*/
wccClass.prototype.unlockWorkItemEventHandler = function(objEvent) {
    com.tibco.bpm.ipc.log('unlockWorkItemEventHandler');
    var workQTag = 'myserver|swadmin|R';
    var wiTag =
 'myserver|CARPOOL|swadmin|R|2564|413290|myserver|REQUEST|1|0';
    try {
    com.tibco.bpm.wcc.Tools.unlockWorkItem(this.getApp(), workQTag,
      wiTag);
    }
    catch (e) {
    alert('Exception issued - ' + e);
    }
 };
```

forwardWorkItem

This static method forwards the specified work item to a different work queue.

Note that not all work items are “forwardable.” When a procedure is defined, the designer specifies whether or not work items representing each step are forwardable. (There is a “Forwardable” column available on the work item list that indicates whether or not a work item is forwardable.)

This method causes a dialog similar to the following to be displayed:



Select work queue to forward item(s) to:

Typ	Name	Description
	adm0002	adm0002
	Adm0003	Adm0003
	franko	Frank Olin
	swadmin	System Administrator
	Teller Supervisors	Teller Supervisors
	Tellers	Tellers
	user0001	User 0001
	user0002	User 0002
	user0003	User 0003

This dialog will list either all work queues on the system, or only the work queues of which the user is a member, depending on how the user's user access profile is set up.

The user selects the desired work queue from the list, then clicks **OK**. The work item specified in the method call is forwarded.

This method is equivalent to selecting **Forward Work Item(s)** from the **Tools** menu on the work item list in the iProcess Workspace (Browser) client application.

Syntax

```
com.tibco.bpm.wcc.Tools.forwardWorkItem(namespaceRef,
                                        workQTag,
                                        workItemTag);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

workQTag - (String) Identifies the work queue in which the work item being forwarded currently resides. For more information about tags, see Introduction.

workItemTag - (String) Identifies the work item to forward. For more information about tags, see Introduction.

Example

```
* Event handler for forwardWorkItem
*/
wccClass.prototype.forwardWorkItemEventHandler = function(objEvent) {
   com.tibco.bpm.ipc.log('forwardWorkItemEventHandler');
   var workQTag = 'myserver|swadmin|R';
   var wiTag =
'myserver|CARPOOL|broker|R|1852|435042|myserver|CARREADY|1|0';
   try {
   com.tibco.bpm.wcc.Tools.forwardWorkItem(this.getApp(), workQTag,
wiTag);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

releaseWorkItem

This static method releases the specified work item.

Note that you can only release work items with this method that are considered “directly releasable”, i.e., *they do not have any input fields on their form* (if they have a form). That does not mean their input fields have been filled in — they cannot have input fields. (There is a **Releasable** column available on the work item list that indicates whether or not a work item is directly releasable.)

Releasing the work item causes the case to advance to the next step in the procedure, possibly resulting in another work item appearing in someone’s work queue.

This method is equivalent to selecting **Release Work Item(s)** from the **Tools** menu on the work item list in the iProcess Workspace (Browser) client application.

Syntax

```
com.tibco.bpm.wcc.Tools.releaseWorkItem(namespaceRef,
                                        workQTag,
                                        workItemTag);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

workQTag - (String) Identifies the work queue in which the work item being released currently resides. For more information about tags, see Introduction.

workItemTag - (String) Identifies the work item to release. For more information about tags, see Introduction.

Example

```
* Event handler for releaseWorkItem
*/
wccClass.prototype.releaseWorkItemEventHandler = function(objEvent) {
   com.tibco.bpm.ipc.log('releaseWorkItemEventHandler');
   var workQTag = 'myserver|swadmin|R';
   var wiTag =
'myserver|CARPOOL|swadmin|R|9728|435052|myserver|SERVICE|1|2';
   try {
   com.tibco.bpm.wcc.Tools.releaseWorkItem(this.getApp(),
     workQTag, wiTag);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

configureSupervisors

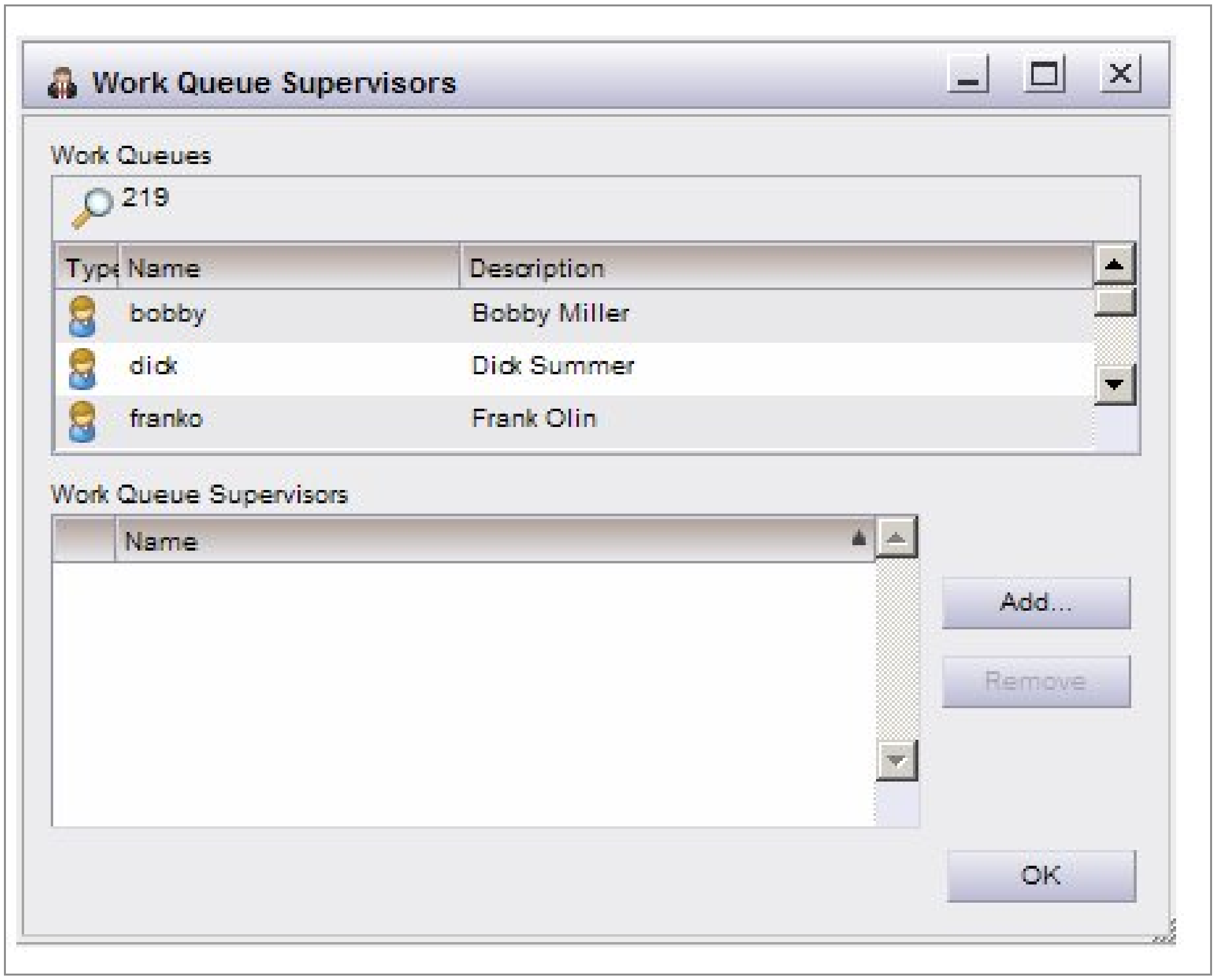
This static method is used to designate users as work queue supervisors. A user must be a work queue supervisor to perform the following tasks:

- Configure **participation schedules**. A participation schedule gives another user temporary access to a work queue. For more information about configuring participation schedules, see configureParticipation.
- Configure **redirection schedules**. A redirection schedule causes work items to be temporarily redirected to another work queue. For more information about configuring redirection schedules, see configureRedirection.

Each work queue can be assigned one or more work queue supervisors.

You must have system administrator authority to configure work queue supervisors.

This method causes a dialog similar to the following to be displayed:



Type	Name	Description
	bobby	Bobby Miller
	dick	Dick Summer
	franko	Frank Olin

The **Work Queues** section of this dialog lists all work queues (i.e., all users and groups) defined on your TIBCO system. The **WorkQueue Supervisors** section lists the supervisors for the currently selected work queue.

For more details about configuring work queue supervisors, see the *TIBCO iProcess Workspace (Browser) User Guide*.

This method is equivalent to selecting **Manage Work Queue Supervisors** from the **Tools** menu on the work queue list.

Syntax

```
com.tibco.bpm.wcc.Tools.configureSupervisors(namespaceRef);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

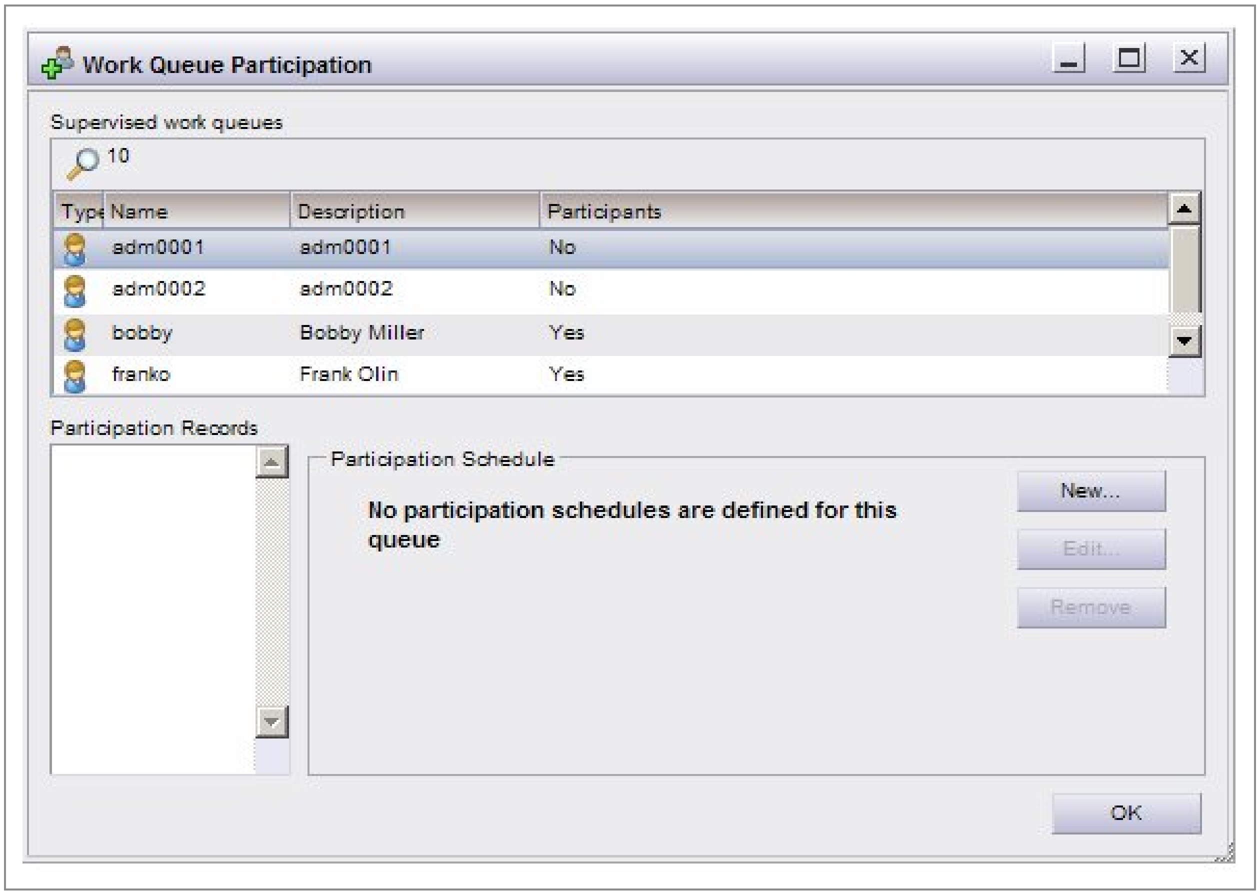
Example

```
* Event handler for configureSupervisors
*/
wccClass.prototype.configureSupervisorsEventHandler = function(objEvent)
{
   com.tibco.bpm.ipc.log('configureSupervisorsEventHandler');
   try {
   com.tibco.bpm.wcc.Tools.configureSupervisors(this.getApp());
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

configureParticipation

This static method is used to configure *participation schedules,* which specify that a user can participate in (i.e., have access to) another user’s work queue for a specified period of time.

This method causes a dialog similar to the following to be displayed:



Type	Name	Description	Participants
	adm0001	adm0001	No
	adm0002	adm0002	No
	bobby	Bobby Miller	Yes
	franko	Frank Olin	Yes

No participation schedules are defined for this queue

The **Supervised work queues** section lists all work queues for which the user has been designated a supervisor — these are the work queues for which the user is authorized to configure participation schedules. (For more information about designating a user a work queue supervisor, see configureSupervisors.)

For details about work queue participation, see the *TIBCO iProcess Workspace (Browser) User Guide*.

This method is equivalent to selecting **Manage Work Queue Participation** from the **Tools** menu on the work queue list.

Syntax

```
com.tibco.bpm.wcc.Tools.configureParticipation(namespaceRef);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

Example

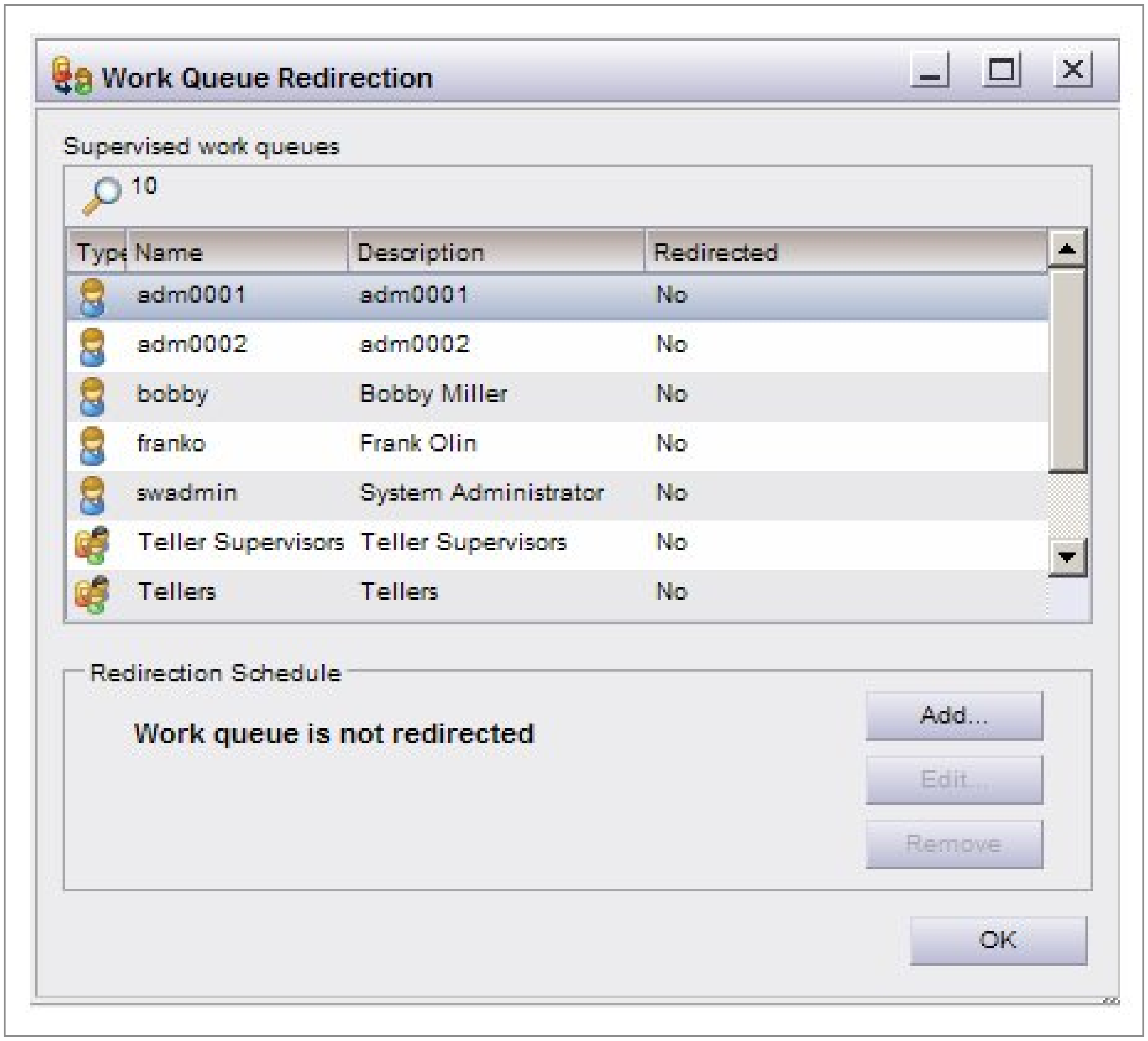
```
* Event handler for configureParticipation
*/
wccClass.prototype.configureParticipationEventHandler = function
(objEvent) {
   com.tibco.bpm.ipc.log('configureParticipationEventHandler');
   try {
   com.tibco.bpm.wcc.Tools.configureParticipation(this.getApp());
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

configureRedirection

This static method is used to configure *redirection schedules,* which are used to *redirect* one user's work items to the work queue of another user or group for a specified period of time.

Note: For more information about forwarding an individual work item from a work queue, see forwardWorkItem.

This method causes a dialog similar to the following to be displayed:



Type	Name	Description	Redirected
	adm0001	adm0001	No
	adm0002	adm0002	No
	bobby	Bobby Miller	No
	franko	Frank Olin	No
	swadmin	System Administrator	No
	Teller Supervisors	Teller Supervisors	No
	Tellers	Tellers	No

Redirection Schedule

Work queue is not redirected

The **Supervised work queues** section lists all work queues for which the user has been designated a supervisor — these are the work queues for which the user is authorized to configure redirection schedules. (For more information about designating a user a work queue supervisor, see configureSupervisors.)

For details about configuring redirection schedules, see the *TIBCO iProcess Workspace (Browser) User Guide*.

This method is equivalent to selecting **Manage Work Queue Redirection** from the **Tools** menu on the work queue list.

Syntax

