

TIBCO ActiveMatrix® BPM Client Application Developer's Guide

*Software Release 4.2
August 2017*

Document Update: December 2017

Important Information

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

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

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

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

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

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

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

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

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

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

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

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

TIBCO Software Inc. Confidential Information

Contents

Figures	13
TIBCO Documentation and Support Services	14
Overview of the Client Application Framework	16
Custom Client Application	17
Example Applications	18
Bundled Applications	18
Using the openworkitem Application in Your Application	19
Using the startbizaction Application in Your Application	20
Hosting Applications on External Web Servers	21
Configure a Theme for a Custom Client Application	23
Theme Color Definitions	24
Internationalizing Applications	25
Internationalizing a Cloned Workapp Application	25
Internationalizing a Custom Application	26
BPMTranslationService	28
LocaleInfoModel	28
BPMTranslationServiceConfig	28
getLocaleInfo	29
registerApp	29
unregisterApp	30
App CDN	30
Business Components	32
Component Showcase Application	32
BUSINESS ACTIONS	34
START BIZ ACTION	35
WORK ITEM DETAILS	36
Case Management Application	37
Configure the Case Management Application	38
Mobile Case Manager Application	39
Configure the Mobile Case Manager Application	40
Adding a Business Component to Your Application	41
Customizing a Business Component	43
Business Components Reference	43
bpm-attributes-view	43
bpm-business-actions-drop-down	44
bpm-case-actions	45

bpm-case-data	46
bpm-case-details-page	47
bpm-case-event-view	49
bpm-case-instance-summary	50
bpm-case-states	51
bpm-case-types	52
bpm-comments	53
bpm-document-upload	54
bpm-event-view	55
bpm-highcharts	56
Define a Data Model in the Controller	57
Define a Chart Configuration Object in the Controller	58
Call the getChart Operation from the Controller	59
Include the Highcharts Library in your HTML File	59
bpm-linked-case-instances	60
bpm-login	60
bpm-process-view	61
bpm-summary-view	61
bpm-work-item-history	62
bpm-work-list	63
bpm-work-views	64
documents-view	65
mbpm-case-actions	66
mbpm-case-creator	67
mbpm-case-data	67
mbpm-case-event-view	68
mbpm-case-instance-summary	69
mbpm-case-states	70
mbpm-case-types	71
mbpm-comments	72
mbpm-documents	72
mbpm-document-upload	73
mbpm-work-list	74
work-item-data	74
work-item-detail-nav	75
workListData Object	76
workViewData Object	76
Business Component Services	78
Invoking a Business Component Service Operation From Your Application	78

Suppressing Errors Displayed by Business Component Services	81
Using the Lightweight TIBCO Component Framework	82
Business Component Services Reference	82
BPMBusinessActionsService	82
BizActionHandler	83
BusinessActionsServiceModel	84
StartBusinessActionServiceModel	84
fetchChildCategories	85
fetchTopCategories	86
startBizAction	86
BPMCaseEventViewService	87
CaseEventViewModel	87
getSearchCaseReport	88
BPMChartService	88
ChartModel	89
getChart	89
getChartData	89
BPMCommentService	90
CommentModel	90
addCaseComment	91
addProcessInstanceComment	92
addWorkItemComment	92
getCaseComments	93
getProclnstComments	94
getWorkItemComments	94
BPMEventViewService	95
EventViewModel	95
getEventView	95
BPMLoginService	95
LoginServiceModel	96
getExtendedUserInfo	96
getUserInfo	97
login	97
logout	98
BPMWorkItemDetailsService	98
WorkItemDetailsModel	98
fetchWorkItem	99
populateWorkItemAttributes	100
populateWorkItemData	100

populateWorkItemDetails	101
populateWorkItemProcessDetails	101
BPMWorkItemHistoryService	101
WorkItemHistoryModel	102
getWorkItemHistory	102
BPMWorklistService	102
OpenWorkitemServiceModel	103
WorklistServiceModel	104
allocateWorkItem	104
getViewsForResource	105
getWorklistItems	106
getWorkListItemsForGlobalData	107
getWorkListItemsForView	108
openNextWorkItem	108
openWorkItem	109
unallocateWorkItem	109
CaseDocumentService	110
CaseDocumentServiceModel	110
deleteDocument	111
deleteOrphanedFolders	111
downloadDocument	112
fetchDocuments	112
openDocument	113
CaseInformationService	113
CaseInformationServiceModel	114
getCaseData	114
getCaseInformation	115
listCaseActions	116
CaseManagementService	117
CaseManagementModel	117
findAllCases	119
getCaseData	120
getCaseInformation	120
getCaseStates	122
getRelatedCases	122
listCaseActions	123
listCaseTypes	124
objectAPI	126
objectAPI Configuration	127

objectAPI Reference	128
RestApiService	129
AuditService	129
Purge Audit Status	135
BusinessDeadlineService	137
Duration Definitions	137
Minimum Hours in Working Day Calculations	138
BusinessServiceService	138
CaseActionService	139
DirectoryService	140
DocumentService	143
Queries for findDocuments	146
DynamicResourceQueryService	147
EntityResolverService	148
ExporterService	148
GlobaldataAdminService	148
GlobaldataService	151
OrgEntityConfigService	158
OrgModelService	159
OrgResourceService	162
PageFlowService	164
ProcessManagementService	165
ResourceQueryService	178
RoleService	179
SecurityService	179
StatisticsService	180
UserSettingsService	180
WorkCalService	180
WorkItemManagementService	187
WorkListService	194
WorkPresentationService	198
Using the objectAPI in BPM Applications	201
Organization Models and LDAP	201
Navigating the Organization Model	201
Navigating Large Organization Models	202
Organization Model Entity Properties	203
Creating an LDAP Container	209
Creating an LDAP Container Using an LDAP Query Source	209
Creating an LDAP Container Using an LDAP Group Source	211

Defining Secondary LDAP Sources in an LDAP Container	212
LDAP Queries	213
Mapping Users	214
Resource Attributes	217
Setting Resource Attribute Values	218
Mapping Resource Attributes to LDAP Attributes	219
Organization Relationships	221
Overriding Organization Relationships	222
Moving a Resource to Different LDAP Container	223
Updating Push Destinations	224
Updating Push Destinations for Organizational Entities	224
Updating Push Destinations for Resources	225
Configuring a Dynamic Organization Model Extension Point	226
Candidate Queries	226
Configuring a Candidate Query	228
Removing a Candidate Query Configuration	229
Processes	230
Listing Available Process Templates	230
Starting a Process Instance	231
Listing Available Process Instances	233
Sorting and Filtering Lists of Process Templates and Instances	233
Query Parameter Syntax	234
Query Parameter Attributes	234
Using User-Defined Attributes	236
Query Parameter Operators	236
Using Wildcards	237
Date Time Data Type Specification	238
Query Parameter Examples	239
Process Instance State Transitions	239
Handling Process Instance Failures	241
Migrating a Process Instance to a Different Version	243
Migration Points	243
Migration Rules	243
How Process Instances Are Migrated	244
Process Migration Functions	244
Business Services	244
Starting a Business Service	245
Injecting Data into a Business Service	247
Pageflows	248

Starting and Processing a Pageflow	248
Injecting Data into a Pageflow	248
Work Lists and Work List Views	249
Work Lists	249
Retrieving a Work List	250
Sorting and Filtering Work Lists	250
Sort and Filter Expression Syntax	250
Wildcard Characters	251
Sort Expressions	251
Sort Filter Criteria	251
Using DateTime Fields	257
Using DateTime Ranges	258
Using Work Item Attribute Fields	259
Work List Views	259
Creating a Work List View	260
Custom Data	260
Sorting and Filtering Work List Views	260
Work List View Access	261
Editing Work List Views	261
Using Work List Views	262
Retrieving Work Items from a Work List View	262
Retrieving a List of Work Views for a Resource	262
Public Work List Views	263
Creating and Managing Supervised Work List Views	263
Creating and Managing Supervised Work List Views - Example 1	264
Creating and Managing Supervised Work List Views - Example 2	265
Deleting Work List Views	265
Work Items	265
Work Item Versions	266
Work Item States	266
Work Item State Transitions	267
Retrieving Information on Work Items	270
Offering and Allocating Work Items	271
Allocating a Work Item to a Target Resource	272
Reallocating a Work Item	273
Reallocating to Offer Set or to World	274
Reoffering a Work Item	274
Required System Actions	274
Opening a Work Item	275

Processing a Work Item	275
openWorkItem	276
closeWorkItem	276
completeWorkItem	276
cancelWorkItem	277
skipWorkItem	277
saveOpenWorkItem	277
pendWorkItem	277
rescheduleWorkItem	277
setWorkItemPriority	279
Example of Processing a Work Item	280
Example of Processing a Work Item with a Form	281
Processing Chained Work Items	282
Handling a Work Item That Contains Business Data	284
Accessing Hidden Work Items	284
Case Data Models	285
Retrieving Case Model Database Scripts	287
Forms	288
TIBCO Forms	288
Binding the Client Application to a Channel	289
Identifying the Client Channel in a Service Call	289
Rendering a TIBCO Business Studio Form	290
Custom Forms	291
Presentation Details	291
Data Format and Structures	291
Displaying a Work Item Form	292
Displaying a Form in a Pageflow	293
Displaying a Form in a Business Service	294
Integrating TIBCO Forms with Custom Client Applications	294
Injecting the Forms Runtime Adapter in the Browser	295
Typical Flow of Events	296
iframe Integration	297
Forms Runtime Adapter	298
com.tibco.forms.client.FormRunner	298
com.tibco.forms.client.Form	302
com.tibco.forms.client.LoadStat	306
Events	307
Executing a Query	307
Dynamic Queries	307

Registered Queries	308
Correlated Events	309
Defining Query Filter Strings	309
Filter Expression Format	309
Functions	309
Operands	312
Using String Literals	313
Using DateTime Literals	313
Using Late-Bound Parameter Literals	314
Operators	314
Using Attributes in Query Filters	315
AUDIT Option Attributes	316
CASE_STATS Option Attributes	320
PROCESS_STATS Option Attributes	321
PROCESS_TEMPLATES Attributes	322
REGISTERED_ATTRIBUTES Option Attributes	322
REGISTERED_COMPONENTS Option Attributes	323
REGISTERED_QUERIES Option Attributes	323
USER_ACTIVITY Option Attributes	324
WORKITEM_STATS Option Attributes	325
Message Categories and Attribute Contents	326
Example Queries	327
Event Measures	328
Purging Audit Data	328
Calendars	328
Base and Overlay Calendars	329
Exclusions	330
Time Zones	330
Calendar References	330
Assigning a Calendar Reference to a Calendar	331
Calendars and User Tasks	332
Creating Calendars	332
Identifying Calendars	335
Adding Recurring Exclusions	335
Recurrence Rule	335
Scheduling	337
Scheduling Example	338
Comments	339
System Actions	340

Figures

Navigating the Organization Model	202
Creating an LDAP Container Using an LDAP Query Source	210
Creating an LDAP Container Using an LDAP Group Source	211
Mapping a User	216
Setting Resource Attribute Values	219
Mapping Resource Attributes to LDAP Attributes	220
Moving a Resource to a Different LDAP Container	223
Updating Push Destinations for Organizational Entities	224
Updating Push Destinations for Resources	225
Configuring a Dynamic Organization Model Extension Point	226
Configuring a Candidate Query	229
Removing a Candidate Query	230
Listing Available Process Templates	231
Starting a Process Instance	232
Listing Available Process Instances	233
Starting a Business Service	246
Injecting Data into a Business Service	247
Injecting a Pageflow Event	249
Supervised Work Lists	264
State Transitions	268
Allocating a Work Item	272
Reallocating a Work Item	273
Processing a Work Item	280
Processing a Work Item with a Form	281
Chained Work Items	283
Creating and Assigning a Calendar Reference	331
Creating Base and Overlay Calendars	334
Calculating a Deadline	337

TIBCO Documentation and Support Services

How to Access TIBCO Documentation

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

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

Product-Specific Documentation

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

`TIBCO_HOME/release_notes/TIB_amx-bpm_version_docinfo.html`

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

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

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

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

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

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

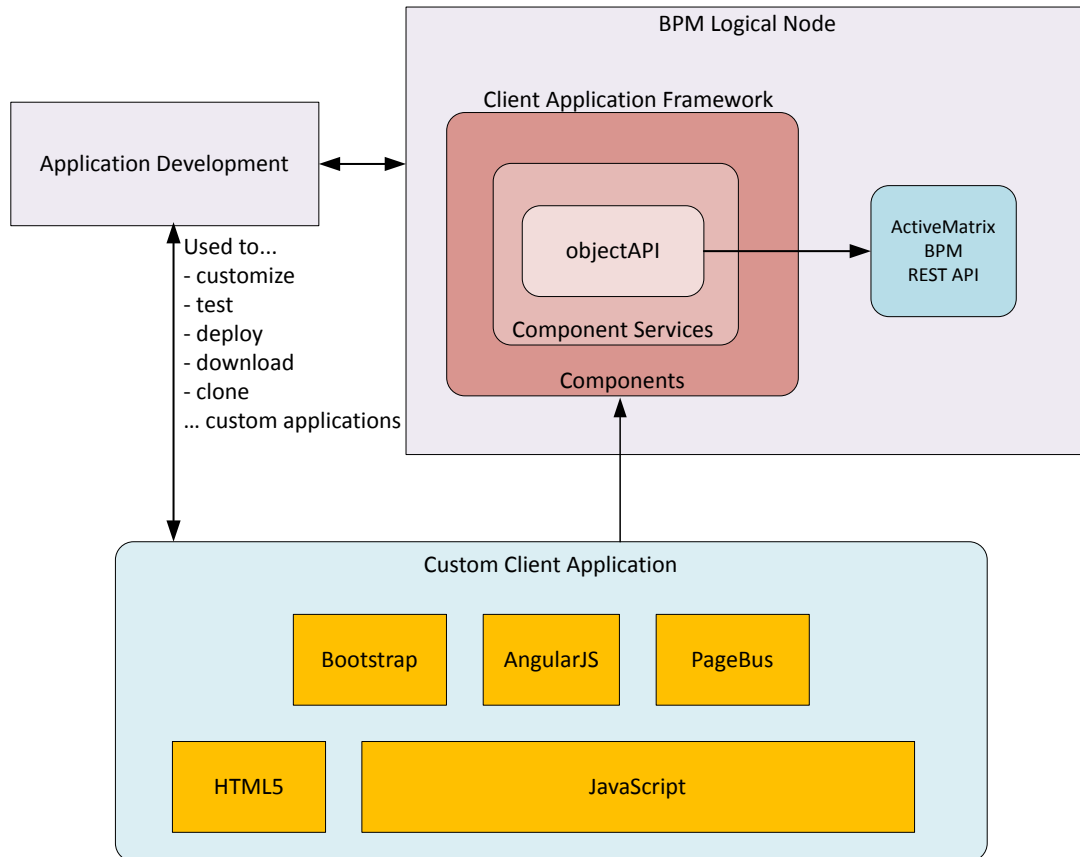
How to Join TIBCO Community

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

Overview of the Client Application Framework

The Client Application Framework is used to create and manage lightweight BPM applications using industry-standard technologies, such as Bootstrap CSS, AngularJS, and JavaScript.