```
com.tibco.bpm.wcc.Tools.configureRedirection(namespaceRef);
```

Parameters

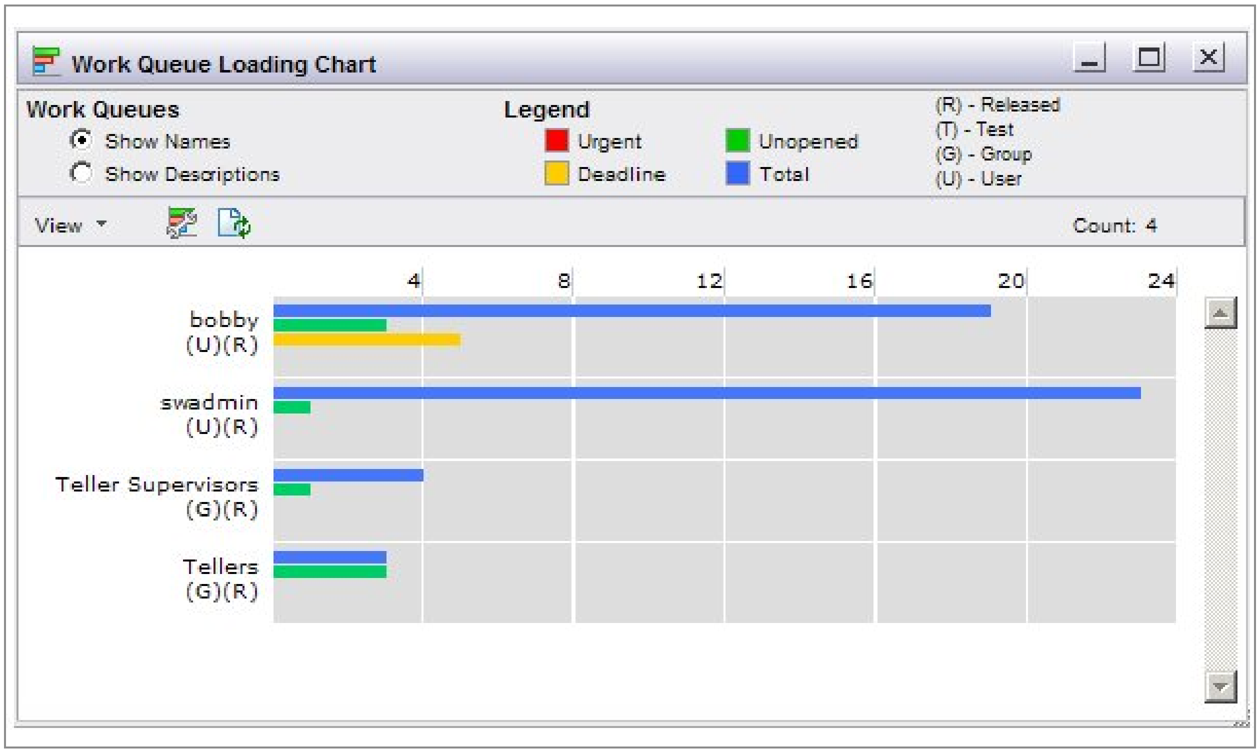
nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

Example

```
* Event handler for configureRedirection
*/
wccClass.prototype.configureRedirectionEventHandler = function(objEvent)
{
   com.tibco.bpm.ipc.log('configureRedirectionEventHandler');
   try {
   com.tibco.bpm.wcc.Tools.configureRedirection(this.getApp());
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

viewWorkQLoadingChart

This static method displays a graphical summary of the work queues available to the user. For example:



This chart provides information about the numbers and types of work items in each work queue.

This method is equivalent to selecting **Work Queue Loading Chart** from the **Tools** menu on the work queue list.

Syntax

```
com.tibco.bpm.wcc.Tools.viewWorkQLoadingChart(namespaceRef,
                                              releasedQs,
                                              testQs,
                                              groupQs,
                                              userQs);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

releasedQs - (Boolean - Optional) Include released work queues in loading chart? Default = True.

testQs - (Boolean - Optional) Include test work queues in loading chart?
Default = True.

Note: You must pass True for either *releasedQs* or *testQs*. If you pass False for both parameters, loading information is displayed for both released and test work queues.

groupQs - (Boolean - Optional) Include group work queues in loading chart? Default = True.

userQs - (Boolean - Optional) Include user work queues in loading chart?
Default = True.

Note: You must pass True for either *groupQs* or *userQs*. If you pass False for both parameters, loading information is displayed for both group and user work queues.

Example

```
* Event handler for viewWorkQLoadingChart
*/
wccClass.prototype.viewWorkQLoadingChartEventHandler = function
(objEvent) {
   com.tibco.bpm.ipc.log('viewWorkQLoadingChartEventHandler');
   try {
   com.tibco.bpm.wcc.Tools.viewWorkQLoadingChart(this.getApp
(),true,false,true,true);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

getStartProcs

This method returns an array of JavaScript objects that represent iProcess procedures for which the logged-in user has permission to start cases.

Each JavaScript object returned represents a procedure, and has a Name, Description, HostingNode, MajorVersion, MinorVersion, and Tag property.

This method can be used prior to calling the startCase method to determine the procedures the logged-in user can start.

Syntax

```
com.tibco.bpm.wcc.Tools.getStartProcs(namespaceRef);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

Example

```
var startProcs = com.tibco.bpm.wcc.Tools.getStartProcs(this.getApp());
var msg = 'Name/Description/HostingNode/MajorVersion/MinorVersion/Tag' +
'\n';
for (var i = 0; i < startProcs.length; i++) {
    msg += startProcs[i].Name + '/' +
       startProcs[i].Description + '/' +
       startProcs[i].HostingNode + '/' +
       startProcs[i].MajorVersion + '/' +
       startProcs[i].MinorVersion + '/' +
       startProcs[i].Tag + '"\n';
}
alert(msg);
```

getAuditProcs

This method returns an array of JavaScript objects that represent iProcess procedures for which the logged-in user has permission to view or add entries to case history.

Each JavaScript object returned represents a procedure, and has a Name, Description, HostingNode, MajorVersion, MinorVersion, and Tag property.

This method can be used prior to calling the viewGraphicalCaseHistory or addCaseHistoryEntry method to determine if the logged-in user has permission to view the graphical case history or add an entry to case history.

Syntax

```
com.tibco.bpm.wcc.Tools.getAuditProcs(namespaceRef);
```

Parameters

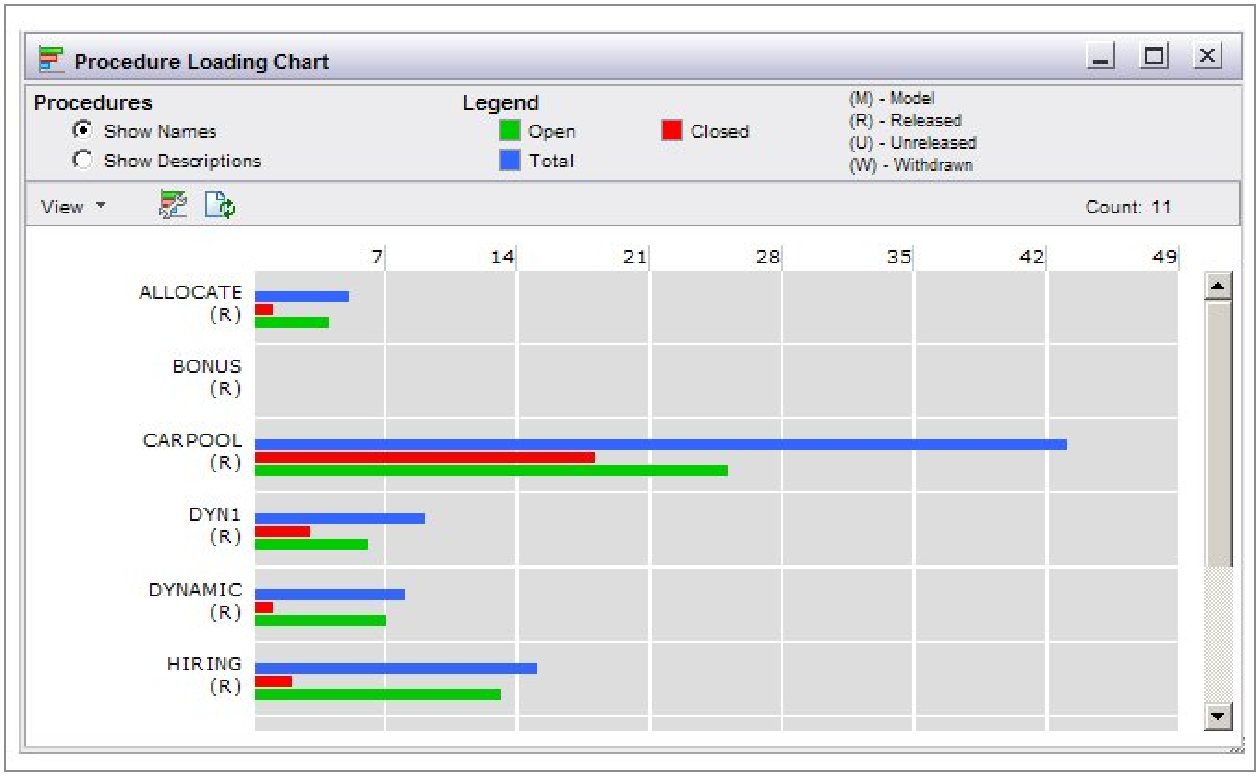
nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

Example

```
var auditProcs = com.tibco.bpm.wcc.Tools.getAuditProcs(this.getApp());
var msg = 'Name/Description/HostingNode/MajorVersion/MinorVersion/Tag' +
'\n';
for (var i = 0; i < auditProcs.length; i++) {
    msg += auditProcs[i].Name + '/' +
       auditProcs[i].Description + '/' +
       auditProcs[i].HostingNode + '/' +
       auditProcs[i].MajorVersion + '/' +
       auditProcs[i].MinorVersion + '/' +
       auditProcs[i].Tag + '"\n';
}
alert(msg);
```

viewProcLoadingChart

This static method displays a graphical summary of the procedures on the system. For example:



This chart provides information about the numbers and types of cases that exist for each procedure.

This method is equivalent to selecting **Procedure Loading Chart** from the **Tools** menu on the procedure list.

Syntax

```
com.tibco.bpm.wcc.Tools.viewProcLoadingChart(namespaceRef,
                                              released,
                                              unreleased,
                                              model,
                                              withdrawn);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

released - (Boolean - Optional) Include released procedures in loading chart? Default = True.

unreleased - (Boolean - Optional) Include unreleased procedures in loading chart? Default = True.

model - (Boolean - Optional) Include model procedures in loading chart? Default = True.

withdrawn - (Boolean - Optional) Include withdrawn procedures in loading chart? Default = True.

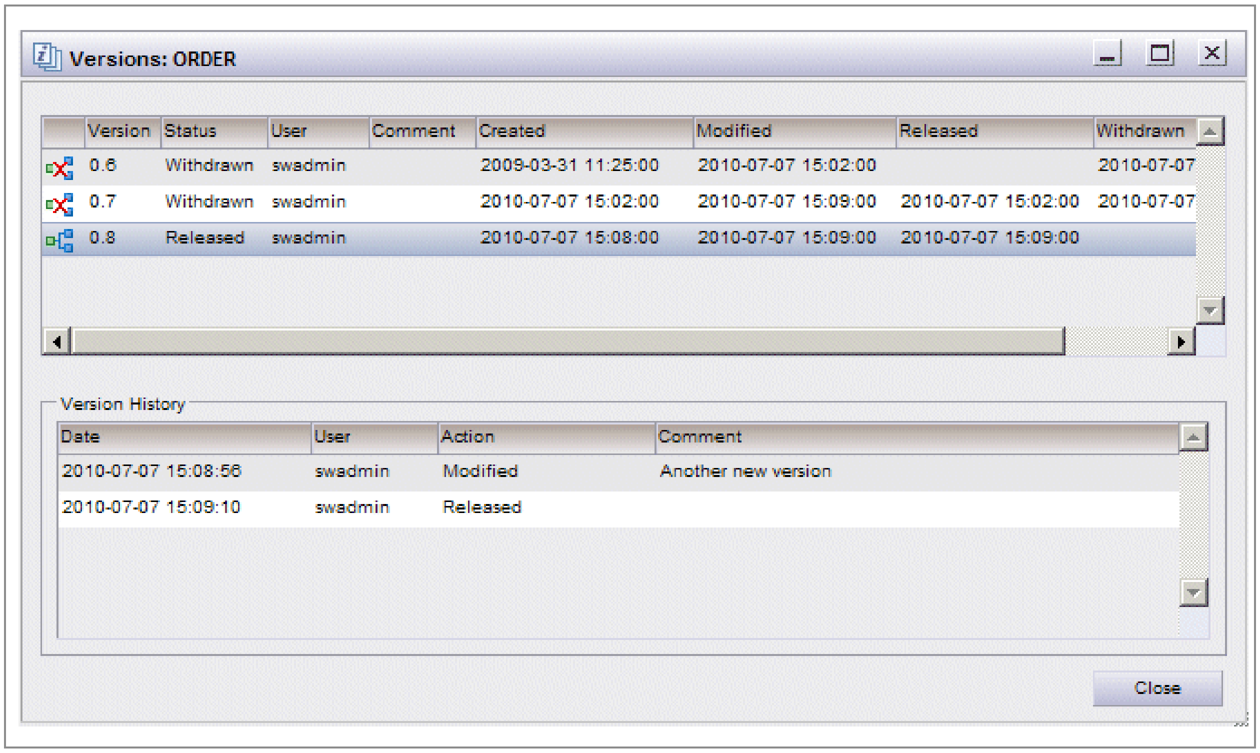
Example

```
* Event handler for viewProcedureLoadingChart
*/
wccClass.prototype.viewProcedureLoadingChartEventHandler = function
(objEvent) {
   com.tibco.bpm.ipc.log('viewProcedureLoadingChartEventHandler');
   try {
   com.tibco.bpm.wcc.Tools.viewProcedureLoadingChart(this.getApp
(),true,false,false,
      false);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

loadProcVersion

This static method displays information about the past and current versions of the specified procedure.

This method displays a dialog similar to the following:



Version	Status	User	Comment	Created	Modified	Released	Withdrawn
0.6	Withdrawn	swadmin		2009-03-31 11:25:00	2010-07-07 15:02:00		2010-07-07
0.7	Withdrawn	swadmin		2010-07-07 15:02:00	2010-07-07 15:09:00	2010-07-07 15:02:00	2010-07-07
0.8	Released	swadmin		2010-07-07 15:08:00	2010-07-07 15:09:00	2010-07-07 15:09:00	

Version History

Date	User	Action	Comment
2010-07-07 15:08:56	swadmin	Modified	Another new version
2010-07-07 15:09:10	swadmin	Released	

Clicking on one of the versions in the top section causes history information about that version to be displayed in the section on the bottom of the dialog.

This method is equivalent to selecting **Versions** from the **Tools** menu on the procedure list.

Syntax

```
com.tibco.bpm.wcc.Tools.loadProcVersion(namespaceRef,
                                         procTag);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

procTag - (String) Identifies the procedure whose version information you want displayed. For more information about tags, see Introduction.

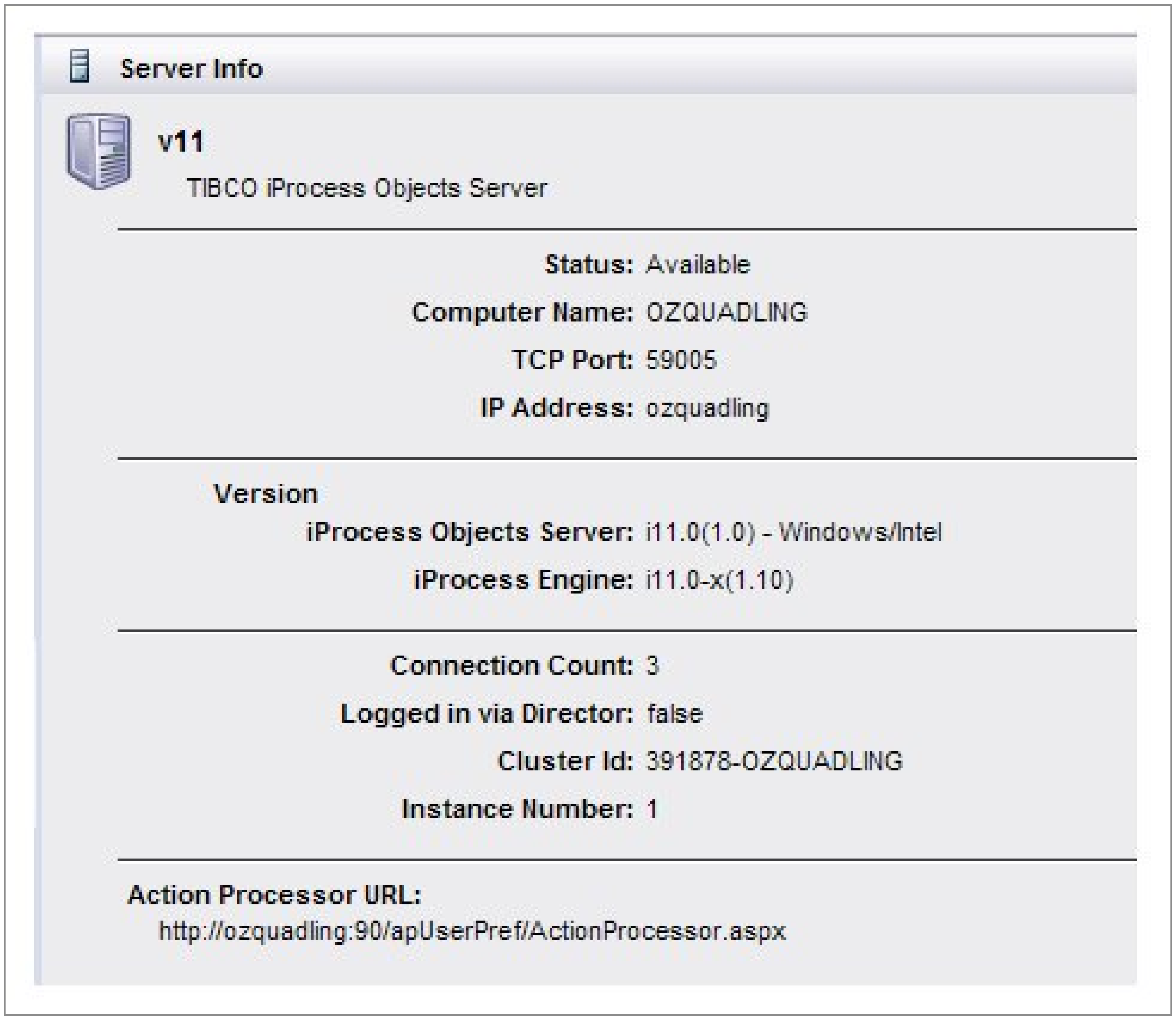
Example

```
* Event handler for loadProcVersion
*/
wccClass.prototype.loadProcVersionEventHandler = function(objEvent) {
   com.tibco.bpm.ipc.log('loadProcVersionEventHandler');
   try {
   com.tibco.bpm.wcc.Tools.loadProcVersion(this.getApp(),true,false,
     false,false);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

loadServerInfo

This static method displays technical information about the iProcess Objects Server the user is currently logged into.

This method causes a dialog similar to the following to be displayed:



Server Info

v11

TIBCO iProcess Objects Server

Status: Available

Computer Name: OZQUADLING

TCP Port: 59005

IP Address: ozquadling

Version

iProcess Objects Server: i11.0(1.0) - Windows/Intel

iProcess Engine: i11.0-x(1.10)

Connection Count: 3

Logged in via Director: false

Cluster Id: 391878-OZQUADLING

Instance Number: 1

Action Processor URL:

http://ozquadling:90/apUserPref/ActionProcessor.aspx

This method is equivalent to clicking on the **Server Info** button in the iProcess Workspace (Browser) client application.

Syntax

```
com.tibco.bpm.wcc.Tools.loadServerInfo(nameSpaceRef,
                                       parent);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

parent - This identifies a TIBCO® General Interface component (object) in which the **Server Info** dialog can be placed. This component must implement the TIBCO® General Interface **setChild()** method; if it doesn’t, an exception is thrown.

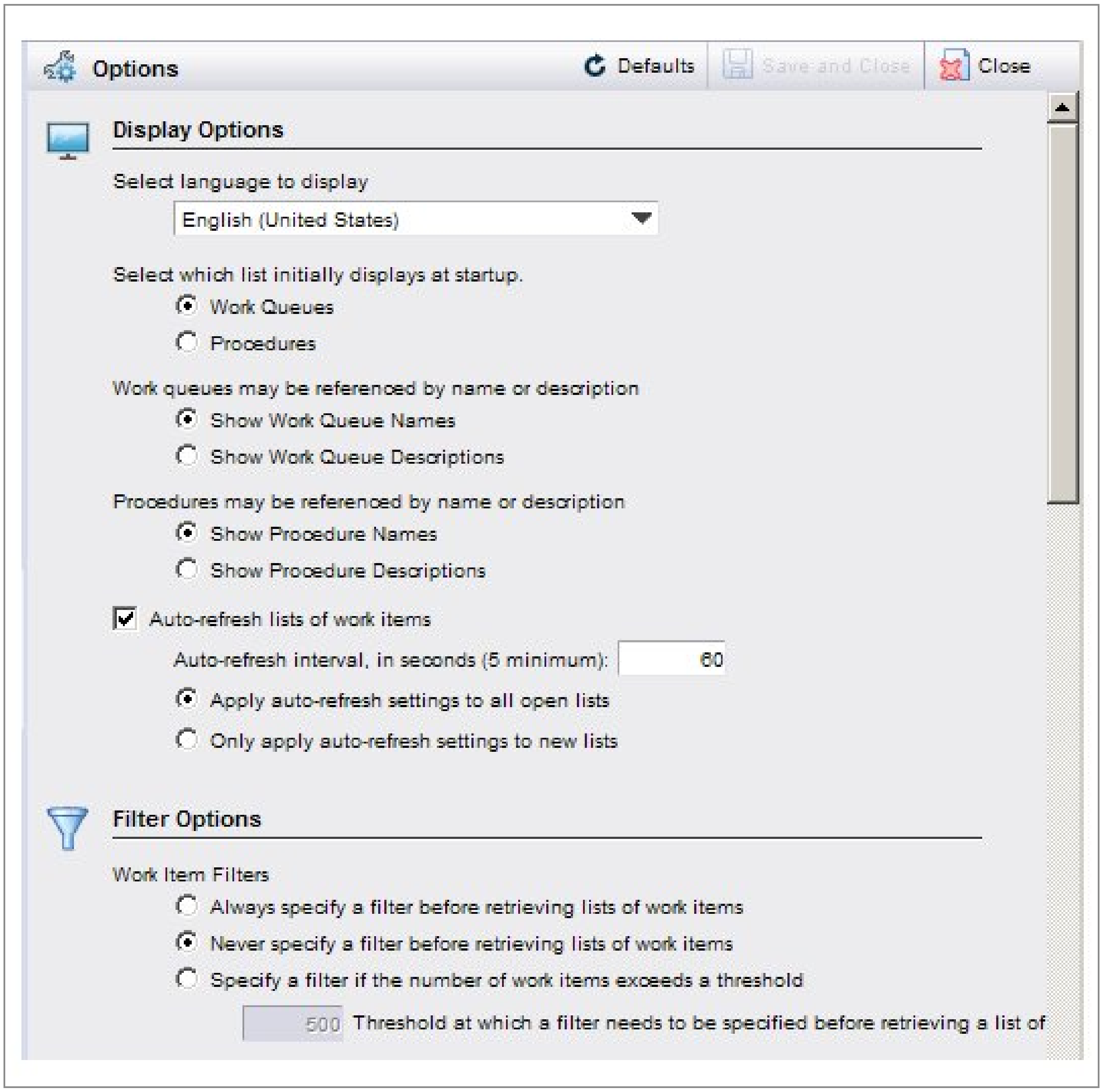
Example

```
* Event handler for loadServerInfo
        */
wccClass.prototype.loadServerInfoEventHandler = function(objEvent) {
   com.tibco.bpm.ipc.log('loadServerInfoEventHandler');
   var parentPane = this.getAppBlock().getDescendantOfName
('parentPane');
   try {
   com.tibco.bpm.wcc.Tools.loadServerInfo(this.getApp(),
     parentPane);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

workspaceOptions

This static method opens the **Options** dialog, which is used to establish default application settings for the user. These include things such as whether preview is turned on by default, the size/location of work item forms, etc.

The **Options** dialog appears as follows:



Display Options

Select language to display: English (United States)

Select which list initially displays at startup.
- [✓ Work Queues]
- Procedures

Work queues may be referenced by name or description
- [✓ Show Work Queue Names]
- Show Work Queue Descriptions

Procedures may be referenced by name or description
- [✓ Show Procedure Names]
- Show Procedure Descriptions

[✓ Auto-refresh lists of work items]

Auto-refresh interval, in seconds (5 minimum): 60
- [✓ Apply auto-refresh settings to all open lists]
- Only apply auto-refresh settings to new lists

Filter Options

Work Item Filters
- Always specify a filter before retrieving lists of work items
- [✓ Never specify a filter before retrieving lists of work items]
- Specify a filter if the number of work items exceeds a threshold

500 Threshold at which a filter needs to be specified before retrieving a list of

This method is equivalent to clicking on the **Options** button in the iProcess Workspace (Browser) client application.

For details about all of the options available, see the *TIBCO iProcess Workspace (Browser) User Guide*.

Note: The only session activity option available with the **workspaceOptions** method is the change password function. The Session Activity Log is not available using this method.

Syntax

```
com.tibco.bpm.wcc.Tools.workspaceOptions(nameSpaceRef,
                                         parent);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

parent - This identifies a TIBCO® General Interface component (object) in which the **Options** dialog can be placed. This component must implement the TIBCO® General Interface **setChild()** method; if it doesn't, an exception is thrown.

Example

```
* Event handler for workspaceOptions
*/
wccClass.prototype.workspaceOptionsEventHandler = function(objEvent) {
   com.tibco.bpm.ipc.log('workspaceOptionsEventHandler');
   var parentPane = this.getAppBlock().getDescendantOfName
('parentPane');
   try {
   com.tibco.bpm.wcc.Tools.workspaceOptions(this.getApp(),
     parentPane);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

workItemTag2CaseTag

This static method returns a case tag extracted from the work item tag passed as an argument.

If the argument is not a valid work item tag (which includes not containing the correct number of elements between the vertical bars), an exception is thrown.

Syntax

```
com.tibco.bpm.wcc.Tools.workItemTag2CaseTag(workItemTag);
```

Parameters

workItemTag - A valid work item tag. For more information about tags, see Introduction.

Example

```
/**
* Event handler for workItemTag2CaseTag
*/
wccClass.prototype.workItemTag2CaseTagEventHandler = function(objEvent)
{
   com.tibco.bpm.ipc.log('workItemTag2CaseTagEventHandler');
   try {
   var wiTag =
'i2tagtest|CARPOOL|swadmin|R|9728|434946|i2tagtest|CARREADY|1|2';
   com.tibco.bpm.wcc.Tools.workItemTag2CaseTag(wiTag);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

workItemTag2WorkQTag

This static method returns a work queue tag extracted from the work item tag passed as an argument.

If the argument is not a valid work item tag (which includes not containing the correct number of elements between the vertical bars), an exception is thrown.

Syntax

```
com.tibco.bpm.wcc.Tools.workItemTag2WorkQTag(nameSpaceRef,
                                             workItemTag);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

workItemTag - A valid work item tag. For more information about tags, see Introduction.

Example

```
/**
* Event handler for workItemTag2WorkQTag
*/
wccClass.prototype.workItemTag2WorkQTagEventHandler = function(objEvent)
{
   com.tibco.bpm.ipc.log('workItemTag2WorkQTagEventHandler');
   try {
   var wiTag =
'i2tagtest|CARPOOL|swadmin|R|9728|434946|i2tagtest|CARREADY|1|2';
   com.tibco.bpm.wcc.Tools.workItemTag2WorkQTag(this.getApp(), wiTag);
   }
   catch (e) {
   alert('Exception issued - ' + e);
   }
};
```

getUserAttributes

This static method returns an array of objects that represent iProcess attributes assigned to a specific user.

Syntax

```
com.tibco.bpm.wcc.Tools.getUserAttributes(namespaceRef,
                                          userName);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

userName - (String) Name of user for which attributes are to be retrieved.

Example

```
var attributes = com.tibco.bpm.wcc.Tools.getUserAttributes(this.getApp
().getApp(), 'swadmin');
for (var i = 0; i < attributes.length; i++) {
   var name = attributes[i].Name;
   var type = attributes[i].Type;
   var value = attributes[i].Value;
};
```

getGroupAttributes

This static method returns an array of objects that represent iProcess attributes assigned to a specific group.

Syntax

```
com.tibco.bpm.wcc.Tools.getGroupAttributes(namespaceRef, groupName);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

groupName - (String) Name of group for which attributes are to be retrieved.

Example

```
var attributes = com.tibco.bpm.wcc.Tools.getGroupAttributes(this.getApp
().getApp(), 'Supervisors');
for (var i = 0; i < attributes.length; i++) {
   var name = attributes[i].Name;
   var type = attributes[i].Type;
   var value = attributes[i].Value;
};
```

manageMER

This static method displays the interface for managing Monitor Event Request (MER) messages on the server. For more information about using the Manage MER interface, see Managing Monitor Event Request (MER) Messages.

Syntax

```
com.tibco.bpm.wcc.Tools.manageMER(namespaceRef);
```

Parameters

nameSpaceRef - This specifies a reference to the application. For information, see Namespace Reference.

Example