The following illustrates the framework:



The Client Application Framework consists of, and uses, the following elements:

- Application Development** - This is a user interface to manage your custom client applications. Using Application Development, you can update, test, and deploy a custom applications.
 For information, see the *TIBCO ActiveMatrix® BPM Client Application Management Guide*.
- Business Components** - Business components are used to render BPM-related controls in an application. These can be used as building blocks to create a client application.
 For information, see [Business Components](#).
- Component Services** - Component services are used to perform BPM-related functions that would require several calls to the lower-level objectAPI. Component services are fully functional AngularJS services, and can be consumed by AngularJS clients.
 For more information, see [Component Services](#).
- objectAPI** - The objectAPI provides services that contain functions for all available ActiveMatrix BPM functionality. Typically, you will use business components and component services to build an application, as those require less effort. However, not *all* ActiveMatrix BPM functionality is available by using business components and component services. If that additional functionality is needed, the objectAPI can be used to access it.

For more information, see [objectAPI](#).

- **Custom Client Application** - This browser-based, Business Process Management (BPM) application can be created with technologies such as Bootstrap CSS and AngularJS. It can use business components to display BPM-specific user interfaces, component services to perform BPM-related functions, as well as make calls to the objectAPI services to perform BPM-specific functions.

For more information, see [Custom Client Application](#).

Custom Client Application

This is a browser-based, BPM application that makes calls to the Client Application Framework.

The custom client application can use the following industry-standard technologies:

- **HTML5** - Applications built using the Client Application Framework can make use of HTML5 capabilities.
- **JavaScript** - JavaScript developers can use the provided JavaScript library to invoke BPM services, which return JavaScript objects containing BPM and client application data.
- **JavaScript Framework:**
 - **AngularJS** - Client applications can be developed using AngularJS. Therefore, AngularJS modules are provided to perform BPM services. Plus, data in client applications are accessed using AngularJS expressions.
 - **Bootstrap** - A framework for developing responsive, mobile-first web sites. It includes HTML- and CSS-based templates for forms, buttons, tables, and so on. Web sites created using Bootstrap automatically adjust for all devices: mobile, tablets, and desktops. The example client applications that are provided (see [Example Applications](#)) use the Bootstrap grid system.
 - **PageBus** - All components available to client applications publish events when actions are initiated in the component (for example, a work item is selected in a work list). These events are available to client application developers by subscribing to events using TIBCO PageBus.

While custom client applications are being developed, they can be stored, tested, deployed (as a WAR file), and so on, using the Application Development Platform. For information, see the *TIBCO ActiveMatrix® BPM Client Application Management Guide*.

Custom client applications are *client resident* AJAX applications, which means that they consist entirely of static resources that are served to the browser, which hosts the applications. Client application do not include servlets nor client-side executables, such as applets or .NET libraries. Although they are typically deployed to a Web Components BPM Logical Node using the Application Development user interface, client applications created with the Client Application Framework can be deployed to any web server (for example, Apache Tomcat).



Application Development internally uses the Apache Jackrabbit database for repository management. There is no need for user interaction with this database, therefore, no user interface or command-line-interface is provided.

File and Folder Naming

When you create a custom client application, do not use any non-alphanumeric characters in file or folder names in the application.



It is good practice to use camelCase as your convention for application file and folder names.

Client Application Execution

After the client application is deployed, it can be served to a browser using the following URL:

```
http://Host:Port/contextRoot/apps/appName.html
```

where:

- *Host* is the name or IP address of the machine containing the BPM Logical Node.
- *Port* is the port number used by the WebApp Implementation Type to communicate with web applications.
- *contextRoot* is the root of applications deployed to the web server.
- *appName* is the name of the WAR file, containing the client application assets, that was deployed to the web server.

Example Applications

Several example applications are provided that illustrate the use of typical BPM-related user interfaces and functions.

The example applications are:

- **My Work Application** - This application demonstrates the integrated use of multiple [Business Components](#) and [Business Component Services](#) to provide the following:
 - **Log in / log out** - This shows standard behavior for logging in and out.
 - **Business actions** - Provides a list of the available "business actions," as well the ability to start business actions. (Business actions accomplish some sort of business function. For example, calling a business service to handle an incoming insurance claim.)
 - **Work lists** - A list of work items the logged-in user can work on, filtered and sorted as desired. From the work list, work items can be opened and progressed through the defined business action.
 - **Process instance lists** - A list of the process instances to which the logged-in user has access. Each process instance is a result of starting a business action.
 - **Event view** - An audit trail that show activity for a given work item, with the ability to navigate from there to the associated process instance.

This application is available as **Workapp** in the Application Development user interface.

- **Component Showcase Application** - This application showcases how to use individual [Business Components](#) and [Business Component Services](#) provided with the Client Application Framework.

This application is available as **Component Showcase** in the Application Development user interface.

- **Case Management Application** - This application provides a simple example of how you can combine different business components to develop feature-rich case management applications.

This application is available as **Case Management** in the Application Development user interface.

For information about accessing and modifying these example applications, see the *TIBCO ActiveMatrix BPM Client Application Management Guide*.

Bundled Applications

There are some bundled applications provided with Application Development. These applications demonstrate how to use specific Business Process Management (BPM) functionality. You can use the bundled applications in your own custom client applications to provide a specific BPM-related function, for example, opening a work item from a URL or starting a business service from a URL.

The bundled applications are browser-based applications that consist entirely of static resources (such as HTML, CSS, Javascript, XML and JSON), which are served to the browser that hosts the application. Applications do not include servlets or client-side executables, such as applets or .NET libraries.

You can use the bundled applications in your own custom client applications. You can also clone and customize the bundled applications, depending on your requirements. However, you cannot edit the bundled applications directly in Application Development.

Authentication

Your calling user must be logged into their TIBCO Active Matrix BPM runtime to access an application that uses bundled applications. If the calling user is not logged in, the Application Development Login screen is displayed. A valid BPM username and password can be added as part of the extended URL for a bundled application. The authentication details do not need to be the same as those of the user who is opening a work item, or starting a business service. However, you must make sure that the credentials you provide are appropriate. For example, for the `openworkitem` application, a valid BPM user who can access and open a work item and for the `startbizaction` application, a valid BPM user who can access and start a business service.

See [Using the openworkitem Application in Your Application](#) and [Using the startbizaction Application in Your Application](#).

Using the openworkitem Application in Your Application

The `openworkitem` application is a browser-based application that provides the ability to open a work item from a URL with a given set of parameters.

The HTML document to which you want to add the `openworkitem` application must add an HTML element to access the `openworkitem` application. The HTML element needs to define and set the query parameter information to be passed to the `openworkitem` application. It is up to you how you obtain the information that populates the query parameters.

The HTML element is an `<a>` tag with an `href` attribute that accesses the `openworkitem` application and specifies the ID of the work item that you want to open. The format is:

```
app_path?ID=workItemID[&version=versionNumber][&displayConfirmation=boolean]
[&username=username][&authType=password]
```

where:

Parameter	Description
<i>appPath</i>	This is the relative path to the <code>openworkitem</code> application. For example, <code>/apps/openworkitem/index.html</code>
<i>workItemID</i>	Unique ID of the work item.
<i>versionNumber</i> (optional)	Version of the work item. For example, 1.0.1
<i>displayConfirmation</i> (optional)	Boolean defining whether the end user receives confirmation after the work item is completed. If <i>displayConfirmation</i> is <code>false</code> , a work item is opened in its own window and when the work item is completed, the window is closed.
<i>userName</i> (optional)	A valid BPM runtime login. For example, tibco-admin

Parameter	Description
<i>authType</i> (optional)	Must be either: <ul style="list-style-type: none"> auth0 - the password of the BPM runtime login must be supplied in plain text. auth1 - the password of the BPM runtime login must be supplied as a Base64 encoded string. You must use a Base64 encoder to encode the password of the BPM runtime login.
<i>password</i>	Password of the BPM runtime login in either plain text or as a Base64 encoded string, depending on the specified <i>authType</i> .

Example

Your HTML page defines the `<a>` tag that accesses the `openworkitem` application and provides the query parameter information to be passed to the URL.

```
<a href="/apps/openworkitem/index.html?
id=1&version=1.0.1&displayConfirmation=true&username=tibco-
admin&auth1=cGFzc3dvcmQ"></a>
```

Using the startbization Application in Your Application

The `startbization` application is a browser-based application that provides the ability to start a business service from a URL.

The HTML document to which you want to add the `startbization` application must add an HTML element to access the `startbization` application. The HTML element needs to define and set the query parameter information to be passed to the `startbization` application. It is up to you how you obtain the information that populates the query parameters.

The HTML element is an `<a>` tag with an `href` attribute that accesses the `startbization` application and specifies the business service that you want to start. The format is:

```
appPath?
module=moduleName&process=processName&version=versionNumber[&payload=payload]
[&displayConfirmation=boolean][&username=username][&authType=password]
```

where:

Parameter	Description
<i>appPath</i>	This is the relative path to the <code>startbization</code> application.
<i>moduleName</i>	Name of the parent module that contains the business service.
<i>processName</i>	Name of the business service to be started.
<i>version</i>	Version number of the business service.
<i>payload</i> (optional)	Page data for the formal parameters associated with the business service. You cannot determine the names of these formal parameters programmatically. Instead, you must obtain them by examining the business service in TIBCO Business Studio.

Parameter	Description
<code>displayConfirmation</code>	Boolean defining whether the end user receives confirmation after the business service is completed. If <code>displayConfirmation</code> is <code>false</code> , a business service is opened in its own window and when the business service is completed, the window is closed.
<code>userName</code> (optional)	a valid BPM runtime login.
<code>authType</code> (optional)	must be either: <ul style="list-style-type: none"> <code>auth0</code> - the password of the BPM runtime login must be supplied in plain text. <code>auth1</code> - the password of the BPM runtime login must be supplied as a Base64 encoded string. You must use a Base64 encoder to encode the password of the BPM runtime login.
<code>password</code>	the password of the BPM runtime login in either plain text or as a Base64 encoded string, depending on the specified <code>authType</code> .

Example

Your HTML page defines the `<a>` tag that accesses the `startbizaction` application and provides the query parameter information of the business services you want to start.

```
<a href=/apps/startbizaction/index.html?module=/XYZInsurance_Process/Process
Packages/
XYZInsurance_Process.xpdl&process=InitiateClaimReportIncident&version=1.0.1&displayC
onfirmation=true&username=tibco-admin&auth1=cGFzc3dvcmQ</a>
```

Hosting Applications on External Web Servers

Custom client applications are functionally complete, standard J2EE web applications, packaged as WAR files. Although they are typically deployed to a Web Components BPM logical node, they can be deployed to other web servers.

You can deploy custom client applications to the following web servers:

- Jetty
- Apache Tomcat
- IBM WebSphere
- Red Hat JBoss Web Server or Wildfly

Use **Application Development** to export the application as a ZIP file, then deploy it to your chosen web server.

If you want to host a custom client application on an external web server, there are a number of issues you need to take account of.

SSO

When an application is hosted outside of ActiveMatrix BPM, it has no inherent SSO capabilities. You must implement your own SSO mechanisms. As long as these conform to the SSO capabilities provided by ActiveMatrix BPM, the application will be able to participate in SSO.

Cross-Domain Calls

If an application hosted on an external server is in a different domain relative to the ActiveMatrix BPM services API, any service invocation from the custom client will encounter Same Origin Policy (SOP)

violations. To overcome this issue, you must provide a suitable proxy that can be used to proxy the service calls to ActiveMatrix BPM.

In addition to the usual proxy settings required, for example, configuring proxy modules, you must amend the `httpd.conf`. The following steps describe how to configure a proxy for Apache HTTP Server Version 2.4.

1. Install Apache HTTP Server Version 2.4 with the modules required for proxy servlets. See the documentation provided with Apache HTTP Server Version 2.4.
2. Add the following to the `httpd.conf` file:

```
<Location /apps/>
ProxyPass http://10.100.87.51:8080/apps/ nocanon
</Location>
ProxyPass "/bpm/" "http://10.100.87.51:8080/bpm/"
ProxyPass "/bpmresources/" "http://10.100.87.51:8080/bpmresources/"
ProxyPassReverse / http://10.100.87.51:8080/
AllowEncodedSlashes NoDecode
```

The REST URL that ActiveMatrix BPM uses for translations contains encoded characters in the URL. Therefore, you must use:

- the `nocanon` argument for the `ProxyPass` directive used for `/apps/` path.
- the `AllowEncodedSlashes` directive.

See the documentation provided with Apache HTTP Server Version 4.2 for more information.

3. Export your custom client application from ActiveMatrix BPM Application Development. See the *TIBCO ActiveMatrix® BPM Client Application Management Guide*.
4. Extract your application from `<apache_config>\htdocs\appName.app` where `apache_config` is the Apache HTTP Server configuration directory. See the documentation provided with Apache HTTP Server Version 4.2 for more information.
5. Launch the URL for your custom client application.

Using a Load Balancer

If you want use a load balancer, add the following to the `httpd.conf` file:

```
<Location /apps/>
ProxyPass http://<bpmserver1>:<port>/apps/ nocanon
ProxyPass http://<bpmserver2>:<port>/apps/ nocanon
</Location>

ProxyPass "/bpm/" "http://<bpmserver1>:<port>/bpm/"
ProxyPass "/bpmresources/" "http://<bpmserver1>:<port>/bpmresources/"
ProxyPassReverse / http://<bpmserver1>:<port>/

ProxyPass "/bpm/" "http://<bpmserver2>:<port>/bpm/"
ProxyPass "/bpmresources/" "http://<bpmserver2>:<port>:8080/bpmresources/"
ProxyPassReverse / http://<bpmserver2>:<port>/

ProxyPreserveHost On
AllowEncodedSlashes NoDecode
```

where `bpmserver1` and `bpmserver2` are the IP addresses of the servers running AMX BPM.



If you are using Apache 2.2, set `ProxyPreserveHost` to `On` which makes it preserve the host header when passing through the proxy. If the header is not set, it causes issues when accessing applications. See http://httpd.apache.org/docs/2.2/mod/mod_proxy.html#proxypreservehost.

Localization

If you have localized your application, the localized resource bundles must be deployed as a separate application and then you must register your application. See "Localizing Applications" in *TIBCO ActiveMatrix® BPM Client Application Management Guide*.

Then, register your new application:

```
BPMTranslationServiceConfig.registerApp("applicationname", "MyPrefix"), null);
```

where

- *applicationname* is the name of the application that contains the resource bundles.
- *MyPrefix* is the application prefix that you want to use in the templates.