```
com.tibco.bpm.wcc.Tools.getGroupAttributes(this.getApp());
```

JavaScript Interface

This section provides a list of the available Action Processor JavaScript interfaces. These interfaces can be used by custom WCC applications to make direct requests to the Action Processor.

Overview

The TIBCO iProcess Workspace (Browser) client application makes requests to the Action Processor using a library of JavaScript interfaces. These interfaces are published and documented so that direct requests can be made to the Action Processor to perform functions such as starting cases, getting various lists, forwarding work items, etc.

Direct requests to the Action Processor using the JavaScript interface methods can be made from:

— WCC applications,

— TIBCO® General Interface forms (for more information, see the TIBCO® General Interface documentation), or

— TIBCO Forms (for more information, see the TIBCO Forms documentation).

This documentation subdivides the JavaScript interfaces into three logical groups:

- **Infrastructure classes** - These provide functions such as setting up the action request, creating a socket, performing an XSL transform, etc.

  For a list and description of the available infrastructure-type classes, see Infrastructure Classes.

- **Action Request classes** - These classes provide the actual action requests, such as getting a list of work items, forwarding a work item, setting case data, etc.

  For a list and description of the available action request classes, see Action Request Classes.

- **Data classes** - These classes provide a means to create data objects that are passed in action requests, such as filter and sort criteria, dates, field objects, etc.

  For a list and description of the available data classes, see Data Classes.

Following the class/method descriptions is an example of code needed to create an Action Processor request, create a socket call, and perform an XSL transform on the XML results returned by the Action Processor. This example can be found in Example Request.

Infrastructure Classes

These classes provide functions such as setting up the action request, creating a socket, performing an XSL transform, etc.

The classes available in this category are listed in the table below (click on the class name to link to the method descriptions for that class).

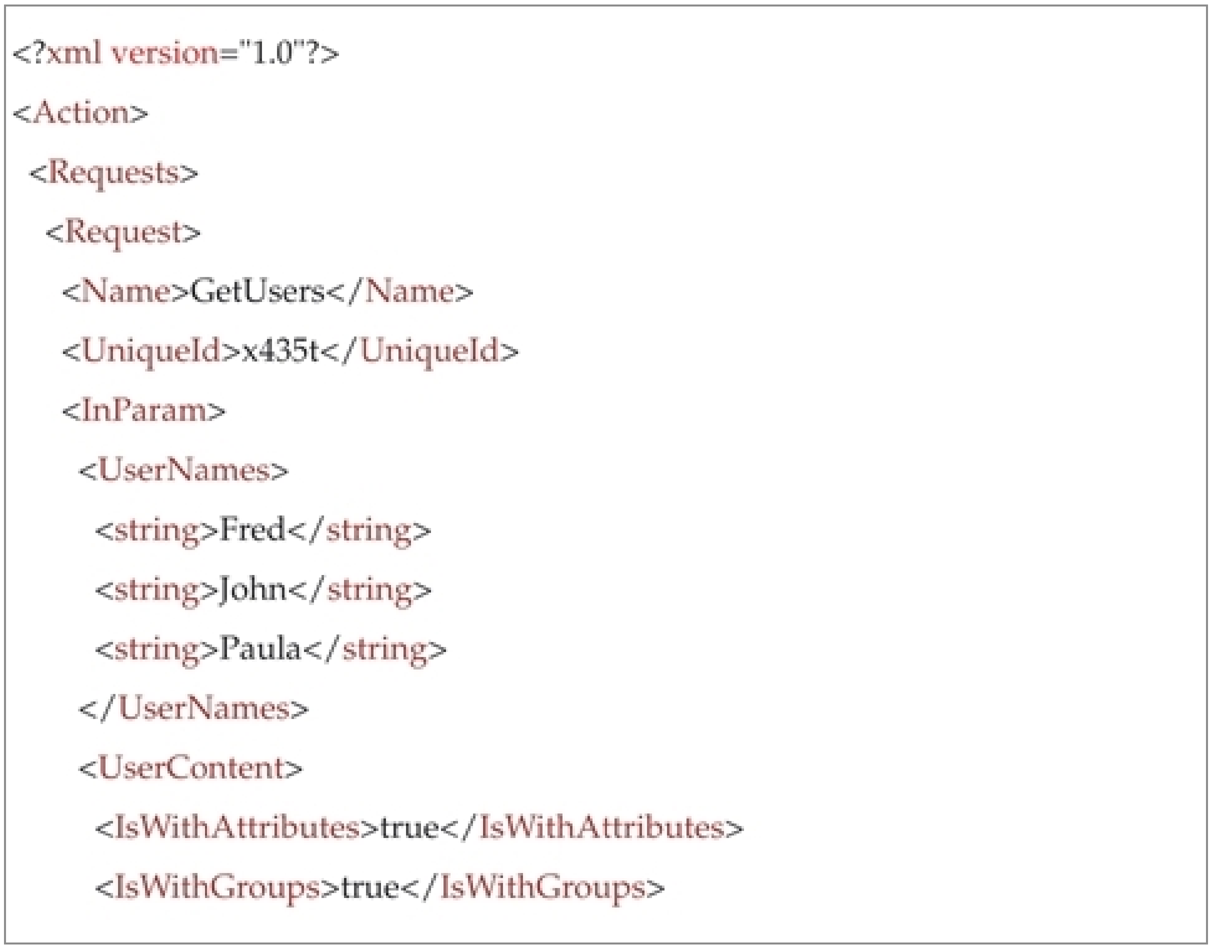
Class[1]	**Description**
Action	Provides an object model interface to support the creation of Action Processor commands.
AuthenticateRequests	Provides login and logout requests.
Socket	Provides socket access for components.
UtilityRequests	Provides general-utility Action Processor requests.
XmlElement	Provides support creating XML elements.
XslTransform	Provides support for performing XSL transforms on XML documents.

Action

The **com.tibco.bpm.ipc.Action** class defines an object model interface to support the TIBCO Action Processor protocol. The Action Processor supports XML communication using the following example protocol:

[1]All classes are prefixed with “com.tibco.bpm.ipc”.

Each of the infrastructure classes is described in the following subsections.

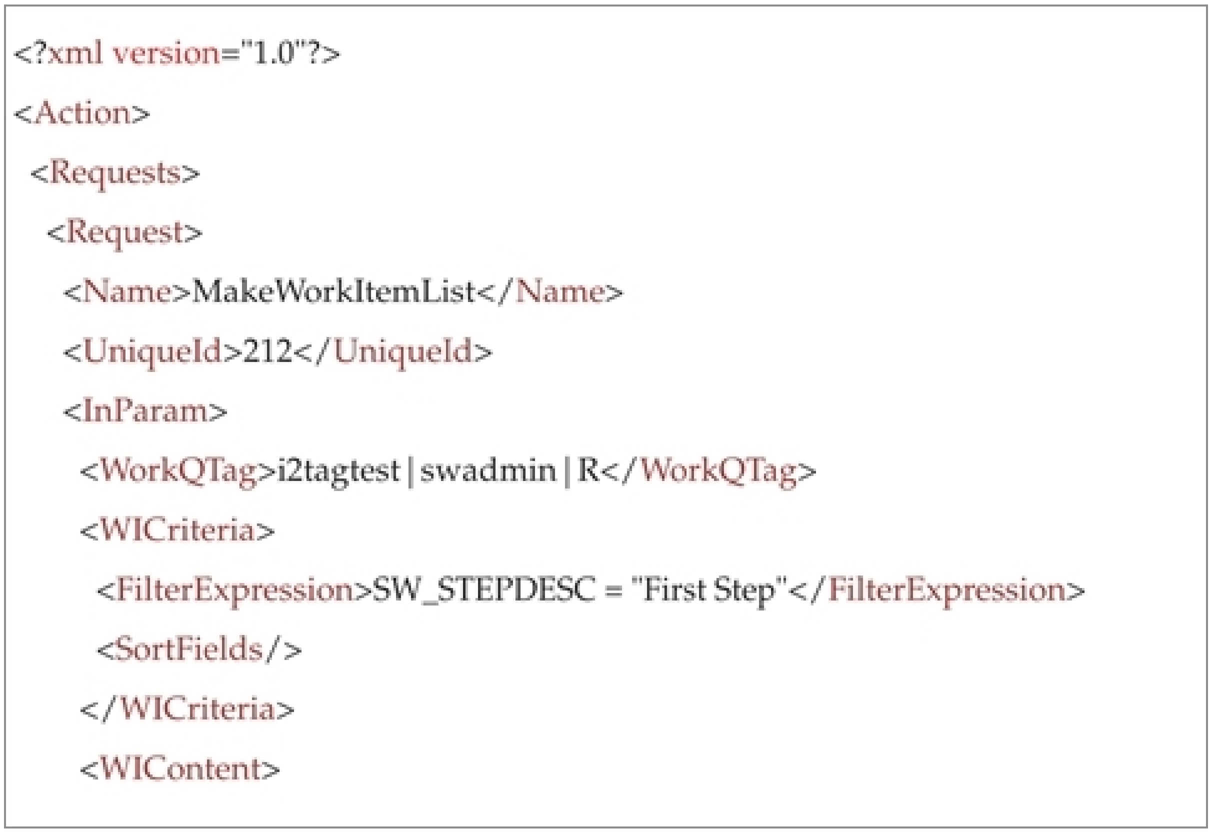


```
<?xml version="1.0"?>
<Action>
 <Requests>
  <Request>
   <Name>GetUsers</Name>
   <UniqueId>x435t</UniqueId>
   <InParam>
    <UserNames>
     <string>Fred</string>
     <string>John</string>
     <string>Paula</string>
    </UserNames>
    <UserContent>
     <IsWithAttributes>true</IsWithAttributes>
     <IsWithGroups>true</IsWithGroups>
```

The command consists of an **Action** that can support multiple requests. This class adds structure around an object model that enables developers to build these requests. The following is an example of how to use this class to construct a simple **MakeWorkItemList** request. The purpose of this request is to obtain the list state object to obtain the available count that the WICriteria filter expression defines:

```
var action = new com.tibco.bpm.ipc.Action();
var request = action.addRequest("MakeWorkItemList", "212");
request.addParameter(new com.tibco.bpm.ipc.XmlElement("WorkQTag",
"i2tagtest|swadmin|R"));
request.addParameter(new com.tibco.bpm.ipc.vWICriteria('SW_STEPDESC =
"First Step"'));
request.addParameter(new com.tibco.bpm.ipc.vWIContent());
request.addParameter(new com.tibco.bpm.ipc.XmlElement("StartIndex", 0));
request.addParameter(new com.tibco.bpm.ipc.XmlElement("ReturnCout", 0));
request.addParameter(new com.tibco.bpm.ipc.XmlElement("Keep", false));
```

Calling **action.toXml()** on the above **Action** object returns the following valid XML string:



```
<?xml version="1.0"?>
<Action>
 <Requests>
  <Request>
   <Name>MakeWorkItemList</Name>
   <UniqueId>212</UniqueId>
   <InParam>
    <WorkQTag>i2tagtest|swadmin|R</WorkQTag>
    <WICriteria>
     <FilterExpression>SW_STEPDESC = "First Step"</FilterExpression>
     <SortFields/>
    </WICriteria>
    <WIContent>
```

The **com.tibco.bpm.ipc.Action** class contains the following public methods.

Method	Description
`init(requests)`	Constructor — initializes the class. Optionally creates an Action Processor command for the specified array of request objects. **Parameters** *requests* - (optional) An array of **Request** objects (com.tibco.bpm.ipc.RequestXml). **Returns** An instance of the **com.tibco.bpm.ipc.Action** class.
`addRequest (strCommand,`	Adds a new **Request** to the **Action** command.

Method	Description
`strRequestId)`	**Parameters** *strCommand* - (String) The request command. *strRequestId* - (String) A unique ID for the request. This is the ID of the XML cache that will hold the XML document returned by the Action Processor. **Returns** Request object (**com.tibco.bpm.ipc.RequestXml**).

`addRequestParameter (strRequestId,`
 	
`objParameter)`
	Adds parameters to a Request and identifies the Request by unique ID.
	
	Parameters
	
	strRequestId - (String) Unique ID of the request.
	
	objParameter - (com.tibco.bpm.ipc.XmlElement) XML element object to add to the request.
	
	Returns
	
	None.

`toXml()`	Outputs a valid XML string for the **com.tibco.bpm.ipc.Action** command. **Returns** Action object serialized as an XML string.

AuthenticateRequests

The **com.tibco.bpm.ipc.AuthenticateRequests** class provides methods for creating server requests to log in and log out. The following examples show the execution of login and logout requests.

Login

```
var socket = this.getApp().newSocket('Login');
var nodeId = new com.tibco.bpm.ipc.vNodeId(name, computername,
ipaddress, tcpport, director);
socket.setAsync(false);
var action = new com.tibco.bpm.ipc.Action([
   com.tibco.bpm.ipc.AuthenticateRequests.login(
   'Login', new com.tibco.bpm.ipc.NodeCtx('userId', 'password',
   nodeId))]);
socket.addParameter('action', action.toXml());
socket.execute();
```

Logout

```
var socket = this.getApp().newSocket('Logout');
socket.setAsync(false);
var action = new com.tibco.bpm.ipc.Action([
  com.tibco.bpm.ipc.AuthenticateRequests.logout('Logout', true)]);
socket.addParameter('action', action.toXml());
socket.execute();
```

The **com.tibco.bpm.ipc.AuthenticateRequests** class contains the following public methods.

Method	Description
`init()`	Constructor — initializes the class. **Returns** An instance of the **com.tibco.bpm.ipc.AuthenticateRequests** class.

`login(id,`
`nodeCtx)`
	Performs login authentication.
	
	Parameters

Method	Description
	id - (String) Unique ID of the request. This is the ID of the XML cache that will hold the XML document returned by the Action Processor. *nodeCtx* - (com.tibco.bpm.ipc.NodeCtx) Node connection object. Note - As there is not a data class for **NodeCtx** that can be created and passed into the **login** method, you must create one as shown in the example on AuthenticateRequests, for example: `new com.tibco.bpm.ipc.NodeCtx('userId', 'password', nodeId)` The *nodeId* parameter provides connection information — this can be created using the **vNodeId** data class — see vNodeId. **Returns** Server Request object for issuing a login request: **com.tibco.bpm.ipc.ServerRequest**.

`logout(id,`
 	
`releaseAllResources)`
	Performs logout and closes the SAL session.
	
	Parameters
	
	id - (String) Unique ID of the request. This is the ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	releaseAllResources - (boolean) If true, the SAL session is closed by the resulting request.
	
	Returns
	
	Server Request object for issuing a logout request: **com.tibco.bpm.ipc.ServerRequest**.

Socket

The **com.tibco.bpm.ipc.Socket** class provides socket access for components.

Note that the constructor for this class is private. Use the following factory method to create an instance:

```
com.tibco.bpm.ipc.Application.newSocket(id)
```

where:

— *id* is the unique ID assigned to the request. This is the ID of the XML cache that will hold the XML document returned by the Action Processor. The **newSocket** method dispatches the **socketResult** event when the result is returned.

For a code example showing the creation of a socket, execution of the request, and error handling, see Example Request.

The **com.tibco.bpm.ipc.Socket** class contains the following public methods.

Method	Description
`addParameter(strName, strValue)`	Registers the URL argument name and value pairs that will be appended to the base URL prior to request communication. **Parameters** *strName* - (String) The parameter name. *strValue* - (String) The parameter value. **Returns** None.
`cancelRequest (strRequestId)`	Cancels the named request. **Parameters** *strRequestId* - (String) Unique ID for the request (assigned when

Method	Description
	the socket was instanced). **Returns** None.
`execute()`	Opens the request and dispatches the socket call. **Returns** None.
`getApExMessage()`	**Returns** A string containing exception messages returned in the Action Processor response. Expressed as a <ap:ExceptionMessage> value in the response XML.
`getApExStackTrace()`	**Returns** A string containing exception stack trace returned in the Action Processor response. Expressed as a <ap:ExceptionStackTrace> value in the response XML.
`getApReturnCode()`	**Returns** A string containing the Action Processor request response code. Expressed as a <ap:ReturnCode> value in the response XML.
`getApReturnComment()`	**Returns** A string containing the Action Processor request response comment. Expressed as a <ap:ReturnComment> value in the response XML.
`getApReturnDate()`	**Returns**

Method	Description
	A string containing the date and time of the Action Processor response. Expressed as a <ap:ReturnDateTime> value in the response XML.
`getBaseUrl()`	**Returns** A string containing the base URL (Action Processor URL) to access the XML data.
`getErrorMessage()`	**Returns** A string containing an error message describing the error that occurred when *isSuccess* returns false, indicating a failure during socket processing.
`getId()`	**Returns** A string containing the object ID value used for request and cache identification (assigned when the socket was instanced). There will never be two **com.tibco.bpm.ipc.Sockets** with the same ID.
`getRequestType()`	**Returns** A string containing 'GET' or 'POST' (default = 'POST').
`getSsoErrorCount()`	**Returns** An integer that is the number of **vExceptionDetail** nodes returned by iProcess Server Objects in the response message.
`getSsoErrorMsg()`	**Returns** A string containing the value of any iProcess Server Objects-generated errors.
`getSsoExceptionNode()`	**Returns** A **jsx3.xml.Entity** object, which is the **sso:Exception** element

Method	Description
	returned in the response message.
`isSuccess()`	**Returns** A boolean that indicates the request communication success (true) or failure (false).
`reportErrors()`	Combines any error message from the socket, or Action Processor, with any iProcess Server Objects exception error messages. Also logs the messages and displays a message to user.
`setAsync(bAsync)`	Specifies whether request communication, with the Action Processor, will be asynchronous or synchronous. **Parameters** *bAsync* - (boolean) true for asynchronous communication, false for synchronous (default = true). **Returns** None.
`setBaseUrl(strUrl)`	Sets the base URL (Action Processor URL) for accessing the XML data. **Parameters** *strUrl* - (String) The base URL to access the XML data. **Returns** None.
`setErrorMsgPrefix (strPrefix)`	Used to specify a prefix for an action-specific error message that may be generated by an action. Use this method to add custom

Method	Description
	messages that will be displayed to the user if the action generates an error message. **Parameters** *strPrefix* - (String) Action-specific error message that will prefix any error messages generated by the action. **Returns** None.
`setRequestType (strType)`	Sets the request type. **Parameters** *strType* - (String) Type of request, either 'GET' or 'POST' (default = 'POST'). **Returns** None.
`setShowUserMsg(bShow)`	Specifies whether or not to display error messages, returned in the action result, to the user. **Parameters** *bShow* - (boolean) If true, any errors generated by an action will be displayed to the user. This can be set to false if the calling class needs to override the message show to the user (default = true). **Returns** None.

UtilityRequests

The **com.tibco.bpm.ipc.UtilityRequests** class provides general-utility Action Processor requests.

This class contains the following public methods.

Method	Description
`init()`	Constructor - initializes the class. **Returns** An instance of the **com.tibco.bpm.ipc.UtilityRequests** class.
`ping()`	Sends a simple request to verify that the Action Processor is responding. **Returns** A Server Request object (**com.tibco.bpm.ipc.ServerRequest**) for issuing a ping request.

XmlElement

The **com.tibco.bpm.ipc.XmlElement** class provides support for creating XML elements.

This class contains the following public methods.

Method	Description
`init(tagName, objText)`	Constructor - initializes the class. **Parameters** *tagName* - (String) The name that defines the XML element tag. *objText* - (boolean, object, number, string) This is an optional argument.

Method	Description

	When passed, the constructor attempts to convert the argument to a valid string, which is the text that the XML element encloses.
	
	Returns
	
	An instance of the **com.tibco.bpm.ipc.XmlElement** class.
	
	Example
	
	`new com.tibco.bpm.ipc.XmlElement("WorkQTag", "i2tagtest|swadmin|R");`
	
	returns the following XML element:
	
	`<WorkQTag>i2tagtest|swadmin|R</WorkQTag>`

`addAttribute (strName,`
 	
`strValue)`
	Adds a name/value pair as an attribute of the XML element.
	
	Parameters
	
	strName - (String) The name of the attribute.
	
	strValue - (String) The value of the attribute.
	
	Returns
	
	None.
	
	Example
	
	`var xmlElement = new com.tibco.bpm.ipc.XmlElement("WorkQ");`
	
	`xmlElement.addAttribute("tag","i2tagtest|swadmin|R");`
	
	creates the following XML element and adds an attribute:
	
	`<WorkQ tag="i2tagtest|swadmin|R"/>`

Method	Description

`addChild (objChild)`	Adds a child to the children of the XML element.
	
	Parameters
	
	objChild - (com.tibco.bpm.ipc.XmlElement) The child object to add to the XML element.
	
	Returns
	
	None.
	
	Example
	
	`var xmlElement = new com.tibco.bpm.ipc.XmlElement("WorkQ");`
	
	`var xmlChild = new com.tibco.bpm.ipc.XmlElement("WorkItem", "1234");`
	
	`xmlElement.addChild(xmlChild);`
	
	creates the following XML element with a child element:
	
	`<WorkQ><WorkItem>1234</WorkItem></WorkQ>`

`toXml()`	Outputs a valid XML string representing the XML element and all of its children. **Returns** A string containing the XML element and all of its children.

XslTransform

The **com.tibco.bpm.ipc.XslTransform** class provides support for performing XSL transforms on XML documents.

Note that the constructor for this class is private. Use the following factory method to create an instance:

```
com.tibco.bpm.ipc.Application.newXslTransform (xmlDoc, xslDoc)
```

where:

- *xmlDoc* is the path to the XML file, XML string, cacheId or **jsx3.xml.Document** object for the XML document.
- *xslDoc* is the path to the XSL file, XSL string, cacheId or **jsx3.xml.Document** object for the XSL document.

This class contains the following public methods.

Method	Description
`addParameter (strName, strValue)`	Adds a parameter to the XSL transform. **Parameters** *strName* - (String) The name of the parameter. *strValue* - (String) The value of the parameter. **Returns** None.
`removeParameter (strName)`	Removes a parameter from the XSL transform. **Parameters** *strName* - (String) The name of the parameter to remove from the XSL transform. **Returns** None.
`doTransform (strCacheId)`	Performs the XSL transform. **Parameters** *strCacheId* - (String) Optional. If provided, a **jsx3.xml.Document**

Method	Description
	containing the transform results is stored in cache under the given *strCacheId*. **Returns** A string containing the results of the XSL transform.

Action Request Classes

The **Action Request** classes provide functions that perform the actual action requests, such as getting a list of work items, forwarding a work item, setting case data, etc.

Note that each of the Action classes corresponds to an XML Server Object in the iProcess Server Objects object model. For instance, the **CaseManagerRequests** class corresponds to the **xCaseManager** Server Object. And, the methods available in the **CaseManagerRequests** class correspond to methods available on the **xCaseManager** Server Object.

Use the iProcess Server Objects (Java) on-line help system in conjunction with this document to get more detailed information about the functions provided by the JavaScript interface methods, as well as details about the parameters that must be passed to the JavaScript interface methods.

The classes available in this category are listed in the table below.

Class[1]	Description
CaseManagerRequests	Supports case management requests. This class corresponds to the **xCaseManager** object in the iProcess Server Objects object model.
NodeRequests	Supports retrieval of data related to a server node.

[1]All classes are prefixed with “com.tibco.bpm.ipc”.

Class[1]	Description
	This class corresponds to the **xNode** object in the iProcess Server Objects object model.
ProcManagerRequests	Provides access to the definition of a procedure. This class corresponds to the **xProcManager** object in the iProcess Server Objects object model.
UserRequests	Provides information relevant to a single user. This class corresponds to the **xUser** object in the iProcess Server Objects object model.
WorkQManagerRequests	Provides access to work queues for configuration, administration and reporting purposes. This class corresponds to the **xWorkQManager** object in the iProcess Server Objects object model.
WorkQRequests	Provides information and functionality relevant to a single work queue. This class corresponds to the **xWorkQ** object in the iProcess Server Objects object model.

Each of the action request classes is described in the following subsections.

CaseManagerRequests

The **com.tibco.bpm.ipc.CaseManagerRequests** class supports case management requests.

This class corresponds to the **xCaseManager** object in the iProcess Server Objects object model. For more information about the **xCaseManager** object and its methods, see the iProcess Server Objects on-line help system.

This class contains the following public methods.

[1]All classes are prefixed with “com.tibco.bpm.ipc”.

Method	Description
`init()`	Constructor - initializes the class. **Returns** An instance of the **com.tibco.bpm.ipc.CaseManagerRequests** class.

`getCaseCnt(id,`
`procTag,`
`filterExpression)`
	Returns the number of cases for a procedure. This can be used to determine the number of cases before calling a method that would return the cases in a list.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	procTag - (String) Tag that identifies the procedure containing the cases. This can be obtained in the </Tag> element in the procedure list XML.
	
	filterExpression - (String) Each case in the specified procedure is evaluated against this string expression, returning either True or False. If the evaluation returns True, the case is included in the count; if the evaluation returns False, the case is not included in the count. For example, to determine the number of cases that were started on 6/18/2007, pass "SW_ STARTEDDATE = !06/18/2007!" in this parameter. See the *Filtering Work Items and Cases* section in the *TIBCO iProcess Server Objects Programmer's Guide* for more information about creating filter expressions.
	
	Returns
	
	Number of cases of the specified procedure that satisfy the filter expression.

`makeACaseList(id,`	Returns the requested number of **vACase** objects, starting at

Method	Description

`procTag,`
`caseCriteria,`
`caseContent,`
`startIndex,`
`returnCount,`
`hold)`
	the specified index, that satisfy the specified filter expression.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	procTag - (String) Tag that identifies the procedure containing the cases you would like returned. This can be obtained in the </Tag> element in the procedure list XML.
	
	caseCriteria - (vACaseCriteria) Contains a string expression; only the cases that satisfy the expression are returned in the list. Also contains sort fields; the list of cases is sorted by the fields specified.
	
	caseContent - (vACaseContent) Specifies the amount of content to return with the **vACase** objects.
	
	startIndex - (int) The index number (zero based) of the first item you want returned from the list.
	
	returnCount (int) - The number of items you want returned from the list, starting at the index number specified in *startIndex* (up to the number of items in the list — e.g., you may ask for 20 items, but only 15 exist in the list).
	
	hold - (boolean) Specifies whether or not to hold the list on the server after this method call. (Note - This parameter does not have any meaning in this context at this time because there is no method provided in this interface that allows you to fetch additional items from the list.)
	
	Returns
	
	XML representation of **vACase**[] objects, as well as a **vACaseListState** object, which contains counts and other “state” information.

Method	Description
`triggerEvent(id, caseTag, eventName, resurrect, fields, updateOutstanding, recalculate)`	Triggers the processing of an Event step. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *caseTag* - (String) Tag that identifies the case in which you are triggering an event. This can be obtained in the </Tag> element in the case list XML. *eventName* - (String) The name of the Event step to trigger. *resurrect* - (boolean) True causes a closed case to be changed to an active case. *fields* - (vField[]) The fields whose case data you want to update. These objects also contain the new values for the case data. (See the *updateOutstanding* parameter For more information about updating work item data also.) *updateOutstanding* - (boolean) True = update both case data and work item data. False = update only case data. Note - "Work item data" is the data associated with a work item while it is in a work queue. When field values are modified and the work item is "kept" in the work queue, only the work item data is affected. When a work item is "released," the modified field values are written to "case data." *recalculate* - (String) Specifies how deadlines are recalculated. The options are: — **swNoReCalc** - Do not recalculate. Used to update outstanding work items, but you are not recalculating deadlines. — **swCaseOnly** - Recalculate deadlines in the main case only. — **swIncludeSubCases** - Recalculate deadlines in the main case and all sub-cases.

Method	Description
	Returns Void.

`jumpTo(id,`
`    caseTag,`
`    withdrawItems,`
`    sendItems,`
`    reason,`
`    wait,`
`    fields,`
`    updateOutstanding)`
	Specifies outstanding steps to withdraw and steps to jump to, making them outstanding.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	caseTag - (String) Tag that identifies the case in which you are withdrawing steps. This can be obtained in the </Tag> element in the case list XML.
	
	withdrawItems - (String[]) The names of the outstanding items to withdraw.
	
	sendItems - (String[]) The names of the steps that will become the new outstanding item(s) (for more information, see the *TIBCO iProcess Server Objects Programmer's Guide*).
	
	reason - (String) Describes the reason for the jump.
	
	wait - (boolean) Specifies whether the withdrawal and jump-to operation will be performed asynchronously (False) or synchronously (True). (Note - At this time, the iProcess Engine does not support asynchronous jumps. Therefore, if False is passed in this parameter, it will be changed to True internally, and the jump will be performed synchronously.)
	
	fields - (vField[]) The fields whose case data you want to update. These objects also contain the new values for the case data. (For more information about updating work item data, see the *updateOutstanding* parameter.)
	
	updateOutstanding - (boolean) True = update both case data and work item data. False = update only case data.

Method	Description
	Returns Void.

`getACases(id,`
`caseTags,`
`caseContent)`
	Returns a list of **vACase** objects, one for each case identified in the *caseTags* parameter.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	caseTags - (String[]) Tags that identify the cases you would like returned. These can be obtained in the </Tag> elements in the case list XML.
	
	caseContent - (vACaseContent) Specifies the amount of content to return with the **vACase** objects.
	
	Returns
	
	XML representation of **vACase**[] objects.

`getOutstandingItems(id,`
`caseTag,`
`outstandingItemContent,`
`4`
	Returns an array of **vOutstanding** objects, one for each item outstanding in a case family.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	caseTag - (String) Tag that identifies the case in which you are requesting outstanding steps. This can be obtained in the </Tag> element in the case list XML.
	
	outstandingItemContent - (vOutstandingItemContent) Specifies the types of outstanding objects to return from this method call. This allows you to limit the amount of content that is

Method	Description
	returned. *includeSubProcs* - (boolean) Specifies whether or not to also return outstanding items from sub-procedures that have been launched from the case specified in the *caseTag* parameter. True = return outstanding items recursively from sub-procedures. **Returns** XML representation of **vOutstanding**[] objects.

`setCaseSuspended(id,`
`caseTags,`
`suspend,`
`wait)`
	Sets or removes a case from a suspended state.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	caseTags - (String[]) Tags that identify the cases you would like suspended or removed from a suspended state. These can be obtained in the </Tag> elements in the case list XML.
	
	suspend - (boolean) Specifies the state to which the case family is set. True = suspend the case family; False = remove the case suspension from the case family.
	
	wait - (boolean) Specifies whether the state change will be performed asynchronously or synchronously:
	
	— If *wait* = False, the state change request is performed asynchronously. The state change is submitted to the background for processing at an indeterminate time and control is returned immediately to the client. Asynchronous state change is preferable for large case families, however, the actual time of the change is unknown.

Method	Description
	— If *wait* = True, the state change is performed synchronously. The client will be suspended until the background has completed the state change request for the entire case family. Synchronous processing is preferable if completion of the state change must be guaranteed. **Returns** Void.

`purgeCases(id,`
`caseTags)`
	Purges the specified cases from the system.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	caseTags - (String[]) Tags that identify the cases you would like purged. These can be obtained in the </Tag> elements in the case list XML.
	
	Returns
	
	Void.

`closeCases(id,`
`caseTags)`
	Closes one or more cases.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	caseTags - (String[]) Tags that identify the cases you would like closed. These can be obtained in the </Tag> elements in the case list XML.

Method	Description
	Returns Void.

`predictCase(id,`
`caseTag,`
`maxSubProc,`
`maxStepLoop)`
	Returns array of **vPredictedItem** objects, one for each work item that is outstanding and each item that is expected to be outstanding in the future.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	caseTag - (String) Tag that identifies the case in which you are performing the prediction operation. This can be obtained in the </Tag> element in the case list XML.
	
	maxSubProc - (int) The maximum depth of sub-procedure calls to make when performing case prediction. If a 0 (zero) or negative number is passed in this parameter, the value in the process attribute, MAX_SUB_PROCEDURE_DEPTH (which defaults to 100), is used - For more information about setting process attributes, see the *TIBCO iProcess Engine Administrator's Guide*.
	
	maxStepLoop - (int) The maximum number of times the case prediction process will loop through a step. If this value is reached, the prediction process for that particular branch of the procedure is halted. If a 0 (zero) or negative number is passed in this parameter, the value in the process attribute, MAX_PREDICTION_LOOPS (which defaults to 500) in used - For more information about setting process attributes, see the *TIBCO iProcess Engine Administrator's Guide*.
	
	Returns
	
	XML representation of **vPredictedItem**[] objects.

Method	Description
`setCaseData(id, caseTag, fields)`	Sets case data for one or more fields in a case. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *caseTag* - (String) Tag that identifies the case in which you want to set case data. This can be obtained in the </Tag> element in the case list XML. *fields* - (vField[]) The fields whose case data you want to set. These objects also contain the new values for the case data. **Returns** Void.

NodeRequests

The **com.tibco.bpm.ipc.NodeRequests** class supports retrieval of data related to a server node.

This class corresponds to the **xNode** object in the iProcess Server Objects object model. For more information about the **xNode** object and its methods, see the iProcess Server Objects on-line help system.

This class contains the following public methods.

Method	Description
`init()`	Constructor - initializes the class. **Returns** An instance of the **com.tibco.bpm.ipc.NodeRequests** class.

Method	Description
`getANode(id)`	Returns a **vANode** object, which provides all available information about the node. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. **Returns** XML representation of **vANode** object.
`makeWorkQIdList(id, startIndex, returnCount, hold)`	Returns the requested number of **vWorkQId** objects, starting at the specified index. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *startIndex* - (int) The index number (zero based) of the first item you want returned from the list. *returnCount* (int) - The number of items you want returned from the list, starting at the index number specified in *startIndex* (up to the number of items in the list — e.g., you may ask for 20 items, but only 15 exist in the list). *hold* - (boolean) Specifies whether or not to hold the list on the server after this method call. (Note - This parameter does not have any meaning in this context at this time because there is no method provided in this interface that allows you to fetch additional items from the list.) **Returns** XML representation of vWorkQId[] objects, as well as a

Method	Description
	vWorkQIdListState object, which contains counts and other “state” information.

`makeUserList(id,`
 `userContent,`
 `startIndex,`
 `returnCount,`
 `hold)`
	Returns the requested number of **vUser** objects, starting at the specified index number.
	
	 Parameters
	
	 id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	 userContent - (vUserContent) Specifies the amount of content to return with the **vUser** objects.
	
	 startIndex - (int) The index number (zero based) of the first item you want returned from the list.
	
	 returnCount (int) - The number of items you want returned from the list, starting at the index number specified in *startIndex* (up to the number of items in the list — e.g., you may ask for 20 items, but only 15 exist in the list).
	
	 hold - (boolean) Specifies whether or not to hold the list on the server after this method call. (Note - This parameter does not have any meaning in this context at this time because there is no method provided in this interface that allows you to fetch additional items from the list.)
	
	 Returns
	
	 XML representation of vUser[] objects, as well as a vUserListState object, which contains counts and other “state” information.

`getUserAttributes`
 `(id,`
 `userName,`
 `attributeNames)`
	Returns an array of **vAttribute** objects, one for each user attribute requested.
	
	 Parameters

Method	Description
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *userName* - (String) The name of the user whose attributes you would like returned. *attributeNames* - (String[]) An array of strings that identify the attributes you would like returned. **Returns** XML representation of **vAttribute**[] objects.

ProcManagerRequests

The **com.tibco.bpm.ipc.ProcManagerRequests** class provides access to the definition of a procedure.

This class corresponds to the **xProcManager** object in the iProcess Server Objects object model. For more information about the **xProcManager** object and its methods, see the iProcess Server Objects on-line help system.

This class contains the following public methods.

Method	Description
`init()`	Constructor - initializes the class. **Returns** An instance of the **com.tibco.bpm.ipc.ProcManagerRequests** class.
`getProcIds(id)`	Returns an array of **vProcId** objects containing information about each procedure defined on the node.

Method	Description
	Parameters *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. **Returns** XML representation of **vProcId**[] objects.

`getProcDefs(id,`
`procTags,`
`procDefContent)`
	Returns an array of **vProcDef** objects, one for each specified procedure, containing information about the definition of the procedures.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	procTags - (String[]) Tags that identify the procedures whose definition you want returned. These can be obtained in the </Tag> elements in the procedure list XML.
	
	procDefContent - (vProcDefContent) Specifies the amount of content to return with the **vProcDef** objects.
	
	Returns
	
	XML representation of **vProcDef**[] objects.

getAProcs(id,
`procTags,`
`procContent)`
	Returns an array of **vAProc** objects, one for each specified procedure, containing information about a procedure's status.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	procTags - (String[]) Tags that identify the procedures whose status

Method	Description
	information you want returned. These can be obtained in the </Tag> elements in the procedure list XML. *procContent* - (vAProcContent) Specifies the amount of content to return with the **vAProc** objects. **Returns** XML representation of **vAProc**[] objects.
`getFieldDefs(id, procTag)`	Returns an array of **vFieldDef** objects, one for each field defined in the procedure identified by the procedure tag. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *procTag* - (String) Tag that identifies the procedure whose field definitions you would like returned. This can be obtained in the </Tag> element in the procedure list XML. **Returns** XML representation of **vFieldDef**[] objects.
`getSteps(id, procTag, stepNames, stepContent)`	Returns an array of **vStepId** objects, one for each specified step in the specified procedure. You can also specify the amount of content to be included with the step objects using the *stepContent* parameter. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *procTag* - (String) Tag that identifies the procedure whose steps you would like returned. This can be obtained in the </Tag> element in the

Method	Description
	procedure list XML. *stepNames* - (String[]) The names of the steps you would like returned. *stepContent* - (vStepContent) Specifies the amount of content to return with the **vStepId** objects. **Returns** XML representation of **vStepId[]** objects.

`getPluginForm(id,`
`procTag,`
`stepName)`
	Returns a **vPluginForm** object containing identifying information about a form plug-in.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	procTag - (String) Tag that identifies the procedure in which the form is defined. This can be obtained in the </Tag> element in the procedure list XML.
	
	stepName - (String) The name of the step associated with the form.
	
	Returns
	
	XML representation of **vPluginForm** object.

`getProcVersions`
`(id,`
`procName,`
`procContent)`
	Returns an array of **vAProc** objects, one for each version of the specified procedure defined on the node (except procedures with a status of swIncomplete, which are not supported).
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.

Method	Description
	procName - (String) The name of the procedure whose versions you would like returned. *procContent* - (vAProcContent) Specifies whether or not **vProcSummary** objects are also retrieved from the server with the procedures. This summary information includes dynamically changing data such as active case count, closed case count, the number of the last case started, etc. Note that this data tends to be "expensive" in terms of the amount of time it takes to retrieve. **Returns** XML representation of **vAProc**[] objects.

UserRequests

The **com.tibco.bpm.ipc.UserRequests** class provides information relevant to a single user.

This class corresponds to the **xUser** object in the iProcess Server Objects object model. For more information about the **xUser** object and its methods, see the iProcess Server Objects on-line help system.

This class contains the following public methods.

Method	Description
`init()`	Constructor - initializes the class. **Returns** An instance of the **com.tibco.bpm.ipc.UserRequests** class.
`getStartProcIds(id)`	Returns an array of **vProcId** objects, one for each procedure for which the user can start a case.

Method	Description
	Parameters *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. **Returns** XML representation of **vProcId**[] objects.
`getAuditProcIds(id)`	Returns an array of **vProcId** objects, one for each procedure for which the user can access audit data. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. **Returns** XML representation of **vProcId**[] objects.
`getCustomAuditMsgDefs (id)`	Returns an array of **vAuditMsgDef** objects, one for each message that has been defined in the *SWDIR*`\etc\language.lng\auditusr.mes` file. This file is used to hold custom (i.e., user defined) audit trail messages, to be used with the **addCaseAuditEntry** method (see Adds user-defined audit trail messages to a live case.) **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. **Returns**

Method	Description
	XML representation of **vAuditMsgDef**[] objects.
`isPasswordExpired(id)`	Returns a boolean indicating whether or not a user's password has expired. True = the user’s password has expired; False = the user’s password has not expired. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. **Returns** boolean

`changePassword(id,`
`oldPassword,`
`newPassword)`
	Modifies the user's password.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	oldPassword - (String) The old password value for the user.
	
	newPassword - (String) The new password for the user.
	
	Returns
	
	Void.

`getProcs(id,`
`procTags)`
	Returns an array of **vProc** objects, one for each procedure identified by the specified tags.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML

Method	Description
	document returned by the Action Processor. *procTags* - (String[]) Tags that identify the procedures you want returned. These can be obtained in the </Tag> elements in the procedure list XML. **Returns** XML representation of **vProc**[] objects.

`startCase(id,`
`procTag,`
`description,`
`subProcPrecedence,`
`startStepName,`
`releaseItem,`
`validateFields,`
`fields)`
	Starts a case of the specified procedure.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	procTag - (String) Tag that identifies the procedure you would like to start. This can be obtained in the </Tag> element in the procedure list XML.
	
	description - (String) The case description to assign to the case that is started. If the procedure definition specifies that the description is optional or hidden, this parameter can contain an empty string ("").
	
	subProcPrecedence - (String) Specifies the precedence of sub-procedure statuses that are launched from the procedure. It allows you to specify that certain statuses are looked for first, second, or third. The valid options are:
	— **swPrecedenceR** - Released status only
	— **swPrecedenceUR** - Unreleased > Released
	— **swPrecedenceMR** - Model > Released
	— **swPrecedenceUMR** - Unreleased > Model > Released
	— **swPrecedenceMUR** - Model > Unreleased > Released

Method	Description
	startStepName - (String) The name of the step at which the case is started (default is to start case at first step).
`startCase` (Cont.)	**Parameters** (Cont.) *releaseItem* - (boolean) If True, the start step will be released after the case start. If False, the start step will be kept on the queue of the user starting the case. This is only relevant if the addressee of the start step is the same as the user starting the case. The addressee must be defined in one of the following ways on the Step Definition Addressee tab: — Explicitly - The user's name is entered in the Users column. — As SW_STARTER — As a role The *releaseItem* parameter is ignored if the user starting the case is not the addressee of the start step, or if the Fields column on the Step Definition Addressee tab is used to specify the addressee of the start step. The *releaseItem* flag is disabled if you specify a start step other than the first step in the *startStepName* argument. *validateFields* - (boolean) If True, the case start will validate that the markings designated as swRequired on the iProcess Modeler form of the start step have values and are sent to the server (using the *fields* parameter). *fields* - (vField[]) These objects contain the names and values of the field(s) within the procedure to include with the case when it's started. **Returns** The case number assigned to the case that is started.

Method	Description
`getSupervisedQIds(id)`	Returns an array of **vWorkQId** objects, one for each work queue that the user can supervise (for the purpose of defining participation and redirection schedules). **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. **Returns** XML representation of **vWorkQId**[] objects.

`getWorkQs(id,`
`workQTags)`
	Returns an array of **vWorkQ** objects, one for each work queue specified by the work queue tag.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	workQTags - (String[]) Tags that identify the work queues you want returned. These can be obtained in the </Tag> elements in the work queue list XML.
	
	Returns
	
	XML representation of **vWorkQ**[] objects.

`addCaseAuditEntry(id,`
`caseTag,`
`stepName,`
`stepDescription,`
`userName,`
	Adds user-defined audit trail messages to a live case.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.

Method	Description
`messageId)`	*caseTag* - (String) Tag that identifies the case in which you want to add a user-defined audit trail message. This can be obtained in the </Tag> element in the case list XML. *stepName* - (String) The name of the step associated with the audit entry. An empty string may be given if a name is not needed. *stepDescription* - (String) A description for the step associated with the audit entry. An empty string may be given if a description is not needed. *userName* - The name of the user associated with the audit entry. An empty string may be given if a description is not needed. *messageId* - A unique number identifying the audit trail message. This number can be in the range 256-999. Note - Although specifying message IDs in the range 128-255 will not cause an error, you are strongly urged to not use them as that range of IDs are used by other applications. Also see the **getCustomAuditMsgDefs** method on Returns an array of vAuditMsgDef objects, one for each message that has been defined in the SWDIR\etc\language.lng\auditusr.mes file. This file is used to hold custom (i.e., user defined) audit trail messages, to be used with the addCaseAuditEntry method (see Adds user-defined audit trail messages to a live case.). **Returns** Void.
`getUserId(id)`	Returns a **vUserId** object containing formation about the user. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.

Method	Description
	Returns XML representation of **vUserId** object.

WorkQManagerRequests

The **com.tibco.bpm.ipc.WorkQManagerRequests** class provides access to work queues for configuration, administration and reporting purposes.

This class corresponds to the **xWorkQManager** object in the iProcess Server Objects object model. For more information about the **xWorkQManager** object and its methods, see the iProcess Server Objects on-line help system.

This class contains the following public methods.

Method	Description
`init()`	Constructor - initializes the class. **Returns** An instance of the **com.tibco.bpm.ipc.WorkQManagerRequests** class.
`getCDQPDefs(id, workQTag)`	Returns an array of **vCDQPDef** objects, one for each CDQP field defined for the specified work queue. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *workQTag* - (String) Tag that identifies the work queue whose CDQPs you would like returned. This can be obtained in the </Tag> element in the work queue list XML.

Method	Description
	Returns XML representation of **vCDQPDef**[] objects.
`getAWorkQs(id, workQTags, workQContent)`	Returns an array of **vAWorkQ** objects, one for each specified work queue. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *workQTags* - (String[]) Tags that identify the work queues you want returned. These can be obtained in the </Tag> elements in the work queue list XML. *workQContent* - (vAWorkQContent) Specifies the amount of content to return with the **vAWorkQ** objects. **Returns** XML representation of **vAWorkQ**[] objects.
`getParticipations(id, workQTag)`	Returns an array of **vParticipation** objects, one for each participation schedule defined on the specified work queue. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *workQTag* - (String) Tag that identifies the work queue whose participation schedules you would like returned. This can be obtained in the </Tag> element in the work queue list XML. **Returns**

Method	Description
	XML representation of **vParticipation**[] objects.
`getRedirection(id, workQTag)`	Returns the redirection schedule (**vRedirection)** for the specified work queue. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *workQTag* - (String) Tag that identifies the work queue whose participation schedules you would like returned. This can be obtained in the </Tag> element in the work queue list XML. **Returns** XML representation of **vRedirection** object.
`getSupervisorNames(id, workQTag)`	Returns an array of strings, one for each user that can supervise the specified work queue. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *workQTag* - (String) Tag that identifies the work queue whose supervisors you would like returned. This can be obtained in the </Tag> element in the work queue list XML. **Returns** String[]
`makeAWorkQList(id, workQContent,`	Returns the requested number of **vAWorkQ** objects, starting at the specified index.

Method	Description
`startIndex,` `returnCount,` `hold)`	**Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *workQContent* - (vAWorkQContent) Specifies the amount of content to return with the **vAWorkQ** objects. *startIndex* - (int) The index number (zero based) of the first item you want returned from the list. *returnCount* (int) - The number of items you want returned from the list, starting at the index number specified in *startIndex* (up to the number of items in the list — e.g., you may ask for 20 items, but only 15 exist in the list). *hold* - (boolean) Specifies whether or not to hold the list on the server after this method call. (Note - This parameter does not have any meaning in this context at this time because there is no method provided in this interface that allows you to fetch additional items from the list.) **Returns** XML representation of vAWorkQ[] objects, as well as a vAWorkQListState object, which contains counts and other “state” information.
`createParticipations(id,` `workQTag,` `participations)`	Creates participation schedules for the specified work queue. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *workQTag* - (String) Tag that identifies the work queue for which you are creating a participation schedule. This can be obtained in the </Tag> element in the work queue list XML.

Method	Description
	participations - (vParticipation[]) These define the participation schedule(s) to create. **Returns** Void.

`changeParticipation(id,`
`workQTag,`
`existingParticipation,`
`changedParticipation)`
	Modifies an existing participation schedule for the specified work queue.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	workQTag - (String) Tag that identifies the work queue for which you are changing a participation schedule. This can be obtained in the </Tag> element in the work queue list XML.
	
	existingParticipation - (vParticipation) Specifies the participation schedule you would like to change.
	
	changedParticipation - (vParticipation) Specifies the new participation schedule.
	
	Returns
	
	Void.

`removeParticipations(id,`
`workQTag,`
`participations)`
	Removes one or more participation schedules for the specified work queue.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.

Method	Description
	workQTag - (String) Tag that identifies the work queue for which you are removing a participation schedule. This can be obtained in the </Tag> element in the work queue list XML. *participations* - (vParticipation[]) Specifies the participation schedules you would like to remove. **Returns** Void.
`changeRedirection(id, workQTag, redirection)`	Modifies the redirection schedule for the specified work queue. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *workQTag* - (String) Tag that identifies the work queue for which you are changing a redirection schedule. This can be obtained in the </Tag> element in the work queue list XML. *redirection* - (vRedirection) Specifies the redirection schedule you would like to change. **Returns** Void.
`cancelRedirection(id, workQTags)`	Cancels the redirection schedule for the specified work queue. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *workQTags* - (String[]) Tags that identify the work queues

Method	Description
	whose redirection schedules you would like canceled. These can be obtained in the </Tag> elements in the work queue list XML. **Returns** Void.
`addSupervisors(id, supervisorNames, workQTags)`	Adds one or more user names to the list of users who can supervise the specified work queues. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *supervisorNames* - (String[]) The user names to add to the list. *workQTags* - (String[]) Tags that identify the work queues for which you are adding supervisors. These can be obtained in the </Tag> elements in the work queue list XML. **Returns** Void.
`removeSupervisors(id, supervisorNames, workQTags)`	Removes one or more user names from the list of users who can supervise the specified work queues. **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *supervisorNames* - (String[]) The user names to remove from the list.

Method	Description
	workQTags - (String[]) Tags that identify the work queues from which you are removing supervisors. These can be obtained in the </Tag> elements in the work queue list XML. **Returns** Void.

WorkQRequests

The **com.tibco.bpm.ipc.WorkQRequests** class provides information and functionality relevant to a single work queue.

This class corresponds to the **xWorkQ** object in the iProcess Server Objects object model. For more information about the **xWorkQ** object and its methods, see the iProcess Server Objects on-line help system.

This class contains the following public methods.

Method	Description
`init()`	Constructor - initializes the class. **Returns** An instance of the **com.tibco.bpm.ipc.WorkQRequests** class.

`lockItems(id,`
`workQTag,`
`workItemTags,`
`wiFGContent)`
	Locks the specified work items and returns an array of **vWIFieldGroup** objects, one for each work item locked. Each **vWIFieldGroup** object represents the group of fields in the work item.
	
	Parameters

Method	Description
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *workQTag* - (String) Tag that identifies the work queue containing the work items you would like to lock. This can be obtained in the </Tag> element in the work queue list XML. *workItemTags* - (String[]) Tags that identify the work items to lock. These can be obtained in the </Tag> elements in the work item list XML. *wiFGContent* - (vWIFGContent) Specifies the amount of content (i.e., which fields) to return with the **vWIFieldGroup** objects. **Returns** XML representation of **vWIFieldGroup[]** objects.

`unlockItems(id,`
`        workQTag,`
`        workItemTags)`
	Unlocks the specified work items. All changes that have been made to the field values in the work items since they were locked are discarded
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	workQTag - (String) Tag that identifies the work queue containing the work items you would like to unlock. This can be obtained in the </Tag> element in the work queue list XML.
	
	workItemTags - (String[]) Tags that identify the work items to unlock. These can be obtained in the </Tag> elements in the work item list XML.

Method	Description
	Returns Void.

`releaseItems(id,`
`workQTag,`
`wiFieldData,`
`validateFields)`
	Releases the specified work items and stores updated work item data. Once released, the work item is removed from the current work queue and the process advances to the next step in the procedure.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	workQTag - (String) Tag that identifies the work queue containing the work items you would like to release. This can be obtained in the </Tag> element in the work queue list XML.
	
	wiFieldData - (vWIFieldData) The **vWIFieldData** object contains an array of vWIFieldGroup objects, which identify the work items to release, as well as provide updated work item data to save.
	
	validateFields - (boolean) If True, the server will validate the markings. If False, markings are not validated. Setting this parameter to True means the following: 1) Validate that the markings exist on the form, 2) validate that all required markings (swRequired) on the form are sent to the server (using the *wiFieldData* parameter) with non-empty data, and 3) validate that display markings (swDisplay) are not sent to the server.
	
	Returns
	
	Void.

`keepItems(id,`	Keeps the specified work items and stores updated work

Method	Description

`workQTag,`
`wiFieldData,`
`validateFields)`
	item data.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	workQTag - (String) Tag that identifies the work queue containing the work items you would like to keep. This can be obtained in the </Tag> element in the work queue list XML.
	
	wiFieldData - (vWIFieldData) The **vWIFieldData** object contains an array of vWIFieldGroup objects, which identify the work items to keep, as well as provide updated work item data to save.
	
	validateFields - (boolean) Indicates whether or not data entered into the work item fields should be validated against the step's form. True = Validate.
	
	Returns
	
	Void.

`getForwardToWorkQIds(id,`
`workQTag,`
`workItemTag)`
	Returns an array of **vWorkQId** objects, one for each work queue to which the specified work item can be forwarded.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	workQTag - (String) Tag that identifies the work queue containing the work item for which you are requesting forward information. This can be obtained in the </Tag> element in the work queue list XML.

Method	Description
	workItemTag - (String) Tag that identifies the work item whose forward information you are requesting. These can be obtained in the </Tag> elements in the work item list XML. **Returns** XML representation of **vWorkQId[]** objects.

`forwardItems(id,`
`workQTag,`
`workItemTags,`
`destWorkQTag)`
	Forwards the specified work items from the current work queue to the specified work queue.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	workQTag - (String) Tag that identifies the work queue containing the work items you are forwarding. This can be obtained in the </Tag> element in the work queue list XML.
	
	workItemTags - (String[]) Tags that identify the work items to forward. These can be obtained in the </Tag> elements in the work item list XML.
	
	destWorkQTag - (String) Tag that identifies the destination work queue. This can be obtained in the </Tag> element in the work queue list XML.
	
	Returns
	
	Void.

`makeWorkItemList(id,`
`workQTag,`
	This method creates a static list of **vWorkItem** objects on the server, and returns the requested number of objects, starting at the specified index.

Method	Description

`wiCriteria,`
`wiContent,`
`startIndex,`
`returnCount,`
`hold)`
	After calling this method, additional **vWorkItem** objects can be requested using either the **fetchWorkItemList** (see Returns additional vWorkItem objects from the list created with the makeWorkItemList method (see This method creates a static list of vWorkItem objects on the server, and returns the requested number of objects, starting at the specified index.). The requested number of objects are returned, starting at the specified index.) or the **fetchWorkItemListIfChanged** method (see Returns additional vWorkItem objects from the list created with the makeWorkItemList method (see This method creates a static list of vWorkItem objects on the server, and returns the requested number of objects, starting at the specified index.). The requested number of objects are returned, starting at the specified index.).
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	workQTag - (String) Tag that identifies the work queue containing the work items. This can be obtained in the </Tag> element in the work queue list XML.
	
	wiCriteria - (vWICriteria) Contains a string expression; only the work items that satisfy the expression are returned in the list. Also contains sort fields; the list of work items is sorted by the fields specified.
	
	wiContent - (vWIContent) Specifies the amount of content to return with the **vWorkItem** objects.
	
	startIndex - (int) The index number (zero based) of the first item you want returned from the list.
	
	returnCount (int) - The number of items you want returned from the list, starting at the index number specified in *startIndex* (up to the number of items in the list — e.g., you

Method	Description
	may ask for 20 items, but only 15 exist in the list). *hold* - (boolean) Specifies whether or not to hold the list on the server after this method call. If held, you can obtain additional items from the list by calling the **fetchWorkItemList** method (see Returns additional vWorkItem objects from the list created with the makeWorkItemList method (see This method creates a static list of vWorkItem objects on the server, and returns the requested number of objects, starting at the specified index.). The requested number of objects are returned, starting at the specified index.). **Returns** XML representation of vWorkItem[] objects, as well as a vWorkItemListState object, which contains counts, the held ID (needed if you will be calling one of the "fetch..." methods, and other "state" information.

`getWorkItems(id,`
`workQTag,`
`workItemTags,`
`wiContent)`
	Returns an array of **vWorkItem** objects, one for each work item identified by the *workItemTags* parameter.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	workQTag - (String) Tag that identifies the work queue containing the work items. This can be obtained in the </Tag> element in the work queue list XML.
	
	workItemTags - (String[]) Tags that identify the work items. These can be obtained in the </Tag> elements in the work item list XML.
	
	wiContent - (vWIContent) Specifies the amount of content to return with the **vWorkItem** objects.

Method	Description
	Returns XML representation of **vWorkItem[]** objects.

`fetchWorkItemList(id,`
`workQTag,`
`heldId,`
`startIndex,`
`returnCount,`
`hold,`
`refresh)`
	Returns additional **vWorkItem** objects from the list created with the **makeWorkItemList** method (see This method creates a static list of vWorkItem objects on the server, and returns the requested number of objects, starting at the specified index.). The requested number of objects are returned, starting at the specified index.
	
	Parameters
	
	id - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor.
	
	workQTag - (String) Tag that identifies the work queue containing the work items. This can be obtained in the </Tag> element in the work queue list XML.
	
	heldId - (String) Identifies the "held" list on the server. This ID is in the **vWorkItemListState** object returned by the **makeWorkItemList** method that originally created the list.
	
	startIndex - (int) The index number (zero based) of the first item you want returned from the list.
	
	returnCount (int) - The number of items you want returned from the list, starting at the index number specified in *startIndex*.
	
	hold - (boolean) Specifies whether or not to continue to hold the list on the server. Pass True if you are going to fetch more items from the list; pass False if you are done with the list and will not be fetching any more items.
	
	refresh - (boolean) Specifies whether or not to refresh (i.e., rebuild) the list of work items, using the same criteria and content passed in the **makeWorkItemList** method that

Method	Description
	originally created the list, before returning the requested work items. This allows you to ensure that the list is current. The held ID remains the same if the list is refreshed. Pass True to refresh the list; pass False to not refresh the list. **Returns** XML representation of vWorkItem[] objects, as well as a vWorkItemListState object, which contains counts and other “state” information.
`fetchWorkItemListIfChanged (id, workQTag, heldId, startIndex, returnCount, hold)`	Returns additional **vWorkItem** objects from the list created with the **makeWorkItemList** method (see This method creates a static list of vWorkItem objects on the server, and returns the requested number of objects, starting at the specified index.). The requested number of objects are returned, starting at the specified index. Note that this method differs from the **fetchWorkItemList** method in that it does not have a *refresh* parameter. This method always refreshes the list, and it returns the requested work items only if any items in the entire list have changed (not just the specified range) since the list was originally created with the **makeWorkItemList** method. If no work items in the list have changed since it was originally created, the returned list is empty (zero length). **Parameters** *id* - (String) The ID of the XML cache that will hold the XML document returned by the Action Processor. *workQTag* - (String) Tag that identifies the work queue containing the work items. This can be obtained in the </Tag> element in the work queue list XML.

Method	Description
	heldId - (String) Identifies the "held" list on the server. This ID is in the **vWorkItemListState** object returned by the **makeWorkItemList** method that originally created the list. *startIndex* - (int) The index number (zero based) of the first item you want returned from the list. *returnCount* (int) - The number of items you want returned from the list, starting at the index number specified in *startIndex* (up to the number of items in the list — e.g., you may ask for 20 items, but only 15 exist in the list). *hold* - (boolean) Specifies whether or not to continue to hold the list on the server. Pass True if you are going to fetch more items from the list; pass False if you are done with the list and will not be fetching any more items. **Returns** XML representation of vWorkItem[] objects, as well as a vWorkItemListState object, which contains counts and other “state” information.

Data Classes

The **Data** classes provide a means to create data objects that are passed in action requests, such as filter and sort criteria, dates, field objects, etc.

These classes correspond to Value Objects of the same name in the iProcess Server Objects object model. See the iProcess Server Objects on-line help system for more details.

The classes available in this category are listed in the table below.

Class[1]	Description
vACaseContent	Provides for specification of dependent objects that will be returned when **vACase** objects are returned by a request.
vACaseCriteria	Provides for specification of sort and filter criteria when a list of cases is requested.
vAProcContent	Provides for specification of dependent **vProcSummary** objects that will be returned when a list of procedures is requested.
vAWorkQContent	Provides for specification of dependent objects that will be returned when **vAWorkQ** objects are returned by a request.
vDate	Creates a valid XML object representing a **vDate** object.
vDateTime	Creates a valid XML object representing a **vDateTime** object.
vField	Creates a valid XML object representing a **vField** object.
vNodeId	Creates a valid XML object representing a **vNodeId** object.
vOutstandingItemContent	Provides for specification of dependent **vOutstandingItemContent** objects that will be returned when a list of **vOutstandingItem** objects is requested.
vParticipation	Creates a valid XML object representing a **vParticipation** object.
vProcDefContent	Provides for specification of dependent objects that will be returned when a list of **vProcDef** objects is requested.
vRedirection	Creates a valid XML object representing a **vRedirection** object.
vSortField	Creates a valid XML object representing a **vSortField** object.

[1]All classes are prefixed with “com.tibco.bpm.ipc”.

Class[1]	Description
vStepContent	Provides for specification of dependent objects that will be returned when a list of step objects (e.g., **vAutoStep**, **vNormalStep**, etc.) is requested.
vTime	Creates a valid XML object representing a **vTime** object.
vUserContent	Provides for specification of dependent objects that will be returned when a list of **vUser** objects is requested.
vWIContent	Provides for specification of dependent objects that will be returned when a list of **vWorkItem** objects is requested.
vWICriteria	Provides for specification of sort and filter criteria when a list of **vWorkItem** objects is requested.
vWIFGContent	Provides for specification of which work item fields to return when locking one or more work items with the **lockItems** method.
vWIFieldData	Creates a valid XML object representing a **vWIFieldData** object.
vWIFieldGroup	Creates a valid XML object representing a **vWIFieldGroup** object.

Each of the data classes is described in the following subsections.

vACaseContent

The **com.tibco.bpm.ipc.vACaseContent** class provides for specification of dependent objects that will be returned when **vACase** objects are returned by a request.

This class only includes an **init** method, which is the constructor for the class. It has the form:

[1]All classes are prefixed with “com.tibco.bpm.ipc”.