For example, if you have registered an application with the name *MyPrefix*, and you have a *MyComponentMessages.properties* file in the *l10n* folder of the application that contains the resource bundles with *myKey*, then you can refer to this key in your custom client application's angular templates as *MyPrefix.MyComponent.myKey*. For example,

```
<div translate="MyPrefix.MyComponent.myKey"></div>
```

Configuration Changes

Any configuration changes required can be done by manually editing the application's configuration files.

Configure a Theme for a Custom Client Application

Custom client applications developed with the Client Application Framework use TIBCO specific themes to define the colors used by the application.

An internal `tibTheming` service uses the AngularJS Material `$mdTheming` service to provide and manage themes.

There are two TIBCO specific themes:

- All applications use the Default theme.
- The Workapp application also uses the Aquamarine theme.



It is not possible to modify the Aquamarine theme or create or add a new theme.

There is a [Theme_Customization_Example](#) application that demonstrates how to use the `tibThemingProvider` function to configure how an application uses the Default theme. This application changes the colors for a clone of the Workapp application.

In Application Development, open the `MyWorkCtrl.js` file in the `Theme_Customization_Example` application. The `config` phase on the application controller module defines the `tibThemingProvider` function. The function does the following:

1. It overrides the existing default theme with a set of custom palettes.
2. It defines the colors used by those palettes.
3. It enables watching for theme changes.

```
app.config(["tibThemingProvider", function(tibThemingProvider) {

    tibThemingProvider.theme('default') // Overriding existing 'default' theme
                                        // with the following custom theme.

    .
    .
    tibThemingProvider.definePalette('customPaletteName-primary', {
    '50': '#d9001e', // for user profile background
    .
    .
    tibThemingProvider.alwaysWatchTheme(true);

}]);
```

See [Theme Color Definitions](#)



Do not delete the palettes: `customPaletteName-accent`, `customPaletteName-warn` and `customPaletteName-background`.

The `tibThemingProvider` function contains color definitions for these palettes, however, the Client Application Framework themes do not use them, and any changes to their color values has no effect. These palette definitions must be present and complete, otherwise the `$mdTheming` service will reject the theme as invalid during registration.

Procedure

1. Copy the example `tibThemingProvider` function to the application controller file, within the `config` phase on the application controller module.
2. Change any of the following color definitions in the `customPaletteName-primary` palette.
3. Save the changes.

Use **PREVIEW** or **LAUNCH LOCAL** to test the app.

Theme Color Definitions

These are the theme color definitions for the `tibThemingProvider` function. AngularJS Material defines the color values, see the AngularJS Material Theming documentation for more information.

Color	Used by components	Defines
50	<code><user-profile></code>	Background color of the user profile displayed in the left-hand navigation pane.
100	<code><bpm-business-actions-drop-down></code>	Background color of the BUSINESS ACTIONS menu item displayed in the left-hand navigation pane.
200	None	Not used. Do not change.
300	<code><bpm-work-views></code>	Background color of the WORK VIEWS menu item displayed in the left-hand navigation pane.
400	<code><bpm-work-list></code>	Font, icon and button color used for the "Work Views" header.
	<code>-> <bpm-work-item-card></code>	Font color of the menu displayed when a user hovers over a work item card in the worklist.
		Menu background color for a selected work item in the worklist.
500	<code><bpm-work-list></code>	Background colour used when hovering on the attribute list displayed when the user clicks "More Info" on a work item card.
	<code>-> <bpm-work-item-card></code>	
	<code>-> -> <bpm-work-item-attr></code>	

Color	Used by components	Defines
600	<bpm-work-views>	Background color of the selected work list displayed in the WORK VIEWS menu.
700	None	Not used. Do not change.
800	None	Not used. Do not change.
900	<documents-view>	Background color of the table header displayed on the DOCUMENTS tab, when a user clicks DETAILS for a work item.
A100	<bpm-work-list> -> <bpm-work-item-card>	Background color of the menu displayed when a user hovers over a work item card in the worklist.
A200	None	Not used. Do not change.
A400	<documents-view>	Background color of even-numbered items displayed in the table on the DOCUMENTS tab, when a user clicks DETAILS for a work item.
A700	None	Not used. Do not change.

Internationalizing Applications

The *internationalization*, i18n, of applications created in the Client Application Framework provides *localization*, l10n, to the desired language.

After an application is internationalized, you can contribute language packs to the application using the Application Development user interface. The set of message properties files used by the Client Application Framework components can be found under the l10n folder of the application in the Application Development user interface. For more information, see the "Localizing Applications" topic in the *TIBCO ActiveMatrix® BPM Client Application Management Guide*.

The way in which you internationalize an application depends on whether the application was cloned from the Workapp sample application, or it is a custom application that was created using the component services, components, and object API available in the Client Application Framework. For information, see:

- [Internationalizing a Cloned Workapp Application](#)
- [Internationalizing a Custom Application](#)

Internationalizing a Cloned Workapp Application

Applications that have been cloned from the Workapp sample application can be internationalized using the procedure provided here.

Procedure

1. Select your custom (cloned) application in the Application Development user interface, then click **Open**.
2. Open the main controller file, `MyWorkCtrl.js`.

3. Locate the following comments and code, then uncomment the last three lines.

```
// NOTE: After cloning the app, if you want to use your own L10N bundle,
// uncomment the following code and pass the right app name
// app.run(['BPMTranslationServiceConfig', function(BPMTranslationServiceConfig)
// {
//     BPMTranslationServiceConfig.registerApp("MyApp_name_here", null);
// }]);
```

4. Add a call to unregister the sample Workapp application.

The sample Workapp application is registered by default. Unregistering it prevent bundles from being loaded from the Workapp application. The code should now look like this:

```
// NOTE: After cloning the app, if you want to use your own L10N bundle,
// uncomment the following code and pass the right app name
app.run(['BPMTranslationServiceConfig', function(BPMTranslationServiceConfig) {
    BPMTranslationServiceConfig.unregisterApp("workapp");
    BPMTranslationServiceConfig.registerApp("MyApp_name_here", null);
}]);
```

5. In the code you uncommented in [Step 3](#), change "MyApp_name_here" to the name of your custom application as deployed in Application Development.
For example:

```
// NOTE: After cloning the app, if you want to use your own L10N bundle,
// uncomment the following code and pass the right app name
app.run(['BPMTranslationServiceConfig', function(BPMTranslationServiceConfig) {
    BPMTranslationServiceConfig.registerApp("MyCustomApp", null);
}]);
```

6. Save the changes to the MyWorkCtrl.js file.

Result

Now when your application is run, the available locales will be listed in the languages field in the upper-right part of the application (provided you have localized the application by adding the language packs).

For information about editing phrases for an available locale, or adding a new language pack, see the "Localizing Applications" topic in the *TIBCO ActiveMatrix® BPM Client Application Management Guide*.

Internationalizing a Custom Application

This section describes internationalizing a custom application that uses the TIBCO Component Services Framework.

The TIBCO Component Services Framework is loaded using `tcf.nocache.js`. For additional information on including the framework in your custom application, see [Business Component Services](#).

Internationalizing your custom application involves using the internal [BPMTranslationService](#), which is configurable using [BPMTranslationServiceConfig](#), an Angular service that uses angular-translate with a custom loader to load the translation data. All directives, services, filters, and syntax supported by angular-translate can also be used in clients developed in the TIBCO Component Services Framework.

Procedure

1. Register your application to the translation service using [BPMTranslationServiceConfig.registerApp](#), which allows the TranslationService to load your message properties files.

```
var yourModule = angular.module("CustomModule", ..
..
yourModule.run(['BPMTranslationServiceConfig', function(BPMTranslationServiceConfig){
    BPMTranslationServiceConfig.registerApp("MyCustomApp", null);
}]);
```

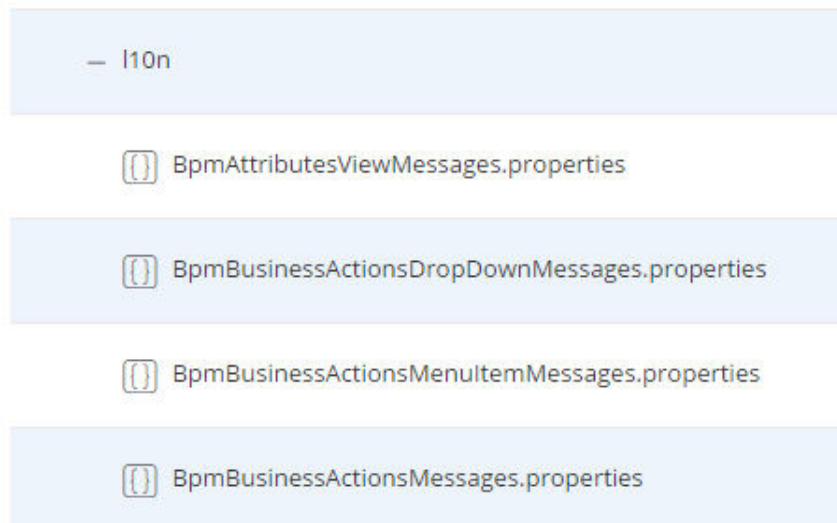
In this example, the Angular module name is 'CustomModule' and the application name as deployed in Application Development is 'MyCustomApp'.

The second parameter in the `registerApp` function specifies the prefix to be used for accessing the translation keys. If null is passed, the prefix defaults to 'tcf'.

2. In the application's `l10n` folder, create message properties files that contain the key / value pairs for localized strings.

All property file names start with a component name and end with "Messages", and have an extension of `properties`. For example, `BpmBusinessActionsMessages.properties`, which means that the component name is `BpmBusinessActions`, and the keys in the file can be accessed using `tcf.BpmBusinessActions.<key_name>`.

For example, in the Workapp sample application, there is a message properties file for each component used in the sample application:



3. Access a translation key, from a message properties file, in a template as follows:

```
<div translate="pfx.component.translationKey"></div>
```

where:

- *pfx* - is the prefix used when referencing message properties files. This must be the value of the second parameter passed in the `registerApp` function (see [Step 1](#)). If null is passed, this must be 'tcf'.
- *component* - is the business component displayed by the template. This is the name of the property file in the `l10n` folder of the application, excluding the 'Messages' suffix.
- *translationKey* - is the key part of the key / value pair in the message properties file.

For example, if you have a `BpmBusinessActionsMessages.properties` file that contains the following key / value pairs...

```
search_holder=Find a Business Action
loading_tips>Loading...
reload_text=RELOAD
reloading_text=RELOADING...
```

... your template can access the "reload_text key as follows:

```
<div class="refresh-button" ng-click="reloadCategories()" ng-
hide="reloadingCategories"
  translate="tcf.BpmBusinessActions.reload_text"
  translate-default="RELOAD">RELOAD</div>
```

This assumes the prefix is the default 'tcf'. The 'tcf' prefix can also be accessed in a more generic way in the templates, as follows:

```
<div class="refresh-button" ng-click="reloadCategories()" ng-
hide="reloadingCategories"
  translate="{{ $root.appConfig.appId }}.BpmBusinessActions.reload_text"
  translate-default="RELOAD">RELOAD</div>
```

This requires that the `$root.appConfig.appId` variable be set in the main controller (for example, `MyWorkCtrl.js`) in the following way:

```
$rootScope.appConfig = {
  appId: 'tcf'
};
```

Result

You can now access the locale specified by the message properties files in your running custom application.

For information about editing phrases for an available locale, or adding a new language pack, see the "Localizing Applications" topic in the *TIBCO ActiveMatrix® BPM Client Application Management Guide*.

BPMTranslationService

The `BPMTranslationService` is an internal service in TIBCO Component Services Framework that is responsible for loading and management of resource bundles.

The `BPMTranslationService` is configurable using the Angular service, [BPMTranslationServiceConfig](#).

LocaleInfoModel

`LocaleInfoModel` is the object model used by the [BPMTranslationServiceConfig](#) to pass information about locales when [getLocaleInfo](#) is called.

Attributes

Name	Type	Description
language	String	The localized language, for example, English.
locale	String	The locale for the language, for example, en_US.
country	String	The localized country text, for example, United States.

BPMTranslationServiceConfig

`BPMTranslationServiceConfig` is an Angular service provided in the Client Application Framework to configure the built-in translation service.

This service can be injected, and has the following functions:

- [registerApp](#) - Registers the application with the translation service.
- [unregisterApp](#) - Unregisters the application.
- [getLocaleInfo](#) - Retrieves a list of locales.

getLocaleInfo

Retrieves the list of locales for which translations are available.

Required System Actions

None

Usage

```
BPMTranslationServiceConfig.getLocaleInfo(callback)
```

Parameters

Parameter	Type	Description
<i>callback</i>	Array<LocaleInfo Model>	A callback handler whose localeInfo(Array<LocaleInfoModel>) method is invoked with the array of LocaleInfo objects. An array of locales is returned on the callback object.

Sample Usage

```
BPMTranslationServiceConfig.getLocaleInfo({
  localeInfo: function(localeInfo) {
    localeInfo.forEach(function(lang) {
      // retrieve lang.language, lang.locale or lang.country from the
      available list
    });
  }
});
```

registerApp

Registers the application in the translation service.

Required System Actions

None

Usage

```
BPMTranslationServiceConfig.registerApp(appId, pfx)
```

Parameters

Parameter	Type	Description
<i>appId</i>	String	The name of the application, as deployed in Application Development, that contributes the property bundles.
<i>pfx</i>	String	The prefix to be used to access the translation keys. This can be passed as null, which causes it to default to a prefix of 'tcf'.

Sample Usage

```
var yourModule = angular.module("CustomModule", ..
..
yourModule.run(['BPMTranslationServiceConfig',function(BPMTranslationServiceConfig){
  BPMTranslationServiceConfig.registerApp("MyCustomApp", null);
}]);
```

unregisterApp

Unregisters the specified application from the translation service.

Required System Actions

None

Usage

```
BPMTranslationServiceConfig.unregisterApp(appId)
```

Parameters

Parameter	Type	Description
<i>appId</i>	String	The name of the already-registered application, as deployed in Application Development, to be unregistered.

Sample Usage

```
var yourModule = angular.module("CustomModule", ..
..
yourModule.run(['BPMTranslationServiceConfig',function(BPMTranslationServiceConfig){
  BPMTranslationServiceConfig.unregisterApp("MyCustomApp");
}]);
```

App CDN

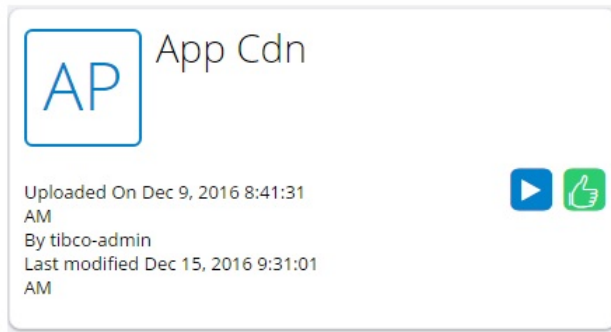
Application Development includes a library application (**App Cdn**), which provides static libraries that can be used by all applications. This eliminates the need to manage libraries on a per-application basis.