```
init(bIsReturnAllFields,
     aStrCaseFieldNames,
     bIsWithAuditData,
     bIsAuditAscending,
     strAuditFilterExpression)
```

where:

— *bIsReturnAllFields* - (boolean) Specifies whether or not ALL fields (vField objects) associated with the case are to be returned from the server. True = return all fields — if set to True, field names specified in the *aStrCaseFieldNames* parameter are ignored; False = use *aStrCaseFieldNames* names to determine the fields to return from the server.

— *aStrCaseFieldNames* - (String[]) The names of the case data fields (each representing a vField object) that are returned when cases are retrieved from the server. Note that these field names are ignored if *bIsReturnAllFields* is set to True.

— *bIsWithAuditData* - (boolean) Specifies whether or not case audit data (**vAuditStep** objects) is returned when **vACase** objects are retrieved from the server. True = case audit data will also be returned from the server. False = case audit data will not be returned from the server.

— *bIsAuditAscending* - (boolean) Specifies whether audit data is returned in chronologically ascending order (oldest to most recent) or chronologically descending order (most recent to oldest). True = audit data is returned in ascending order; False = audit data is returned in descending order.

— *strAuditFilterExpression* - (String) A filter expression used to filter **vAuditStep** objects returned with the case (**vACase**). If the *bIsWithAuditData* flag is True, only **vAuditStep** objects satisfying the filter criteria are returned.

Used as an Input Parameter on these Methods

`CaseManagerRequests.makeACaseList` - getCaseCnt

`CaseManagerRequests.getACases` - getACases

vACaseCriteria

The **com.tibco.bpm.ipc.vACaseCriteria** class provides for specification of sort and filter criteria when a list of cases is requested.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(strFilterExpression, aSortFields)
```