The primary advantage of using App CDN (Content Development Network) is that it provides central loading of third-party libraries, rather than being duplicated per application. It also results in improved loading time; once a library has been referenced by a browser, it is cached so that it can be re-used by subsequent calls (by the same, or different, application).

The App Cdn application is hidden by default. To make it visible:

1. Access the **Permissions** tab in Application Development.
2. Change the permission for App Cdn to "Full Access".

The library application now appears in the **Applications** list:



Opening the App Cdn application lists all of the libraries that are included in the application:

Select	File Name
<input type="checkbox"/>	app_cdn.app.desc.json
<input type="checkbox"/>	+ ace
<input type="checkbox"/>	+ angular
<input type="checkbox"/>	+ angular_animate
<input type="checkbox"/>	+ angular_aria
<input type="checkbox"/>	+ angular_filter
<input type="checkbox"/>	+ angular_material

You can expand the libraries to view details, such as the specific version.

Referencing Libraries in App Cdn

If your application uses the functions available in the libraries in App Cdn, you must include the libraries in the HTML document as follows:

```
<script type="text/javascript" src="/apps/app-cdn/tcf/tcf.nocache.js"></script>
<script type="text/javascript" src="/apps/app-cdn/jquery/jquery-3.1.1.js"></script>
<script src="/apps/app-cdn/angular/1.6.0/angular.min.js"></script>
<script src="/apps/app-cdn/angular-ui-grid/ui-grid.min.js"></script>
```

For additional information about including the libraries in App Cdn, see:

- [Adding a Business Component to Your Application](#)
- [Invoking a Business Component Service Operation From Your Application](#)

Business Components

Business components are used to render BPM-related controls in an application. These can be used as building blocks to create a client application.

Business components are BPM-specific components, implemented as AngularJS directives, that can be embedded in an application as HTML elements.

Each business component:

- provides a complete piece of BPM-related functionality - for example, displaying a user's work list and providing controls to manipulate that list and the work items within it - which you can add to your application simply by adding an HTML element to a page.
- has one or more attributes that can be used to pass in data and configuration information.



Some attributes are mandatory - without them, the business component will not render. If these attributes are constructed with empty objects, the business component invokes the appropriate component service to fetch and render the content.

- uses a default, internal HTML template to render its content. A copy of this template is also supplied as part of the My Work App application. You can modify this template to suit your requirements and then use it in place of the default template. See [Customizing a Business Component](#).
- has an inbuilt controller that implements the component logic (using component services). Controllers are supplied as part of the TIBCO Component Framework library (`tcf.nocache.js`). They must be included in your application, but cannot be modified.

Component Showcase Application

The Component Showcase application demonstrates the use of individual business components and component services.

Component Showcase uses the same business components and component services as My Work App. However, whereas My Work App presents an integrated, fully functional client application, Component Showcase breaks that functionality down into a number of stand alone examples, which you can use to more easily understand how they work, or as the basis for your own development.

Each menu item in Component Showcase demonstrates the use of a particular business component or component service (which may in turn invoke others), as shown.

Menu item	Description	Demonstrates the use of the following...	
		Business Components	Component Services
START BIZ ACTION	Enter the details for a business action and start it. View the form and Cancel or Submit it. A tabbed display allows you to have multiple open business actions.		BPMBusinessActionsService
OPEN WORK ITEM	Enter the ID of a work item and open it. View the form and cancel, close or submit it. A tabbed display allows you to have multiple open work items.		BPMWorklistService.openWorkItem operation

		Demonstrates the use of the following...	
Menu item	Description	Business Components	Component Services
WORK LIST	<ul style="list-style-type: none"> • Display the contents of your work list as cards, a list or a table. • Sort the work list by priority, process or date. • Search the work list. • Open a work item and cancel, close or submit it to return to the work list. • Select one or more work items, giving you further options to open, view details for, allocate or re-offer. 	work-list	
		bpm-event-view	
		bpm-work-item-history	
WORK VIEWS	Display your available work views as an accordion control. Select a work view to see its details (without opening that work view).	work-views	
BUSINESS ACTIONS	Display your available business actions as an accordion control. Select a business action and start it. View the form and Cancel or Submit it.	business-actions-drop-down	
CASE DOCUMENTS	Enter either a case reference or a work item ID to display a list of the associated case documents.	documents-view	
CASE ACTIONS	Enter a case reference to display a list of the available case actions.	bpm-case-actions	

Menu item	Description	Demonstrates the use of the following...	
		Business Components	Component Services
WORK ITEM DETAILS	<p>Enter a work item ID to populate the tabbed list with the relevant information for that work item. The tabs are:</p> <ul style="list-style-type: none"> Summary - provides summary information about the work item. Data - shows the current data values for the work item. Attributes - lists the names and current value of every work item attribute defined for the work item. Processes - provides information about the process that generated the work item. Documents - displays the case documents associated with the work item. 	<p>work-item-detail-nav, which in turn uses:</p> <ul style="list-style-type: none"> bpm-summary-view work-item-data bpm-attributes-view bpm-process-view documents-view 	

Component Showcase is available in the Application Development UI. You can browse through it there, or export it.

BUSINESS ACTIONS

The BUSINESS ACTIONS tab of the Component Showcase application demonstrates the use of the `<bpm-business-actions-drop-down>` business component.

To view or edit the source files for this example, open the Component Showcase application and browse to the `componentDemos/bizActions` folder.

BusinessActions.html

The page:

- displays the [bpm-business-actions-drop-down](#) business component, which lists the available business actions in an accordion control. You can use the control to select and start a business action.

```
<!-- Left Navigation Content -->
<md-sidenav md-is-locked-open="$mdMedia('gt-sm')" class="appTraySideNav md-whiteframe-z2" md-component-id="left" layout="row">
  <business-actions-drop-down actions-count="currActionsCount" set-main-content="setMainContent" form-div="formDiv" action-groups="actionGroups" expand-actions="expandActions"></business-actions-drop-down>
</md-sidenav>
```

- defines an element to be used to display a started business action form.

```
<!-- MAIN CONTENT -->
<md-content flex id="content" layout="row">
  <div class="mainContent" flex>
    <div ng-show="isMainContentShowingForm()" id="{{formDiv}}"></div>
```

```
</div>
</md-content>
```

js/BizActionSample.js

The BizActionSampleCtrl controller defines the parameters that need to be passed to the [bpm-business-actions-drop-down](#) business component on the \$scope object:

```
$scope.currActionsCount = 0;
$scope.actionGroups = [];
$scope.expandActions = true;
$scope.setMainContent = function(content) {
    $scope.main_content = content;
};
$scope.isMainContentShowingForm = function() {
    return ($scope.main_content == "form");
};
$scope.formDiv = "formAddDiv";
```

START BIZ ACTION

The ACTION tab of the Component Showcase application demonstrates the use of the startBizAction operation from the BPMBusinessActionsService component service.

To view or edit the source files for this example, open the Component Showcase application and browse to the componentDemos/startBizAction folder.

StartBusinessActions.html

The page:

- displays a form in which you can enter and submit the details of the business action that you want to start.
- defines an element to be used to display the started business action form.

```
<div flex="60">
    <md-tabs flex md-selected="selectedIndex" md-border-bottom md-autoselect=""
md-dynamic-height>
        <md-tab ng-repeat="bizActionFormTab in bizActionFormTabs" ng-
disabled="bizActionFormTab.disabled"
label="{{bizActionFormTab.businessActionItemData.processName}}">
            <div class="demo-tab tab{{ $index%4 }}">
                <div id="{{bizActionFormTab.formDiv}}" style="overflow:auto;"></
div>
            </div>
        </md-tab>
    </md-tabs>
</div>
```

js/StartBizActionSample.js

The StartBizActionSampleCtrl controller:

1. uses the supplied parameters to create a businessActionTemplate object that contains the data required by the [startBizAction](#) operation.

```
$scope.startBizAction = function(){
    for(i=0;i<$scope.bizActionFormTabs.length;i++)
    {
        if($scope.bizActionFormTabs[i].businessActionItemData.processName==
$scope.processName)
            return;
    }
    var businessActionTemplate = {};
    businessActionTemplate.businessActionItemData = {};
```

```

        businessActionTemplate.businessActionItemData.moduleName =
$scope.moduleName;
        businessActionTemplate.businessActionItemData.processName =
$scope.processName;
        businessActionTemplate.businessActionItemData.version = $scope.version;
        businessActionTemplate.payload = $scope.payload;

        businessActionTemplate.formDiv = "formBizDiv#" + $scope.processName;

        $scope.bizActionFormTabs.push(businessActionTemplate);

```

2. calls the [startBizAction](#) operation, passing the businessActionTemplate in the request.

```

    if (BPMBusinessActionsService){
        BPMBusinessActionsService.startBizAction(businessActionTemplate,
{onSuccess: function(){

$scope.bizActionFormTabs.splice($scope.bizActionFormTabs.indexOf(businessActionTe
mplate),1);

        $scope.$apply("");
    },
    onFailure: function(){

$scope.bizActionFormTabs.splice($scope.bizActionFormTabs.indexOf(businessActionTe
mplate),1);

        $scope.$apply("");
    }
});
}

```

3. uses the callback function of the request to render the business action form in the <div> element identified by businessActionTemplate.formDiv.

WORK ITEM DETAILS

The WORK ITEM DETAILS tab of the Component Showcase application demonstrates the use of the <work-item-detail-nav> business component.

To view or edit the source files for this example, open the Component Showcase application and browse to the componentDemos/workitemDetails folder.

DetailsViews.html

The page displays:

- a form in which you can enter and submit a work item ID.

```

<div flex="30" layout="column">
<form name="sampleForm" ng-submit="show()">
    <md-input-container>
        <label>Id:</label>
        <input required name="id" ng-model="id">
        <div ng-messages="sampleForm.id.$error">
            <div ng-message="required">This is required.</div>
        </div>
    </md-input-container>
    <md-input-container>
        <label>Version:</label>
        <input name="version" ng-model="version">
    </md-input-container>
    <md-button ng-disabled="sampleForm.$invalid" type="submit" class="md-raised
md-primary">Show Details</md-button>
</form>
</div>

```

- the [work-item-detail-nav](#) Business component, which is then used to display the appropriate data for the selected work item.

```

<div flex="60">
    <md-content flex layout-padding>
        <work-item-detail-nav tabs="tabs" selected-work-
item="selectedWorkItem"></work-item-detail-nav>

```

```
</md-content>
</div>
```



[work-item-detail-nav](#) invokes further business components and component services to display the appropriate data for each tab.

js/DetailsViewsSample.js

The DetailsViewSampleCtrl controller defines:

- a `tabs` object containing the tab titles to be passed to the [work-item-detail-nav](#) business component.

```
$scope.tabs = [ {
  title : 'Summary',
  content : ""
}, {
  title : 'Data',
  content : ""
}, {
  title : 'Attributes',
  content : ""
}, {
  title : 'Processes',
  content : ""
}, {
  title : 'Documents',
  content : ""
} ];
```

- a `show` function that is invoked when you click **SHOW DETAILS**. The function calls the [BPMWorkItemDetailsService.fetchWorkItem](#) operation to pass details of the selected work item to the [work-item-detail-nav](#) business component.

```
$scope.show = function() {
  BPMLoginService.getExtendedUserInfo({
    onSuccess : function(userInfo) {
      $scope.guid = userInfo.guid;
      if (BPMWorkItemDetailsService) {
        $scope.wi = BPMWorkItemDetailsService.fetchWorkItem($scope, $scope.id,
$scope.version, {
          onSuccess : function() {
            $scope.$apply("");
          },
          onFailure : function() {
            $scope.$apply("");
          }
        });
      }
    }
  });
};
```

Case Management Application

The Case Management application demonstrates how to combine different business components and develop feature-rich case management applications.



The Case Management application requires the details of a case model that is available on the ActiveMatrix BPM system. Edit the `CaseManagement.config.json` file to add this information, see [Configure the Case Management Application](#).

Log on to the Case Management application, and it displays a left sidebar with three navigation buttons, that open the following business components:

- **Business Action** calls the [<bpm-business-actions-drop-down>](#) business component. This lists the available business actions. Use the control to select and start a business action.
- **Cases** calls the [<bpm-case-types>](#) business component. This lists the available case types for the current case model. By default, the component also selects the first case type, and displays its cases.

- **Work Items** calls the `<bpm-work-list>` business component. This lists the work items available to the user.

Click one of the available Case Types calls the `<bpm-case-instance-summary>` business component. This lists all of the active cases for that Case Type.

Click one of the available Cases calls the `<bpm-case-details-page>` business component. This includes an information pane, and a right sidebar. Toggle the right sidebar to open a **Case Menu**.

The information pane includes the following items:

- A summary of the state of the case calls the `<bpm-case-states>` business component.
- Any available case actions calls the `<bpm-case-actions>` business component.
- The read-only case data calls the `<bpm-case-data>` business component.

The right sidebar contains navigation buttons, and the **Case Menu** contains menu items that open the following business components:

- **Case Overview** calls three business components: `<bpm-case-states>`, `<bpm-case-actions>`, and `<bpm-case-data>`. These provide the state of the case, read only case data, and case actions.
- **Case Work Items** calls the `<bpm-linked-work-items>` business component. Access work items linked to the case, applicable for the current user.
- **Linked Cases** calls the `<bpm-linked-case-instances>` business component. This provides links to any other case linked to this case.
- **Case Documents** calls two business components: `<bpm-document-upload>`, and `<documents-view>`. This provides a list of documents for the case. Upload a new document to the case. Select a document provides options to view, delete or download that document.
- **Case Comments** calls the `<bpm-comments>` business component. Add a new comment, and review the comment history for the case.
- **Audit** calls the `<bpm-case-event-view>` business component. Review the audit history for the case.

Configure the Case Management Application

The Case Management application requires the details of a case model that is available on the ActiveMatrix BPM system. Edit the `caseMgtAppName.config.json` file to add this information.

`caseMgtAppName` is the application name. For the Case Management application this is `CaseManagement`. For a clone of the Case Management application it is the name chosen for the cloned application.

When the Case Management application opens, the `index.html` file calls the `<bpm-case-types>` business component to display the available case types for the current case model. The `caseMgtAppName.config.json` file defines the current model.

This file contains the JSON definition for the default `selectedCaseModel` object. For information about this object, see [CaseModelBasicInfoType](#).

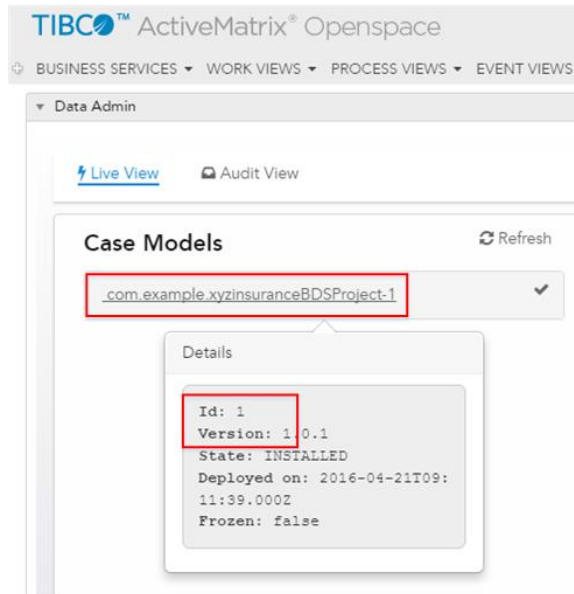
```
{
  "selectedCaseModel":
  { "caseModelID": "1", "appName": "com.example.customer-1", "majorVersion": "1" }
}
```

Prerequisites

There are three alternatives to obtain the required information for the case model from the ActiveMatrix BPM system:

- From the ActiveMatrix BPM web service API, use the `getCaseModelBasicInfo` operation, from the `BusinessDataServicesAdminService`.

- From the objectAPI, use the `GlobaldataAdminService.getCaseModelBasicInfo` method.
- From Openspace, use the **DATA ADMIN** pane. For example:



Procedure

1. Open the `caseMgtAppName.config.json` file.
2. Edit the `selectedCaseModel` definition, and replace the default details with those of the required case model.

```
{
  "selectedCaseModel":
  {"caseModelID": "1", "appName": "com.example.customer-1", "majorVersion": "1"}
}
```

3. Do the following for a clone of the Case Management application.

This is not required for the Case Management application.

- a) Open the `js/CaseManagement.js` file.
- b) Edit the `$http.get` call, and replace `CaseManagement.config.json` with the `caseMgtAppName.config.json` filename.

```
$http.get("CaseManagement.config.json")
  .then(function(response) {
```

4. Save the changes.
5. **Launch** the application to test it.
Remember to launch the **Latest** version.

Mobile Case Manager Application

The Mobile Case Manager application demonstrates how to combine different mobile business components and develop feature-rich case management applications for use with TIBCO® Mobilespace.



The Mobile Case Manager application requires the details of a case model that is available on the ActiveMatrix BPM system. Edit the `MobileCaseManager.config.json` file to add this information, see [Configure the Mobile Case Manager Application](#).

Log on to TIBCO® Mobilespace and open the Mobile Case Manager application, and it opens the Case Types window. There is a **Mobilespace** menu with two items:

- **Case Types** calls the `<mbpm-case-types>` business component. This lists the available case types for the current case model. For returning users, the component selects the case type that the user previously had open, and displays its cases.
- **Business Actions** calls the `<mbpm-case-creator>` business component. This lists the available business actions. Use the control to select and start a business action.

Click one of the available Case Types, and it calls the `<mbpm-case-instance-summary>` business component. This lists all of the active cases for that Case Type.

Click one of the available Cases, and it calls the Data window. This includes the following items:

- A summary of the state of the case. This calls the `<mbpm-case-states>` business component.
- Any available case actions. This calls the `<mbpm-case-actions>` business component.
- The read-only case data. This calls the `<mbpm-case-data>` business component.

There is a bottom bar that contains five navigation buttons. **Data** is the first button, and opens the Data window. The other navigation buttons open the following business components:

- **Work Item** calls the `<mbpm-work-list>` business component. Access work items linked to the case, applicable for the current user.
- **Audit** calls the `<mbpm-case-event-view>` business component. Review the audit history for the case.
- **Case Comments** calls the `<mbpm-comments>` business component. Add a new comment, and review the comment history for the case.
- **Case Documents** calls two business components: `<mbpm-document-upload>`, and `<mbpm-documents>`. This provides a list of documents for the case. Upload a new document to the case. Select a document provides options to view, delete or download that document.

Configure the Mobile Case Manager Application

The Mobile Case Manager application requires the details of a case model that is available on the ActiveMatrix BPM system. Edit the `mobileCaseMgrAppName.config.json` file to add this information.

`mobileCaseMgrAppName` is the application name. For the Mobile Case Manager application, this is `MobileCaseManager`. For a clone of the Mobile Case Manager application, it is the name chosen for the cloned application.

When the Mobile Case Manager application opens, the `index.html` file calls the `<mbpm-case-types>` business component to display the available case types for the current case model. The `mobileCaseMgrAppName.config.json` file defines the current model.

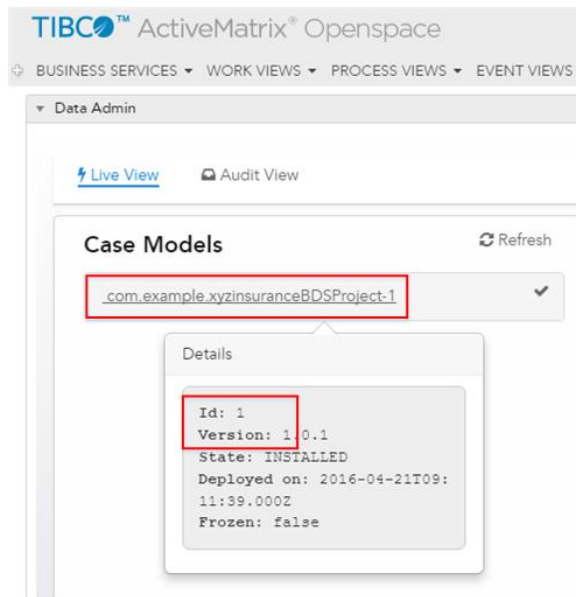
This file contains the JSON definition for the default `selectedCaseModel` object. For information about this object, see [CaseModelBasicInfoType](#).

```
{
  "selectedCaseModel":
  {"caseModelID":"1","appName":"com.example.customer-1","majorVersion":"1"}
}
```

Prerequisites

There are three alternatives to obtain the required information for the case model from the ActiveMatrix BPM system:

- From the ActiveMatrix BPM web service API, use the `getCaseModelBasicInfo` operation, from the `BusinessDataServicesAdminService`.
- From the objectAPI, use the `GlobaldataAdminService.getCaseModelBasicInfo` method.
- From Openspace, use the **DATA ADMIN** pane. For example:



Procedure

1. Open the `mobileCaseMgrAppName.config.json` file.
2. Edit the `selectedCaseModel` definition, and replace the default details with those of the required case model.

```
{
  "selectedCaseModel":
  {"caseModelID": "1", "appName": "com.example.customer-1", "majorVersion": "1"}
}
```

3. Do the following for a clone of the Mobile Case Manager application.
This is not required for the Mobile Case Manager application.
 - a) Open the `js/controllers.js` file.
 - b) Edit the `$http.get` call, and replace `MobileCaseManager.config.json` with the `mobileCaseMgrAppName.config.json` filename.
- ```
$http.get("MobileCaseManager.config.json")
 .then(function(response) {
```
4. Save the changes.
  5. **Launch** the application to test it.  
Remember to launch the **Latest** version.

## Adding a Business Component to Your Application

The HTML document to which you want to add the business component must:

- include the following libraries. (`tcf.nocache.js` contains the controllers and underlying framework logic used by each component service. The other libraries provide additional core functions used by component services)

```
<script type="text/javascript" src="/apps/app-cdn/tcf/tcf.nocache.js"></script>
<script type="text/javascript" src="/apps/app-cdn/jquery/jquery-3.1.1.js"></script>
<script src="/apps/app-cdn/angular/1.6.0/angular.min.js"></script>
<script src="/apps/app-cdn/angular-ui-grid/ui-grid.min.js"></script>
```

- attach an application controller to the DOM.

- add the HTML element for the business component you want to include.

The associated application controller must:

- specify a dependency on any required component services.
- define and set the required component data to be passed in the HTML element attributes.

### Example - Adding the <bpm-business-actions-drop-down> Business Component to a Page

The [BUSINESS ACTIONS](#) menu option in the Component Showcase application demonstrates how to add the [bpm-business-actions-drop-down](#) business component to a page.

The `componentShowcase/componentDemos/bizActions/BusinessActions.html` file:

- includes the required libraries and the source for the application controller.

```
<script type="text/javascript" src="/apps/app-cdn/tcf/tcf.nocache.js"></script>
<script type="text/javascript" src="/apps/app-cdn/jquery/jquery-3.1.1.js"></script>
<script src="/apps/app-cdn/angular/1.6.0/angular.min.js"></script>
<script src="/apps/app-cdn/angular-ui-grid/ui-grid.min.js"></script>

<script src="js/BizActionSample.js"></script>
```

- attaches the `BizActionSampleCtrl` controller to the DOM.

```
<body layout="column" ng-controller="BizActionSampleCtrl" ng-cloak>
```

- adds the `<bpm-business-actions-drop-down>` element to the page's sidebar.

```
<md-sidenav md-is-locked-open="$mdMedia('gt-sm')"
class="appTraySideNav md-whiteframe-z2" md-component-id="left"
layout="row">
 <bpm-business-actions-drop-down actions-
count="currActionsCount" set-main-content="setMainContent" form-
div="formDiv" action-groups="actionGroups" expand-
actions="expandActions"></bpm-business-actions-drop-down>
</md-sidenav>
```

The `componentShowcase/componentDemos/bizActions/js/BizActionSample.js` file defines the `BizActionSampleCtrl` controller, which has dependencies on the [BPMWorklistService](#) and [BPMLoginService](#) service.

```
(function() {
 var componentShowcase = angular.module("componentShowcase",
['TibcoFramework']);
 componentShowcase.controller("BizActionSampleCtrl", function
($scope, $rootScope, BPMBusinessActionsService) {
```

The `BizActionSampleCtrl` controller sets the attribute data required by the `<bpm-business-actions-drop-down>` element.

```
$scope.currActionsCount = 0;
$scope.actionGroups = [];
$scope.expandActions = true;
$scope.setMainContent = function(content) {
 $scope.main_content = content;
};
$scope.isMainContentShowingForm = function() {
 return ($scope.main_content == "form");
};
$scope.formDiv = "formAddDiv";
```

## Customizing a Business Component

Each business component uses a default HTML template. A copy of this template is also available to customize the appearance of the component, if required.

Custom templates are supplied as part of the example applications:

- Workapp: The template for each component is available in the file: `customTemplate/componentName/componentName.html`.
- Case Management: The template for each component is available in the file: `template/Customcomponent.html`.
- Mobile Case Manager: The template for each component is available in the file: `customTemplate/componentName/componentName.html`.



All the custom templates have built in styles provided inside the template. It is best practice to provide all the styles for a custom template inside the template and not in the Cascading Style Sheets (.css) file.

Each application has a `css` folder that contains the `.css` files.

### Procedure

1. Export the Workapp application.
2. Copy the custom template to a suitable location in the application and modify it to suit the requirements.
3. Use the `template-path` attribute to specify the location of the custom template as part of the HTML element for the business component inside the HTML document.

## Business Components Reference

### bpm-attributes-view

The `<bpm-attributes-view>` business component lists the names and current value of every work item attribute defined for the selected work item.

The component's internal controller invokes the `BPMWorkItemDetailsService.populateWorkItemAttributes` operation to provide the data that is displayed from the `WorkItemDetailsModel` object.

If you want to customize the contents of the `<bpm-attributes-view>` business component, see [Customizing a Business Component](#) for more information.

### Usage

```
<bpm-attributes-view
 selected-work-item="object"
 [template-path="string"]>
</bpm-attributes-view>
```

### Attributes

Name	Type	Description
selected-work-item	<a href="#">WorkItem</a> object	Currently selected work item

Name	Type	Description
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template

### Example

The `<bpm-attributes-view>` business component is used:

- on the [WORK ITEM DETAILS](#) tab of the Component Showcase application.
- in the My Work App application, when you select a work item in a Work View, then click **DETAILS** in the footer menu, then click **Attributes**.

In each case, `<bpm-attributes-view>` is called from the [work-item-detail-nav](#) business component.

## bpm-business-actions-drop-down

The `<bpm-business-actions-drop-down>` business component displays a menu of the business actions available to the logged in user.

The data is displayed using an accordion control, which:

- allows you to start a selected business action (using the `BPMBusinessActionsService.startBizAction` operation).
- provides a search control.
- provides a count of the total number of business actions in the menu.

If you want to customize the contents of the `<bpm-business-actions-drop-down>` business component, see [Customizing a Business Component](#) for more information.

### Usage

```
<bpm-business-actions-drop-down
 actions-count="number"
 set-main-content="functionReference"
 form-div="string"
 action-groups="array"
 toggle-expand-fn="functionReference"
 expand-actions="boolean"
 [template-path="string"]>
</bpm-business-actions-drop-down>
```

In ActiveMatrix BPM 4.0 the `<bpm-business-actions-drop-down>` business component was called `<business-actions-drop-down>`. In this release of ActiveMatrix BPM you can use either name to call this business component, as follows:

```
<business-actions-drop-down
 // all attributes are the same as for <bpm-business-actions-drop-down>
 // ...
</business-actions-drop-down>
```

However, best practise is to use `<bpm-business-actions-drop-down>`. `<business-actions-drop-down>` has been deprecated in ActiveMatrix BPM 4.1 and may be removed in a later release.



## Attributes

Name	Type	Description
actions-count	Number	Number of business actions in the business actions menu. This value is populated by the BPMBusinessActionsService. <a href="#">fetchTopCategories</a> or <a href="#">fetchChildCategories</a> operation.
set-main-content	Function reference	Updates the parent scope or application about what is being shown in the main content area.
form-div	String	ID of the <div> element in which an opened business action form is to be rendered.
action-groups	Array	List of business actions, mapped to their categories, available to the logged in user. This array is populated by the BPMBusinessActionsService. <a href="#">fetchTopCategories</a> or <a href="#">fetchChildCategories</a> operation.
toggle-expand-fn	Function reference	Notifies the parent scope about toggling of the business actions menu.
expand-actions	Boolean	Defines whether the business actions menu should be initially displayed as expanded ( <code>true</code> ) or contracted ( <code>false</code> ).
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template.

## Example

The <bpm-business-actions-drop-down> business component is used:

- on the BUSINESS ACTIONS tab of the Component Showcase application. See [BUSINESS ACTIONS](#) for more information about how it is used.
- in the My Work App application, available as the **Business Actions** accordion control in the side bar. (See `workapp/index.html`).

## bpm-case-actions

The <bpm-case-actions> business component displays the available case actions for a specified case.

The component's internal controller invokes the CaseManagementService.[listCaseActions](#) operation to provide the data that is displayed from the CaseManagementModel.[caseActions](#) object.

If you want to customize the contents of the <bpm-case-actions> business component, see [Customizing a Business Component](#) for more information.