where:

— *strFilterExpression* - (String) A filter expression used to build the case list. Only the cases that satisfy the filter expression are included in the list.

— *aSortFields* - (vSortField[]) Define the order in which the cases are listed.

Used as an Input Parameter on these Methods

`CaseManagerRequests.makeACaseList` - makeACaseList

vAProcContent

The **com.tibco.bpm.ipc.vAProcContent** class provides for specification of dependent **vProcSummary** objects that will be returned when a list of procedures is requested.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(withProcSummary)
```

where:

— *withProcSummary* - (boolean) True = **vProcSummary** objects are also returned from the server; False = **vProcSummary** objects data are not returned from the server.

Used as an Input Parameter on these Methods

`ProcManagerRequests.getAProcs` - getAProcs

`ProcManagerRequests.getProcVersions` - getProcVersions

vAWorkQContent

The **com.tibco.bpm.ipc.vAWorkQContent** class provides for specification of dependent objects that will be returned when **vAWorkQ** objects are returned by a request.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(withParticipation,
     withRedirection,
     withSupervisorNames,
     withCDQPDefs)
```

where:

— *withParticipation* - (boolean) True = participation schedules (vParticipation) will also be returned from the server. False = participation schedules will NOT be returned from the server.

— *withRedirection* - (boolean) True = redirection schedules (vRedirection) will also be returned from the server. False = redirection schedules will NOT be returned from the server.

— *withSupervisorNames* - (boolean) True = names of the supervisors will also be returned from the server. False = names of the supervisors will NOT be returned from the server.

— *withCDQPDefs* - (boolean) True = CDQP definitions (**vCDQPDef**) will also be returned from the server. False = CDQP definitions will NOT be returned from the server.

Used as an Input Parameter on these Methods

`WorkQManagerRequests.getAWorkQs` - getAWorkQs

`WorkQManagerRequests.makeAWorkQList` - makeAWorkQList

vDate

The **com.tibco.bpm.ipc.vDate** class creates a valid XML object representing a **vDate** object. This object is used when creating participation schedules.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(tagName,
     valueSet,
     date)
```

where:

- *tagName* - (String) Specifies the XML tag in which the date will be contained. Because **vDate** is only used in participation schedules, this tag will be one of the following:

  `‘StartingDate’`

  `‘EndingDate’`

- *valueSet* - (boolean) True if a date has been assigned to the **vDate** object. False if the **vDate** object represents an “empty” date. (An empty starting date causes participation to begin on the next date allowed by the IsSunday - IsSaturday parameters. An empty ending date causes participation to last indefinitely.)
- *date* - (Date - in XML format) The date to start or end participation.

Used as an Input Parameter on:

```
vParticipation contructor - vParticipation
```

vDateTime

The **com.tibco.bpm.ipc.vDateTime** class creates a valid XML object representing a **vDateTime** object. This object is used when creating redirection schedules.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(strName,
     valueSet,
     dateTime)
```

where:

- *strName* - (String) Specifies the XML tag in which the DateTime will be contained. Because **vDateTime** is only used in redirection schedules, this tag will be one of the following:

  `‘StartingDateTime’`

  `‘EndingDateTime’`

- *valueSet* - (boolean) True if a DateTime has been assigned to the **vDateTime** object. False if the **vDateTime** object represents an “empty” DateTime. (An empty starting DateTime causes redirection to start immediately. An empty ending DateTime causes redirection to last indefinitely.)

- *dateTime* - (DateTime - in XML format) The date and time to start or end redirection.

Used as an Input Parameter on these Methods

```
vRedirection contructor - vRedirection
```

vField

The **com.tibco.bpm.ipc.vField** class creates a valid XML object representing a **vField** object.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(name,
     value,
     fieldType)
```

where:

- *name* - (String) The name of the field.

- *value* - The value of the field. The value type depends on the field type. It may be a String, Double, or a Date in XML format — see the table below.

- *fieldType* - (String) Describes the type of data in the field. The valid entries are the following **SWFieldType** enumerations:

fieldType	Description	value Type
swAttachment	Attachment	String
swComma	Comma-separated numeric	Double
swCompositeTable	Composite table	String
swDate	Date	Date
swMemo	Memo	String
swNumeric	Real number	Double
swText	ASCII text	String
swTime	Time	Date
swTimeStamp	Combination of date and time	String
swArrayOfComma	Array of comma-separated numeric	Array of Double
swArrayOfCompositeTable	Array of composite tables	Array of String
swArrayOfDate	Array of dates	Array of Date
swArrayOfMemo	Array of memos	Array of String
swArrayOfNumeric	Array of real numbers	Array of Double
swArrayOfText	Array of ASCII text values	Array of String
swArrayOfTime	Array of times	Array of Date

Used as an Input Parameter on these Methods

`CaseManagerRequests.triggerEvent` - getAProcs

`CaseManagerRequests.jumpTo` - getAProcs

`CaseManagerRequests.setCaseData` - setCaseData `userRequests.startCase` - startCase

Note: The **vField** object is also used in the vWIFieldGroup constructor.

vNodeId

The **com.tibco.bpm.ipc.vNodeId** class creates a valid XML object representing a **vNodeId** object.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(strName,
     strComputerName,
     strIpAddress,
     nTcpPort,
     bDirector)
```

where:

- *strName* - (String) Name of the node.
- *strComputerName* - (String) The name of the machine on which the iProcess Objects Server or iProcess Objects Director is installed.
- *strIpAddress* - (String) The IP address of the iProcess Objects Server or iProcess Objects Director.
- *nTcpPort* - (int) The TCP port number used to connect to the iProcess Objects Server or iProcess Objects Director.
- *bDirector* - (boolean) If True, the node represents an iProcess Objects Director. If False, the node represents an iProcess Objects Server.