## Usage

```
<bpm-case-actions
 case-ref="string"
 [case-action-success="functionReference"]
 [template-path="string"]>
</bpm-case-actions>
```

## Attributes

Name	Type	Description
case-ref	String	Case reference for which case actions should be displayed.
case-action-success	functionReference	(Optional) Function that executes after the case action is successfully executed.
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template

## User Interaction

The default component template displays the case actions as a horizontal list of buttons. Each button contains the case action's name. If there are more case actions than will fit in the available display space, a **More** button is displayed providing a drop-down list of the case actions that cannot be displayed.

When a user clicks a button, the case action is started and the appropriate form is displayed in a tab, titled with the case action's name. The user must Cancel or Submit the form before they can start another instance of the same case action. Clicking the same button when a tab containing that case action is already open simply sets the focus on that tab. It does not open a new tab containing another instance.

## Example

The `<bpm-case-actions>` business component is used in the Case Management application when you click **Case Actions** on the **Case Menu** for a case.

## bpm-case-data

The `<bpm-case-data>` business component displays the case data for a specified case.


The component's internal controller invokes the `CaseManagementService.getCaseData` operation to provide the data that is displayed from the `CaseManagementModel.caseDataTree` object.

If you want to customize the contents of the `<bpm-case-data>` business component, see [Customizing a Business Component](#) for more information.

## Usage

```
<bpm-case-data
 case-ref="string" | case-data-array="object"
 [template-path="string"]>
</bpm-case-data>
```

## Attributes

Name	Type	Description
case-ref	String	Case reference for which case data should be displayed.
case-data-array	Array	<p>A previously populated <code>CaseManagementModel.caseDataTree</code> object containing the case data to be displayed.</p> <div>  <p>The <code>getCaseData</code> operation is not invoked when you use this attribute.</p> </div>

Name	Type	Description
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template



case-ref and case-data-array are mutually exclusive.

### User Interaction

Case data is displayed as a read-only form, displaying the objects as a hierarchy. The user cannot modify any of the displayed fields.

### Example

The <bpm-case-data> business component is used in the Case Management application when you click **Case Overview** on the **Case Menu** for a case. This is also the default option displayed when you click a **Case** from the main **Cases** pane.

The **Case Overview** display shows both the <bpm-case-data> business component and, above it, the [bpm-case-states](#) business component.

## bpm-case-details-page

The <bpm-case-details-page> business component displays case-specific information and options for a specific case, using a main pane and a right-hand sidebar.

If you want to customize the contents of the <bpm-case-details-page> business component, see [Customizing a Business Component](#) for more information.

### Usage

```
<bpm-case-details-page
 case-ref="string"
 current-state="string"
 work-list-data="object"
 on-click-case-instance="functionReference"
 side-bar="string"
 nav-template="string"
 main-content="string"
 [template-path="string"]>
</bpm-case-details-page>
```

### Attributes

Name	Type	Description
case-ref	String	Case reference of the currently selected case instance.
current-state	String	Current state of the case instance.
work-list-data	Object	Work list for the selected case instance.
on-click-case-instance	Function reference	Defines the actions to be performed when the user selects a linked case.

Name	Type	Description
side-bar	String	Relative pathname of the custom template file to use to display components in the sidebar.
nav-template	String	Relative pathname of the custom template file to use to display the Case menu provided in the sidebar.
main-content	String	Relative pathname of the custom template file to use to display components in the main pane.
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template

## User Interaction

The `<bpm-case-details-page>` business component is called when the user clicks one of the available cases for a selected case type:

- The main pane displays a summary of the case state, any available case actions for this case, and the case data (read-only), using the following business components:
  - `<bpm-case-states>`
  - `<bpm-case-actions>`
  - `<bpm-case-data>`
- A right-hand sidebar provides additional case-specific information and functionality, using the following business components:

Sidebar section...	uses business component(s)...	to display...
Case Work items	<code>&lt;bpm-linked-work-items&gt;</code>	Any work items for the currently logged in user that are linked to this case.
Linked Cases	<code>&lt;bpm-linked-case-instances&gt;</code>	Any other cases that are linked to this case.
Case Documents	<code>&lt;documents-view&gt;</code>	List of documents for this case.
Case Comments	<code>&lt;bpm-comments&gt;</code>	The case's comment history. You can also add further comments.
Audit	<code>&lt;bpm-case-event-view&gt;</code>	The case's audit trail.

Alternatively, you can toggle the right-hand sidebar to instead display a simple **Case Menu**. Selecting one of the menu items displays the appropriate information for the case in the main pane, as follows:



Menu item...	uses business component(s)...	to display...
Case Overview	<code>&lt;bpm-case-states&gt;</code> , <code>&lt;bpm-case-actions&gt;</code> , <code>&lt;bpm-case-data&gt;</code>	Case state, case actions and case data (read-only). You can select and perform a case action.
Case Work items	<code>&lt;bpm-linked-work-items&gt;</code>	Any work items for the currently logged in user that are linked to this case.
Linked Cases	<code>&lt;bpm-linked-case-instances&gt;</code>	Any other cases that are linked to this case. Clicking a linked case displays the case overview for that case.
Case Documents	<code>&lt;bpm-document-upload&gt;</code> , <code>&lt;documents-view&gt;</code>	List of documents for this case. Selecting a particular document gives you further options to view, delete or download that document. You can also upload new documents to the case.
Case Comments	<code>&lt;bpm-comments&gt;</code>	The case's comment history. You can also add further comments.
Audit	<code>&lt;bpm-case-event-view&gt;</code>	The case's audit trail.

### Example

The `<bpm-case-details-page>` business component is used in the Case Management application when you click one of the available **Cases** for a selected **Case Type**.

### bpm-case-event-view

The `<bpm-case-event-view>` business component displays the audit trail for a specified case.


The component's internal controller invokes the `CaseEventViewService.getSearchCaseReport` operation to provide the audit data from the `CaseEventViewModel.eventDataList` object. The data is displayed using an accordion control, which can have optional sort controls.

If you want to customize the contents of the `<bpm-case-event-view>` business component, see [Customizing a Business Component](#) for more information.

### Usage

```
<bpm-case-event-view
 case-reference="string" | event-data-list="array"
 show-controls="boolean"
 just-open-one-item="boolean"
 default-open-index="number"
 [template-path="string"]>
</bpm-case-event-view>
```

## Attributes

Name	Type	Description
case-reference	String	Case reference for which the audit trail should be displayed.
event-data-list	Array	<p>A previously populated <a href="#">CaseEventViewModel.eventDataList</a> object containing the audit trail data to be displayed.</p> <p> The <a href="#">getSearchCaseReport</a> operation is not invoked when you use this attribute.</p>
show-controls	Boolean	Defines whether the <b>Sort events</b> control is displayed ( <code>true</code> ) or hidden ( <code>false</code> ). The default option is <code>false</code> .
just-open-one-item	Boolean	Defines whether expanding an event item in the accordion causes an already open event item to close ( <code>true</code> ) or not ( <code>false</code> ). The default option is <code>false</code> .
default-open-index	Number	Index number of the event item in the accordion to open by default.
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template



case-reference and event-data-list are mutually exclusive.

## Example

The `<bpm-case-event-view>` business component is used in the Case Management application when you click **Case Events** on the **Case Menu** for a case.

## bpm-case-instance-summary

The `<bpm-case-instance-summary>` business component displays case summary information for one or more cases.

The component's internal controller invokes the `CaseManagementService.getCaseInformation` operation to provide the data that is displayed from the `CaseManagementModel.caseRefInformation` object.

If you want to customize the contents of the `<bpm-case-instance-summary>` business component, see [Customizing a Business Component](#) for more information.

## Usage

```
<bpm-case-instance-summary
 case-ref="string" | case-references-array="array"
 on-click-case-instance-summary="functionReference"
 [config="object"]
 [template-path="string"]>
</bpm-case-instance-summary>
```

## Attributes

Name	Type	Description
case-ref	String	Single case reference for which case summary information should be displayed.
case-references-array	Array	Array of case references for which case summary information should be displayed.
on-click-case-instance-summary	Function reference	Defines the actions to be performed when the user selects one of the displayed case instances.
config	Object	(Optional) Defines whether the data should be displayed using a CARD or LIST format. The default option is CARD. The parameter is viewType, of type String.
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template



case-ref and case-references-array are mutually exclusive.

## Example

The <bpm-case-instance-summary> business component is used in the Case Management application when you click a **Case Type** on the main **Cases** pane.

## bpm-case-states

The <bpm-case-states> business component displays the available case states and current case state for a specified case.

The component's internal controller invokes the CaseManagementService.[getCaseStates](#) operation to provide the data that is displayed from the [CaseManagementModel.caseStates](#).


If you want to customize the contents of the <bpm-case-states> business component, see [Customizing a Business Component](#) for more information.

## Usage

```
<bpm-case-states
 case-ref="string" | case-states-array="array" [current-state="string"]
 [template-path="string"]>
</bpm-case-states>
```

## Attributes

Name	Type	Description
case-ref	String	Case reference for which the case state data should be displayed.

Name	Type	Description
case-states-array	Array	<p>A previously populated <a href="#">CaseManagementModel.caseStates</a> object containing the case state data to be displayed.</p> <p> The <a href="#">getCaseStates</a> operation is not invoked when you use this attribute.</p>
current-state	String	(Optional) Current state of the case. This allows you to update the data displayed by the case-states-array attribute if the case state has been updated since that object was populated (for example, by the execution of a case action).
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template



Use either case-ref, or case-states-array and current-state. These options are mutually exclusive.

### Example

The <bpm-case-states> business component is used in the Case Management application when you click **Case Overview** on the **Case Menu** for a case. This is also the default option displayed when you click a **Case** from the main **Cases** pane.

The **Case Overview** display shows both the <bpm-case-states> business component and, below it, the <bpm-case-data> business component.

## bpm-case-types

The <bpm-case-types> business component displays the available case types for a specified case model.

The component's internal controller invokes the CaseManagementService.[listCaseTypes](#) operation to provide the data that is displayed from the [CaseManagementModel.caseTypes](#) object.


If you want to customize the contents of the <bpm-case-types> business component, see [Customizing a Business Component](#) for more information.

### Usage

```
<bpm-case-types
 case-model="object" | case-types-array=array"
 on-click-case-type=functionReference
 [config=object]
 [template-path="string"]>
</bpm-case-types>
```

### Attributes

Name	Type	Description
case-model	<a href="#">CaseModelBasicInfoType</a> object	<p>Case model for which case types should be displayed.</p> <p>You can obtain the information needed to populate the <a href="#">CaseModelBasicInfoType</a> object using the <a href="#">GlobaldataAdminService.getCaseModelBasicInfo</a> method from the objectAPI.</p>

Name	Type	Description
case-types-array	Array	<p>A previously populated <a href="#">CaseManagementModel.caseTypes</a> object containing the case types to be displayed.</p> <p> The <a href="#">listCaseTypes</a> operation is not invoked when you use this attribute.</p>
on-click-case-type	Function reference	Defines the actions to be performed when the user selects one of the displayed case types.
config	Object	(Optional) Defines whether the list of case types should be displayed using a CARD or LIST format. The default option is CARD. The parameter is <code>viewType</code> , of type String.
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template



`case-model` and `case-types-array` are mutually exclusive.

### Example

The `<bpm-case-types>` business component is used in the Case Management application when you click the **Cases** button on the main navigation sidebar. This is also the default option displayed when you run the Case Management application.



The Case Management application uses a function to populate the [CaseModelBasicInfoType](#) object with data from the `CaseManagement.config.json` file. Before using the application, you must edit the `CaseManagement.config.json` file so that it contains the details of a case model that exists on your ActiveMatrix BPM system. See [Configuring the Case Management Application](#) for more information.

## bpm-comments

The `<bpm-comments>` business component renders a user interface that can be used to add comments to a work item, process instance, or case, or to display comments that had been previously added to those entities.

A data model is passed to the component that contains a work item ID, process instance ID, or case reference, to identify the type of entity for which comments are being displayed or added. Depending on whether the data model contains a work item ID, process instance ID, or case reference, the appropriate operation is called in the [BPMCommentService](#).

Unlike most of the other business components provided in the framework, this component does not have a template, and it is not demonstrated in the Component Showcase Application. Instead, an example of displaying and adding case comments is illustrated in the Case Management application that is provided in Application Development.

### Usage

The `<bpm-comments>` business component can be invoked in your HTML file using the following form:

```
<bpm-comments
 position="string"
 avatarSrc="string"
 comment-model="object">
</bpm-comments>
```

## Attributes

Name	Type	Description
position	String	The position of the user icon (avatar); see avatarSrc below. The valid values are "left" and "right". Default = "left"
avatarSrc	String	A graphical representation of the user. This is a relative path to the image file. Images must be of type SVG.
comment-model	Object	Identifies the data model that identifies the work item, process instance, or case for which comments are added or displayed.

## Example

The following is from the example Case Management application:

```
<div ng-show="showData=='Case Comments'" layout="column" flex>
 <div class="md-whiteframe-z1 panel" flex layout="column">
 <md-toolbar class="md-theme-light panel-heading md-toolbar-tools">
 <h3>Comments</h3>
 </md-toolbar>
 <div class="comment-parent" layout="column" flex>
 <bpm-comments position="left" avatarSrc="Images/userprofile.svg" comment-
model="commentModel"></bpm-comments>
 </div>
 </div>
```

The `<bpm-comments>` business component is used in the Case Management application when you click **Case Comments** on the **Case Menu** for a case.

## bpm-document-upload

The `<bpm-document-upload>` business component uploads a document from the file system for a case instance.

If you want to customize the contents of the `<bpm-document-upload>` business component, see [Customizing a Business Component](#) for more information.

## Usage

```
<bpm-document-upload
 case-ref="string"
 [template-path="string"]>
</bpm-document-upload>
```

## Attributes

Name	Type	Description
case-ref	String	Case reference for which a document should be uploaded.
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template

## User Interaction

The default component template displays **Choose File** and **UPLOAD** buttons, which the user can use to select and upload the desired file.

## Example

The `<bpm-document-upload>` and `<bpm-documents-view>` business components are used in the Case Management application when you click **Case Documents** on the **Case Menu** for a case.

## bpm-event-view

The `<bpm-event-view>` business component displays the audit trail for a specified process instance.

The component's internal controller invokes the `BPMEventViewService.getEventView` operation to provide the audit data. The data is displayed using an accordion control, which can have optional sort controls.

If you want to customize the contents of the `<bpm-event-view>` business component, see [Customizing a Business Component](#) for more information.

## Usage

```
<bpm-event-view
 event-data-list="array"
 process-instance-id="number"
 show-controls="boolean"
 just-open-one-item="boolean"
 default-open-index="number"
 [template-path="string"]>
</bpm-event-view>
```

## Attributes

Name	Type	Description
event-data-list	Array	Audit trail data for the specified <code>process-instance-id</code> , returned by the <code>BPMEventViewService.getEventView</code> operation.
process-instance-id	Number	ID of the process instance for which the audit trail should be displayed.
show-controls	Boolean	Defines whether the <b>Sort events</b> control is displayed ( <code>true</code> ) or hidden ( <code>false</code> ).
just-open-one-item	Boolean	Defines whether expanding an event item in the accordion causes an already open event item to close ( <code>true</code> ) or not ( <code>false</code> ).
default-open-index	Number	Index number of the event item in the accordion to open by default.
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template.

## Example

The `<bpm-event-view>` business component is used:

- on the **WORK LIST** tab of the Component Showcase application, when a work item is selected and you click the **E (Events)** button in the right-hand pane.
- in the My Work App application, when you select a work item in a Work View, then click **DETAILS** in the footer menu, then click the **E (Events)** button in the right-hand pane.

In each case, `<bpm-event-view>` is called from the [work-list](#) business component.

## bpm-highcharts

The `<bpm-highcharts>` business component displays a chart that contains information about process templates, process instances, audit events, cases, or work items.

This component's internal controller invokes the [BPMChartService.getChart](#) operation to aggregate the requested data, return it in a tabular format, then display the data in a chart using the [Highcharts](#) library.

Unlike most of the other business components provided in the framework, this component does not have a template, and it is not demonstrated in the Component Showcase Application.

### Prerequisites

- [Define a Data Model in the Controller](#) on page 57
- [Define a Chart Configuration Object in the Controller](#) on page 58
- [Call the getChart Operation from the Controller](#) on page 59
- [Include the Highcharts Library in your HTML File](#) on page 59

### Usage

After the prerequisites listed above are fulfilled, the `<bpm-highcharts>` business component can be invoked in your HTML file using the following form:

```
<bpm-highcharts config="chartConfiguration"></bpm-highcharts>
```

### Attributes

Name	Type	Description
config	String	<p>This specifies the configuration for the chart, for example, the type of chart (bar, pie, and so on), the text to be displayed on the x-axis and y-axis, and so on. The available configuration options can be found on the <a href="#">Highcharts</a> website.</p> <p>This configuration is defined in the controller.</p>

### Example

```
<body layout="column" ng-controller="BPMHighchartsCtrl" ng-cloak >
 <div flex>
 <bpm-highcharts config="chartConfig1"></bpm-highcharts>
 <bpm-highcharts config="chartConfig2"></bpm-highcharts>
 <bpm-highcharts config="chartConfig3"></bpm-highcharts>
 </div>
</body>
```

This causes three charts to be rendered, each configured differently as defined in configuration objects `chartConfig1`, `chartConfig2`, and `chartConfig3`.

The configuration objects and data models are defined in the controller, `BPMHighchartsCtrl`.



## Define a Data Model in the Controller

The data model specifies the data to return and display in the chart. This needs to be defined in the controller.

The data model consists of a `dataPayload` object in the form:

```
$scope.chartModel1 = {
 "dataPayload" : {
 "query": {
 "filter": "",
 "area": "",
 "group": "",
 "category": {
 "attribute": "",
 "type": ""
 }
 },
 "options": ""
 }
};
```

where:

- **filter** - A query string that restricts the amount of data returned for the chart. For information about valid query syntax, see [Defining Query Filter Strings](#).
- **area** - Specifies the type of information that is to be displayed in the chart. Must be one of the following:
  - WORK\_ITEMS
  - PROCESS\_TEMPLATES
  - PROCESS\_INSTANCES
  - AUDIT
  - CASE\_OBJECTS
- **group** (optional; default = all audit data is returned) - This is an attribute to further separate the data in the chart. The attributes that can be specified depends on the type of data specified in the **area** attribute. See "Attributes" below.
- **category** - This is an attribute/type pair that specifies the type of data to show in the chart.
  - **attribute** - The name of the attribute that contains the data used to construct the chart. The attributes that can be specified depends on the type of data specified in the **area** attribute. See "Attributes" below.
  - **type** - Type of data displayed in the chart. Currently, the only valid entry is COUNT (that is, the number of items specified by the other attributes).
- **options** - Not currently used.

### Attributes

Some of the parameters above require an attribute name. The allowable attributes depends on the type of data displayed in the chart. The following links provide lists of attributes for each type of data:

Work items	See <a href="#">WORKITEM_STATS Option Attributes</a>
Process templates	See <a href="#">PROCESS_TEMPLATES Attributes</a>
Process instances	See <a href="#">PROCESS_STATS Option Attributes</a>

Events	See <a href="#">AUDIT Option Attributes</a>
Cases	See <a href="#">CASE_STATS Option Attributes</a>

### Example Data Model

The following is an example data model that charts process instance counts, arranged by status:

```
$scope.chartModel1 = {
 "dataPayload" : {
 "query": {
 "filter": "NOT (status in ('pi_cancelled','pi_failed','pi_completed'))",
 "area": "PROCESS_INSTANCES",
 "group": "status",
 "category": {
 "attribute": "processTemplateName",
 "type": "COUNT"
 }
 },
 "options": ""
 }
};
```

### Define a Chart Configuration Object in the Controller

The chart configuration object defines the type of chart, for example bar or pie chart, as well other aspects like titles of columns. This object must be specified in the controller.

After the chart configuration object is defined in the controller, it is then passed in the **config** attribute when calling the <bpm-highcharts> business component in your HTML file. For more information, see [bpm-highcharts](#) on page 56.

The structure of the chart configuration object is defined by Highcharts. For complete information about the configuration options, see <http://api.highcharts.com/highcharts>.

An example for a bar chart with the title "Process Instances" is shown below:

```
$scope.chartConfig1={
 chart: {
 type: 'bar'
 },
 title: {
 text: 'Process Instances'
 },
 xAxis: {
 categories: [],
 title: {
 text: null
 }
 },
 yAxis: {
 min: 0,
 title: {
 text: 'Process Instances',
 align: 'high'
 },
 labels: {
 overflow: 'justify'
 }
 },
 plotOptions: {
 series: {
 stacking: 'normal'
 }
 },
 credits: {
 enabled: false
 },
};
```

```

 series: []
}

```

## Call the getChart Operation from the Controller

The getChart operation is called from the controller, passing in the data model for the chart.

The following is an example of calling `BPMChartService.getChart` from an application, passing in the `chartModel11` data model, which is defined in the controller:

```

BPMChartService.getChart($scope.chartModel11,{
 onSuccess: function(result){
 var chartData = result.data;

 console.log(chartData);

 // categories object
 var categories = chartData.categories.category;

 $scope.chartConfig1.xAxis.categories = categories;

 // series object
 var series = [];

 if(chartData && chartData.data){
 for(var i =0; i< chartData.data.length; i++){
 var temp = {
 name: typeof chartData.data[i].group === 'string'?
chartData.data[i].group: "None" ,
 data: chartData.data[i].dataset.value
 }

 series.push(temp);
 }
 }

 $scope.chartConfig1.series = series;
 },
 onFailure: function(exception){

 // GetChartFailed:
 console.log(exception);
 }
})

```

## Include the Highcharts Library in your HTML File

To call the <bpm-highcharts> business component from your HTML file, you must include the Highcharts library in the HTML file.

Include the Highcharts library with the other libraries required for client application framework-based applications, as follows:

```

<script type="text/javascript" src="/apps/app-cdn/tcf/tcf.nocache.js"></script>
<script type="text/javascript" src="/apps/app-cdn/jquery/jquery-3.1.1.js"></script>
<script src="/apps/app-cdn/angular/1.6.0/angular.min.js"></script>
<script src="/apps/app-cdn/angular-ui-grid/ui-grid.min.js"></script>
<script src="/apps/app-cdn/highcharts/4.2.3/highcharts.js"></script>

```

## bpm-linked-case-instances

The <bpm-linked-case-instances> business component displays the case instances that are linked to a specified case instance.

The component's internal controller invokes the `CaseManagementService.getRelatedCases` operation to provide the data that is displayed from the `CaseManagementModel.relatedCaseReferences` object.

If you want to customize the contents of the <bpm-linked-case-instances> business component, see [Customizing a Business Component](#) for more information.

### Usage

```
<bpm-linked-case-instances
 case-ref="string"
 on-click-case-instance="functionReference"
 [config="object"]
 [template-path="string"]>
</bpm-linked-case-instances>
```

### Attributes

Name	Type	Description
case-ref	String	Case reference for which linked case instances should be displayed.
on-click-case-instance	Function reference	Defines the actions to be performed when the user selects one of the displayed linked case instances.
config	Object	(Optional) Defines whether the data should be displayed using a CARD or LIST format. The default option is CARD. The parameter is viewType, of type String.
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template

### Example

The <bpm-linked-case-instances> business component is used in the Case Management application when you click **Linked Cases** on the **Case Menu** for a case.

## bpm-login


Allows users to log in.

If you want to customize the contents of the <bpm-event-view> business component, see [Customizing a Business Component](#) for more information.

### Usage

```
<bpm-login
 redirect="string"
 [template-path="string"]>
</bpm-login>
```

## Attributes

Name	Type	Description
redirect	String	URL to redirect to after a successful login.  Alternatively, you can use the <code>tcf-redirect</code> parameter on the login URL.
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template.

## bpm-process-view

The `<bpm-process-view>` business component provides information about the process that generated the selected work item.

The component's internal controller invokes the `BPMWorkItemDetailsService.populateWorkItemProcessDetails` operation to provide the data that is displayed from the `WorkItemDetailsModel` object.

If you want to customize the contents of the `<bpm-process-view>` business component, see [Customizing a Business Component](#) for more information.

## Usage

```
<bpm-process-view
 selected-work-item="object"
 [template-path="string"]>
</bpm-process-view>
```

## Attributes

Name	Type	Description
selected-work-item	<a href="#">WorkItem</a> object	Currently selected work item
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template.

## Example

The `<bpm-process-view>` business component is used:

- on the [WORK ITEM DETAILS](#) tab of the Component Showcase application.
- in the My Work App application, when you select a work item in a Work View, then click **DETAILS** in the footer menu, then click **Processes**.

In each case, `<bpm-process-view>` is called from the [work-item-detail-nav](#) business component.

## bpm-summary-view

The `<bpm-summary-view>` business component provides summary information about the selected work item.

The component's internal controller invokes the `BPMWorkItemDetailsService.populateWorkItemDetails` operation to provide the data that is displayed from the `WorkItemDetailsModel` object.

If you want to customize the contents of the `<bpm-summary-view>` business component, see [Customizing a Business Component](#) for more information.

### Usage

```
<bpm-summary-view
 selected-work-item="object"
 [template-path="string"]>
</bpm-summary-view>
```

### Attributes

Name	Type	Description
selected-work-item	<a href="#">WorkItem</a> object	Currently selected work item
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template.

### Example

The `<bpm-summary-view>` business component is used:

- on the [WORK ITEM DETAILS](#) tab of the Component Showcase application.
- in the My Work App application, when you select a work item in a Work View, then click **DETAILS** in the footer menu, then click **Summary**.

In each case, `<bpm-summary-view>` is called from the [work-item-detail-nav](#) business component.

## bpm-work-item-history

The `<bpm-work-item-history>` business component displays the audit trail for a specified work item.

The component's internal controller invokes the `BPMWorkItemHistoryService`.[getWorkItemHistory](#) operation to provide the audit data.

If you want to customize the contents of the `<bpm-work-item-history>` business component, see [Customizing a Business Component](#) for more information.

### Usage

```
<bpm-event-view
 history-data-list="array"
 work-item-id="number"
 [template-path="string"]>
</bpm-event-view>
```

### Attributes

Name	Type	Description
history-data-list	Array	Audit trail data for the specified <code>work-item-id</code> , populated by the <code>BPMWorkItemHistoryService</code> . <a href="#">getWorkItemHistory</a> operation.
work-item-id	Number	ID of the work item for which the audit trail should be displayed.
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template.

## Example

The `<bpm-work-item-history>` business component is used:

- on the **WORK LIST** tab of the Component Showcase application, when a work item is selected and you click the **H** (History) button in the right-hand pane.
- in the My Work App application, when you select a work item in a Work View, then click **DETAILS** in the footer menu, then click the **H** (History) button in the right-hand pane.

In each case, `<bpm-work-item-history>` is called from the [work-list](#) business component.

## bpm-work-list

The `<bpm-work-list>` business component displays a work list for a specified user.

The component's internal controller invokes [BPMWorklistService](#) operations to populate the work list. The component also allows you to:

- display the work list contents as cards, a list or a table.
- sort the work list by priority, process or date.
- search the work list.
- select, deselect or open work items.
- display an additional menu of controls when you have one or more work items selected, which you can use to perform additional tasks. (The controller makes individual controls available or unavailable depending on whether one or more work items is selected, and the state and context of those work items). Each control invokes a particular business component or component service to provide the required functionality.

Control	Action	Invokes
<b>OPEN</b>	Open a selected work item.	BPMWorklistService. <a href="#">openWorkItem</a> operation
<b>OPENNEXT</b>	Open the next work item in a chained or piled sequence.	BPMWorklistService. <a href="#">openNextWorkItem</a> operation
<b>DETAILS</b>	Display additional information about the selected work item.	<a href="#">work-item-detail-nav</a> business component
<b>ALLOCATE</b>	Allocate one or more work items.	BPMWorklistService. <a href="#">allocateWorkItem</a> operation
<b>RE-OFFER</b>	Re-offer one or more work items.	BPMWorklistService. <a href="#">unallocateWorkItem</a> operation

If you want to customize the contents of the `<bpm-work-list>` business component, see [Customizing a Business Component](#) for more information.

## Usage

```
<bpm-work-list
 template-data="object"
 [template-path="string"]>
</bpm-work-list>
```

In ActiveMatrix BPM 4.0 the `<bpm-work-list>` business component was called `<work-list>`. In this release of ActiveMatrix BPM you can use either name to call this business component, as follows:

```
<work-list
 // all attributes are the same as for <bpm-work-list>
 // ...
/>
```

However, best practise is to use `<bpm-work-list>`. `<work-list>` has been deprecated in ActiveMatrix BPM 4.1 and may be removed in a later release.

### Attributes

Name	Type	Description
template-data	<a href="#">workListData Object</a>	Data for the work list.
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template.

### Example

The `<bpm-work-list>` business component is used:

- on the **WORK LIST** tab of the Component Showcase application.
- in the My Work App application, in the main display area. (See `workapp/index.html`).
- in the Case Management application, when you click **Case Work Items** on the **Case Menu** for a case.

## bpm-work-views

The `<bpm-work-views>` business component displays a menu of the work views that are available to the logged in user.

The component's internal controller invokes the `BPMWorklistService.getViewForResource` operation to populate the list of work views. The data is displayed using an accordion control. Selecting a particular work list updates the list of work items displayed by the `work-list` business component.

If you want to customize the contents of the `<bpm-work-views>` business component, see [Customizing a Business Component](#) for more information.

### Usage

```
<bpm-work-views
 template-data="object"
 [template-path="string"]>
</bpm-work-views>
```

In ActiveMatrix BPM 4.0 the `<bpm-work-views>` business component was called `<work-views>`. In this release of ActiveMatrix BPM you can use either name to call this business component, as follows:

```
<work-views
 // all attributes are the same as for <bpm-work-views>
 // ...
/>
```

However, best practise is to use `<bpm-work-views>`. `<work-views>` has been deprecated in ActiveMatrix BPM 4.1 and may be removed in a later release.