Used as an Input Parameter on

The data class is used when creating a new node context for logging in — see the **login** method on Performs login authentication..

vOutstandingItemContent

The **com.tibco.bpm.ipc.vOutstandingItemContent** class provides for specification of dependent objects that will be returned when a list of **vOutstandingItem** objects is requested.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(withNormalItems,
     withEventItems,
     withEAIItems,
     withSubProcCallItems,
     withDynamicSubProcItems,
     withGraftItems,
     withTransactionControlItems)
```

where:

— *withNormalItems* - (boolean) True = outstanding items representing normal steps (**vNormalItem** objects) will be returned from the server.

— *withEventItems* - (boolean) True = outstanding items representing event steps (**vEventItem** objects) will be returned from the server.

— *withEAIItems* - (boolean) True = outstanding items representing EAI steps (**vEAIItem** objects) will be returned from the server.

— *withSubProcCallItems* - (boolean) True = outstanding items representing sub-procedure call steps (**vSubProcCallItem** objects) will be returned from the server.

— *withDynamicSubProcItems* - (boolean) True = outstanding items representing dynamic sub-procedure call steps (**vDynamicSubProcItem** objects) will be returned from the server.

— *withGraftItems* - (boolean) True = outstanding items representing graft steps (**vGraftItem** objects) will be returned from the server.

— *withTransactionControlItems* - (boolean) True = outstanding items representing transaction control steps (**vTransactionControlItem** objects) will be returned from the server.

Used as an Input Parameter on these Methods

`CaseManagerRequests.getOutstandingItems` - getOutstandingItems

vParticipation

The **com.tibco.bpm.ipc.vParticipation** class creates a valid XML object representing a **vParticipation** object. This object represents a participation schedule. This schedule allows you to specify that a user has access to another user's work queue for a specified period of time.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(tagName
     name,
     index,
     qName,
     startingDate,
     endingDate,
     startingTime,
     endingTime,
     sunday,
     monday,
     tuesday,
     wednesday,
     thursday,
     friday,
     saturday,
     userNames)
```

where:

— tagName - (String) The string you pass in this parameter will depend on what you are doing with the participation schedule. The tagName is used to create an XML tag that will enclose the participation schedule XML. The following are the strings you must pass according to the schedule’s use:

  ‘vParticipation’ - Pass this in the *tagName* parameter if you are passing the **vParticipation** object in the **createParticipation** or **removeParticipation** methods.

‘ExistingParticipation’ - Pass this in the *tagName* parameter if you are passing the **vParticipation** object in the *existingParticipation* parameter in the **changeParticipation** method.

‘ChangedParticipation’ - Pass this in the *tagName* parameter if you are passing the **vParticipation** object in the *changedParticipation* parameter in the **changeParticipation** method.

— *name* - (String) If you are changing or removing an existing participation schedule, the name is obtained by calling the **getParticipations** method (see getParticipations). If you are creating a new participation schedule with the **createParticipations** method, pass an empty string (““) in this parameter.

— *index* - (int) If you are changing or removing an existing participation schedule, the index is obtained by calling the **getParticipations** method (see getParticipations). If you are creating a new participation schedule with the **createParticipations** method, pass a 0 (zero) in this parameter.

— *qname* - (String) The name of the work queue to which the participation schedule is associated.

— *startingDate* - (vDate) The date on which participation starts. If the start date is empty, the participation schedule will start as soon as the schedule is added to the Participations list.

— *endingDate* - (vDate) The date on which participation ends. If the end date is empty, participation will continue indefinitely.

— *startingTime* - (vTime) the time this participation schedule starts each day, for the days of the week on which participation is allowed. For example, if 08:00 is specified as the start time, participation starts at 8am on all days that participation is allowed (based on the start date and days of the week allowed specified in the participation schedule). If the start time is empty, participation will start directly after midnight.

— *endingTime* - (vTime) The time participation ends each day, for the days of the week on which participation is allowed. For example, if 17:00 is specified as the end time, participation ends at 5pm on all days that participation is allowed (based on the end date and days of the week allowed specified in the participation schedule). If the end time is empty, participation ends at midnight.

— *sunday* - (boolean) True = participation is allowed on Sunday.

— *monday* - (boolean) True = participation is allowed on Monday.

— *tuesday* - (boolean) True = participation is allowed on Tuesday.

- — *wednesday* - (boolean) True = participation is allowed on Wednesday.
- — *thursday* - (boolean) True = participation is allowed on Thursday.
- — *friday* - (boolean) True = participation is allowed on Friday.
- — *saturday* - (boolean) True = participation is allowed on Saturday.
- — *userNames* - (String[]) The users that can participate in a work queue.

Used as an Input Parameter on these Methods

`WorkQManagerRequests.createParticipations` - createParticipations
`WorkQManagerRequests.changeParticipation` - changeParticipation
`WorkQManagerRequests.removeParticipations` - removeParticipations

vProcDefContent

The **com.tibco.bpm.ipc.vProcDefContent** class provides for specification of dependent objects that will be returned when a list of **vProcDef** objects is requested.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(withAdminBy,
     withStartBy,
     withNetworkNodes,
     withStepNames,
     withFieldDefs,
     withAuditData)
```

where:

- — *withAdminBy* - (boolean) True = administrative access data (in **vAccessUserRef** objects) will also be returned from the server.
- — *withStartBy* - (boolean) True = case start access data (**vAccessUserRef** objects) will also be returned from the server.
- — *withNetworkNodes* - (boolean) True = names of the nodes will also be returned from the server.
- — *withStepNames* - (boolean) True = step information (**vStepId** objects) will also be returned from the server.

- — *withFieldDefs* - (boolean) True = field definitions (**vFieldDef** objects) will also be returned from the server.
- — *withAuditData* - (boolean) True = procedure audit data (**vProcAudit** objects) is returned from the server.

Used as an Input Parameter on these Methods

`ProcManagerRequests.getProcDefs` — getAProcs

vRedirection

The **com.tibco.bpm.ipc.vRedirection** class creates a valid XML object representing a **vRedirection** object. This object represents a redirection schedule for a work queue. This schedule allows you to “redirect” one user’s or group’s work items to the work queue of another user or group for a specified period of time.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(startDateTime,
     endDateTime,
     workQName)
```

where:

- — *startDateTime* - (vDateTime) The date and time that the redirection starts. If the starting date and time is empty, redirection will start as soon as the redirection schedule is created.
- — *endDateTime* - (vDateTime) The date and time that the redirection ends. If the end date and time is empty, redirection will continue indefinitely.
- — *workQName* - (String) The user or group to whom the work queue is redirected.

Used as an Input Parameter on these Methods

`WorkQManagerRequests.changeRedirection` - changeRedirection
`WorkQManagerRequests.cancelRedirection` - cancelRedirection

vSortField

The **com.tibco.bpm.ipc.vSortField** class creates a valid XML object representing a **vSortField** object. This object specifies a field on which the cases or work items are to be sorted when listing them in a list. It also allows you to specify whether the cases/work items in the list should be in ascending or descending order, and what data type should be used for the sort comparison.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(strFieldName,
     bIsAscending,
     strSortTypeAs)
```

where:

- *strFieldName* - (String) The name of the field upon which the cases or work items will be sorted. This can be either a system field or a case data field.
- *bIsAscending* - (boolean) True = sort in ascending order; False = sort in descending order.
- *strSortTypeAs* - (String) Specifies that the values in the sort fields are to be converted to a specific data type before performing the sort comparison. The data types allowed are one of the following **SWSortType** enumerations:

SortTypeAs	Description
swDateSort	Sort as date
swDateTimeSort	Sort as date/time
swNumericSort	Sort as real number
swTextSort	Sort as text
swTimeSort	Sort as time

Sorting by a specified type is only applicable when sorting on:

- Case description (SW_CASEDESC)

- Case Data Queue Parameter (CDQP) fields
- Work Queue Parameter fields (SW_QPARAM1-4)

The value of the sort field will be converted to the specified sort type before doing the sorting. For example, text fields containing numeric information could be sorted as numbers by setting the sort type accordingly. Note, however, that if the sort field does not contain something readily convertible to the specified type, the sort results may be unexpected. For example, if sorting text as a numeric field but some of the text fields contain non-numeric data, the results of the conversion are not defined, so the sort results may not be what you expected.

Used as an Input Parameter on:

`vACaseCriteria contructor` - vACaseCriteria
`vWICriteria constructor` - vWICriteria

vStepContent

The **com.tibco.bpm.ipc.vStepContent** class provides for specification of dependent objects that will be returned when a list of step objects (e.g., **vAutoStep**, **vNormalStep**, etc.) is requested.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(withRouting,
     withMarkings,
     withPublicFields)
```

where:

— *withRouting* - (boolean) True = dependent objects will also be returned.

— *withMarkings* - (boolean) True = marking data will also be returned from the server.

— *withPublicFields* - (boolean) True = public field data will also be returned from the server.

Used as an Input Parameter on these Methods

`ProcManagerRequests.getSteps` - getSteps

vTime

The **com.tibco.bpm.ipc.vTime** class creates a valid XML object representing a **vTime** object. This object is used when creating participation schedules.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(tagName,
     valueSet,
     time)
```

where:

— *tagName* - (String) Specifies the XML tag in which the time will be contained. Because **vTime** is only used in participation schedules, this tag will be one of the following:

  `‘StartingTime’`

  `‘EndingingTime’`

— *valueSet* - (boolean) True if a time has been assigned to the **vTime** object. False if the **vTime** object represents an “empty” time. (An empty starting time causes participation to start directly after midnight on the days that participation is allowed (according to the other parameters). An empty ending time causes participation to end at midnight on the days that participation is allowed (according to the other parameters).

— *time* - (String) The time to start or end participation.

Used as an Input Parameter on:

`vParticipation contructor` - vParticipation

vUserContent

The **com.tibco.bpm.ipc.vUserContent** class provides for specification of dependent objects that will be returned when a list of **vUser** objects is requested.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(withAttributes,
     withGroups,
     withRoles,
     withWorkQs)
```

where:

- *withAttributes* - (boolean) True = attribute data (**vAttribute** objects) will also be returned from the server.
- *withGroups* - (boolean) True = names of the groups will be returned from the server.
- *withRoles* - (boolean) True = names of the roles will also be returned from the server.
- *withWorkQs* - (boolean) True = work queues (**vWorkQ** objects) will also be returned for the user.

Used as an Input Parameter on these Methods

`NodeRequests.makeUserList` - makeUserList

vWIContent

The **com.tibco.bpm.ipc.vWIContent** class provides for specification of dependent objects that will be returned when a list of **vWorkItem** objects is requested.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(bIsReturnAllFields,
     aStrCaseFieldNames,
   bIsReturnAllCDQPs,
   aStrCDQPNames)
```

where:

- *bIsReturnAllFields* - (boolean) Specifies whether or not ALL fields (vField objects) associated with the work item are to be returned from the server. True = return all fields — if set to True, field names specified in the *aStrCaseFieldNames* parameter are ignored; False = use *aStrCaseFieldNames* names to determine the fields to return from the server.
- *aStrCaseFieldNames* - (String[]) The names of the case data fields (each representing a vField object) that are returned when work items are retrieved from the server. Note that these field names are ignored if *bIsReturnAllFields* is set to True.
- *bIsReturnAllCDQPs* - (boolean) Specifies whether or not ALL Case Data Queue Parameter (CDQP) fields (**vCDQP** objects) associated with the work item are to be returned from the server. True = return all CDQPs — if set to True, CDQPs specified in the *aStrCDQPNames* parameter are ignored; False = use *aStrCDQPNames* to determine the CDQPs to return from the server.
- *aStrCDQPNames* - (String[]) Specifies the names of the Case Data Queue Parameter (CDQP) fields (each representing a **vCDQP** object) that are returned when work items are retrieved from the server.

Used as an Input Parameter on these Methods

`WorkQRequests.makeWorkItemList` - makeWorkItemList

`WorkQRequests.getWorkItems` - getWorkItems

vWICriteria

The **com.tibco.bpm.ipc.vWICriteria** class provides for specification of sort and filter criteria when a list of **vWorkItem** objects is requested.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(strFilterExpression,
     aSortFields)
```

where:

— *strFilterExpression* - (String) A filter expression used to build the work item list. Only the work items that satisfy the filter expression are included in the list.

— *aSortFields* - (vSortField[]) These define the order in which the work items are listed.

Used as an Input Parameter on these Methods

`WorkQRequests.makeWorkItemList` - makeWorkItemList

vWIFGContent

The **com.tibco.bpm.ipc.vWIFGContent** class provides for specification of which work item fields to return when locking one or more work items with the **lockItems** method.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(namesOrFieldsOptionType)
```

where:

— *namesOrFieldsOptionType* - (String[] or String) You can pass in an array of field names to explicitly state which fields to return, or you can pass in one of the following **SWFieldsOptionType** enumerations:

  ssoFormMarkings - Returns only visible fields/markings on the form (based on conditional statements on the form).

  ssoAllMarkings - Returns all fields/markings, even if not visible on the form (based on conditional statements on the form).

Used as an Input Parameter on these Methods

`WorkQRequests.lockItems` - lockItems

vWIFieldData

The **com.tibco.bpm.ipc.vWIFieldData** class creates a valid XML object representing a **vWIFieldData** object.

Note that this object is simply a wrapper around an array of vWIFieldGroup objects.

This object is used to identify the work items to keep or release, as well as provide updated work item data to save. It is also used when rendering a form for a specific work item.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(wiFieldGroups)
```

where:

— *wiFieldGroups* - (vWIFieldGroup[]) Identify the work items to keep, release, or render form for.

Used as an Input Parameter on these Methods

`WorkQRequests.keepItems` - keepItems
`WorkQRequests.releaseItems` - releaseItems

vWIFieldGroup

The **com.tibco.bpm.ipc.vWIFieldGroup** class creates a valid XML object representing a **vWIFieldGroup** object.

This object identifies a work item and its associated fields.

This class only includes an **init** method, which is the constructor for the class. It has the form:

```
init(workItemTag, fields)
```

where:

— *workItemTag* - (String) Tag that identifies the work item. These can be obtained in the </Tag> elements in the work item list XML.

— *fields* - (vField[]) The fields in the work item.

Used as an Input Parameter on:

`vWIFieldData contructor` - vWIFieldData

Note - An array of **vWIFieldGroup** objects is also returned by the **WorkQRequest.lockItems** method.

Example Request

The following example shows the code needed to create an Action Processor request, create a socket call, and perform an XSL transform on the XML results returned by the Action Processor.

```
//create an instance of the Action class and pass the request object
into
//the constructor
var action = new com.tibco.bpm.ipc.Action([
//call method of WorkQRequests class to create a getWorkItems request
object
     com.tibco.bpm.ipc.WorkQRequests.getWorkItems(
     this.getXmlCacheId(),   //the xml cache id of the application
     workQTag,               //variable containing the work queue tag
     itemTags,               //an array of work item tag
     //create instance of vWIContent class
     new com.tibco.bpm.ipc.vWIContent(false, false, false, false) )]);
//create instance of Socket class using xml cache id
var socket = this.getApp().newSocket(this.getXmlCacheId());
socket.setAsync(false);                    //make request synchronous
socket.addParameter('action', action.toXml());  //add the xml produced
by the
                                //Action object to the request URL
socket.setShowUserMsg(false);   //suppress any user error message that
may result
                            //if work items are no longer in queue
socket.execute();           //dispatch the socket call
 //get xml element containing any iPSO exceptions returned by action
processor
var ssoExNode = socket.getSsoExceptionNode();
if (ssoExNode != null) {    //iPSO returned an exception - check the
error code to
                                //see if it was work item not found
    if (ssoExNode.selectSingleNode('sso:ErrorCode').getValue() !=
'swItemErrErr') {
         //some other exception occurred - check for other socket errors
         if (!socket.isSuccess() || socket.getSsoErrorMsg() != null) {
              //either a socket or action processor error occurred
              socket.setShowUserMsg(true);  //remove suppression of user
error messages
              socket.reportErrors(); //report the errors to the user
              return;
         }
    }
}
//perform an xsl transform on the resulting xml document in cache
var xsltPath = '/components/ListContainer/xsl/actionProcessorToCdf.xsl';
 //path to
 //the xsl file
//create instance of XslTransform class
var transform = this.getApp().newXslTransform(this.getXmlCacheId(),
xsltPath);
//perform xsl transform to convert action processor xml to CDF xml for
GI components
var cdfDoc = transform.doTransform();
```

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

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.