## Attributes

Name	Type	Description
template-data	<a href="#">templateDataModel</a> object	Data for the work view.
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template.

## Example

The `<bpm-work-views>` business component is used:

- on the **WORK VIEWS** tab of the Component Showcase application.
- in the My Work App application, available as the **Work Views** accordion control in the side bar. (See `workapp/index.html`).

## documents-view

The `<documents-view>` business component displays the case documents associated with either a selected work item or a specified case.

The component's internal controller invokes the `CaseDocumentService.fetchDocuments` operation to provide the list of case documents.

If you want to customize the contents of the `<documents-view>` business component, see [Customizing a Business Component](#) for more information.

## Usage

```
<documents-view
 case-ref="string" | selected-work-item="object"
 [template-path="string"]>
</documents-view>
```

## Attributes

Name	Type	Description
case-ref	String	Case reference for which documents should be shown.
selected-work-item	<a href="#">WorkItem</a> object	Currently selected work item
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template.



`case-ref` and `selected-work-item` are mutually exclusive.

## Example

The `<documents-view>` business component is used:

- on the **CASE DOCUMENTS** tab of the Component Showcase application.

- on the [WORK ITEM DETAILS](#) tabs of the Component Showcase application, or in the My Work App application (when you select a work item in a Work View, then click **DETAILS** in the footer menu, then click **Documents**). In both cases, <documents-view> is called from the [work-item-detail-nav](#) business component.
- in the Case Management application, when you click **Case Documents** on the **Case Menu** for a case.

## mbpm-case-actions

The <mbpm-case-actions> business component displays the available case actions for a specified case.


The <mbpm-case-actions> business component calls the CaseManagementService.[listCaseActions](#) operation. This operation calls the [CaseManagementModel.caseActions](#) object.

To customize the contents of the <mbpm-case-actions> business component, see [Customizing a Business Component](#).

### Usage

```
<mbpm-case-actions
 case-ref="string" | case-actions-array="array"
 [template-path="string"]>
</mbpm-case-actions>
```

### Attributes

Name	Type	Description
case-ref	String	The <mbpm-case-actions> business component displays the case actions for this case reference.
case-actions-array	Array	<p>A previously populated <a href="#">CaseManagementModel.caseActions</a> object that contains the case actions to display.</p> <p> The use of this attribute does not call the CaseManagementService.<a href="#">listCaseActions</a> operation.</p>
template-path	String	(Optional) The relative pathname for a custom template file to use in place of the internal template.



case-ref and case-actions-array are mutually exclusive.

### User Interaction

The default component template displays the case actions as a horizontal set of buttons. Each button contains the name of the case action. If there are more case actions than will fit in the available display space, the user can swipe sideways to access the additional case actions.

When a user clicks a button, the case action starts and the window opens the applicable form. The user must **Submit** or **Cancel** the form. The <mbpm-case-actions> business component disables the case action buttons until the user clicks **Submit** or **Cancel**.

### Example

The Mobile Case Manager application uses the <mbpm-case-actions> business component as part of the Data window, which also uses the <mbpm-case-data> and <mbpm-case-states> business components. The Data window opens when the user clicks on an available Case, or clicks the **Data** button, which is one of the five navigation buttons on the bottom bar.

## mbpm-case-creator

The <mbpm-case-creator> business component uses a hierarchical list to display a menu of the business actions that are available to the user.

The <mbpm-case-creator> calls the BPMBusinessActionsService.[fetchTopCategories](#) and [fetchChildCategories](#) to build the hierarchical list.

When the user selects a business action, the <mbpm-case-creator> business component opens the associated form, and starts the business action. It uses the BPMBusinessActionsService.[startBizAction](#) operation to start the business action.

To customize the contents of the <mbpm-case-creator> business component, see [Customizing a Business Component](#).

### Usage

```
<mbpm-case-creator
 [config="object"]
 [template-path="string"]>
</mbpm-case-creator>
```

### Attributes

Name	Type	Description
config	Object	(Optional) The configuration object that allows the parent controller to refresh the view.
template-path	String	(Optional) The relative pathname for a custom template file to use in place of the internal template.

### Example

The Mobile Case Manager application uses the <mbpm-case-creator> business component as one of the two items on the Mobilespace menu.

## mbpm-case-data

The <mbpm-case-data> business component displays the case data for a specified case.


The <mbpm-case-data> business component calls the CaseManagementService.[getCaseData](#) operation. This operation calls the [CaseManagementModel.caseDataTree](#) object.

To customize the contents of the <mbpm-case-data> business component, see [Customizing a Business Component](#).

### Usage

```
<mbpm-case-data
 case-ref="string" | case-data-array="array"
 [config="object"]
 [template-path="string"]>
</mbpm-case-data>
```

## Attributes

Name	Type	Description
case-ref	String	The <mbpm-case-data> business component displays the case data for this case reference.
case-data-array	Array	<p>A previously populated <a href="#">CaseManagementModel.caseDataTree</a> object that contains the case data to display.</p> <div>  <p>The use of this attribute does not call the <a href="#">CaseManagementService.getCaseData</a> operation.</p> </div>
config	Object	(Optional) The configuration object that allows the parent controller to refresh the view.
template-path	String	(Optional) The relative pathname for a custom template file to use in place of the internal template.



case-ref and case-data-array are mutually exclusive.

## User Interaction

The <mbpm-case-data> business component displays the case data as a read-only form, and as a hierarchy. The user cannot modify any of the fields.

## Example

The Mobile Case Manager application uses the <mbpm-case-data> business component as part of the Data window, which also uses the <mbpm-case-actions> and <mbpm-case-states> business components. The Data window opens when the user clicks on an available Case, or clicks the **Data** button, which is one of the five navigation buttons on the bottom bar.

## mbpm-case-event-view

The <mbpm-case-event-view> business component displays the audit trail for a specified case.

The <mbpm-case-event-view> business component calls the [CaseEventViewService.getSearchCaseReport](#) operation to provide the audit data. This operation calls the [CaseEventViewModel.eventDataList](#) object.

To customize the contents of the <mbpm-case-event-view> business component, see [Customizing a Business Component](#) for more information.

## Usage

```
<mbpm-case-event-view
 config="object"
 [template-path="string"]>
</mbpm-case-event-view>
```

## Attributes

Name	Type	Description
config	Object	The configuration object that allows the parent controller to refresh the view. It also includes an <code>eventDataList</code> array that contains the audit trail data. The <code>&lt;mbpm-case-event-view&gt;</code> business component displays this audit trail.
template-path	String	(Optional) The relative pathname of a custom template file to use in place of the internal template

## Example

The Mobile Case Manager application uses the `<mbpm-case-event-view>` business component for the **Audit** button, which is one of the five navigation buttons on the bottom bar.

## mbpm-case-instance-summary

The `<mbpm-case-instance-summary>` business component displays case summary information for one or more cases.


The `<mbpm-case-instance-summary>` business component calls the `CaseManagementService.getCaseInformation` operation. This operation calls the `CaseManagementModel.caseRefInformation` object.

To customize the contents of the `<mbpm-case-instance-summary>` business component, see [Customizing a Business Component](#).

## Usage

```
<mbpm-case-instance-summary
 case-type="object"
 case-references-array="array"
 on-click-case-instance-summary="functionReference"
 [config="object"]
 [template-path="string"]>
</mbpm-case-instance-summary>
```

## Attributes

Name	Type	Description
case-type	Object	The <code>&lt;mbpm-case-instance-summary&gt;</code> business component displays case summary information for this case type.
case-references-array	Array	<p>A previously populated <a href="#">CaseManagementModel.caseRefInformation</a> object that contains the case summary information to display.</p> <div>  <p>The use of this attribute does not call the <code>CaseManagementService.getCaseInformation</code> operation.</p> </div>
on-click-case-instance-summary	Function reference	Defines the actions to perform when the user selects one of the case instances.

Name	Type	Description
config	Object	(Optional) The configuration object that allows the parent controller to refresh the view.
template-path	String	(Optional) The relative pathname for a custom template file to use in place of the internal template.

### Example

The Mobile Case Manager application uses the <mbpm-case-instance-summary> business component as part of the Case Types window, when the user clicks one of the available Case Types.

## mbpm-case-states

The <mbpm-case-states> business component displays the available case states and current case state for a specified case.


The <mbpm-case-states> business component calls the `CaseManagementService.getCaseStates` operation. This operation calls the `CaseManagementModel.caseStates` object.

To customize the contents of the <mbpm-case-states> business component, see [Customizing a Business Component](#).

### Usage

```
<mbpm-case-states
 case-ref="string" | case-states-array="array" [current-state="string"]
 [template-path="string"]>
</mbpm-case-states>
```

### Attributes

Name	Type	Description
case-ref	String	The <mbpm-case-states> business component displays the available case states and current case state for this case reference.
case-states-array	Array	<p>A previously populated <a href="#">CaseManagementModel.caseStates</a> object that contains the available case states and current case state to display.</p> <div>  <p>The use of this attribute does not call the <code>CaseManagementService.getCaseStates</code> operation.</p> </div>
current-state	String	(Optional) The current state of the case. If the case state changes after case-states-array was populated, this attribute updates the case-states-array attribute. For example, after the execution of a case action.
template-path	String	(Optional) The relative pathname of a custom template file to use in place of the internal template



Use either case-ref, or case-states-array and current-state. These options are mutually exclusive.

## Example

The Mobile Case Manager application uses the <mbpm-case-states> business component as part of the Data window, which also uses the <mbpm-case-actions> and <mbpm-case-data> business components. The Data window opens when the user clicks on an available Case, or clicks the **Data** button, which is one of the five navigation buttons on the bottom bar.

## mbpm-case-types

The <mbpm-case-types> business component displays the available case types for a specified case model.

The <mbpm-case-types> business component calls the `CaseManagementService.listCaseTypes` operation. This operation calls the `CaseManagementModel.caseTypes` object.

To customize the contents of the <mbpm-case-types> business component, see [Customizing a Business Component](#).

## Usage

```
<mbpm-case-types
 case-model="object"
 on-click-case-type=functionReference
 [config=object]
 [template-path="string"]>
</mbpm-case-types>
```

## Attributes

Name	Type	Description
case-model	<a href="#">CaseModelBasicInfoType</a> object	The <mbpm-case-types> business component displays the case types for this case model.  To obtain the information to populate the <a href="#">CaseModelBasicInfoType</a> object, use the <a href="#">GlobaldataAdminService.getCaseModelBasicInfo</a> method from the objectAPI.
on-click-case-type	Function reference	Defines the actions to perform when the user selects one of the case types.
config	Object	(Optional) The configuration object that allows the parent controller to refresh the view.
template-path	String	(Optional) The relative pathname for a custom template file to use in place of the internal template.

## Example

The Mobile Case Manager application uses the <mbpm-case-types> business component for the Case Types window.



The Mobile Case Manager application uses the `MobileCaseManager.config.json` file to configure the [CaseModelBasicInfoType](#) object with details of a case model that is available on the ActiveMatrix BPM system. To edit `MobileCaseManager.config.json`, see [Configure the Mobile Case Manager Application](#).

## mbpm-comments

The <mbpm-comments> business component renders a user interface so that a user can add comments to a work item, process instance, or case. It also displays existing comments.

The <mbpm-comments> business component receives a data model that contains a work item ID, process instance ID, or case reference, that it uses to identify the type of entity. It then uses the applicable operation from the [BPMCommentService](#) to display the comments for this entity.

### Usage

```
<mbpm-comments
 config="object"
 [template-path="string"]>
</mbpm-comments>
```

### Attributes

Name	Type	Description
config	Object	The configuration object that allows the parent controller to refresh the view. It also includes a <code>commentModel</code> object that identifies a data model, that identifies a work item, process instance, or case. The <mbpm-comments> business component displays the comments for this entity.
template-path	String	(Optional) The relative pathname of a custom template file to use in place of the internal template

### Example

The Mobile Case Manager application uses the <mbpm-comments> business component for the **Case Comments** button, which is one of the five navigation buttons on the bottom bar.

## mbpm-documents

The <mbpm-documents> business component displays the case documents associated with either a selected work item or a specified case.

The <mbpm-documents> business component calls the `CaseDocumentService.fetchDocuments` operation to provide the list of case documents.

To customize the contents of the <mbpm-documents> business component, see [Customizing a Business Component](#) for more information.

### Usage

```
<mbpm-documents
 case-ref="string" | selected-work-item="object"
 [template-path="string"]>
</mbpm-documents>
```

### Attributes

Name	Type	Description
case-ref	String	The <mbpm-documents> business component displays the documents for this case reference.



Name	Type	Description
selected-work-item	<a href="#">WorkItem</a> object	The <mbpm-documents> business component displays the documents for the currently selected work item.
template-path	String	(Optional) The relative pathname for a custom template file to use in place of the internal template.



case-ref and selected-work-item are mutually exclusive.

### Example

The Mobile Case Manager application uses the <mbpm-documents> and <mbpm-document-upload> business components for the **Case Documents** button, which is one of the five navigation buttons on the bottom bar.

## mbpm-document-upload

The <mbpm-document-upload> business component uploads a document from the file system for a case instance.

To customize the contents of the <mbpm-document-upload> business component, see [Customizing a Business Component](#) for more information.

### Usage

```
<mbpm-document-upload
 case-ref="string"
 [config="object"]
 [template-path="string"]>
</mbpm-document-upload>
```

### Attributes

Name	Type	Description
case-ref	String	The <mbpm-document-upload> business component uploads a document for this case reference.
config	Object	(Optional) The configuration object that allows the parent controller to refresh the view.
template-path	String	(Optional) The relative pathname for a custom template file to use in place of the internal template

### User Interaction

The default component template displays **Choose File** and **UPLOAD** buttons for a user to select and upload the desired file.

### Example

The Mobile Case Manager application uses the <mbpm-documents> and <mbpm-document-upload> business components for the **Case Documents** button, which is one of the five navigation buttons on the bottom bar.

## mbpm-work-list

The <mbpm-work-list> business component displays a work list for the current user.

The <mbpm-work-list> business component calls the [BPMWorklistService](#) operations to populate the work list and open a work item.

To customize the contents of the <mbpm-work-list> business component, see [Customizing a Business Component](#) for more information.

### Usage

```
<mbpm-work-list
 template-data="object"
 [template-path="string"]>
</mbpm-work-list>
```

### Attributes

Name	Type	Description
template-data	<a href="#">workListData</a> object	Data for the work list.
template-path	String	(Optional) The relative pathname for a custom template file to use in place of the internal template.

### Example

The Mobile Case Manager application uses the <mbpm-work-list> business component for the **Work Item** button, which is one of the five navigation buttons on the bottom bar.

## work-item-data

The <work-item-data> business component shows the current data values for the selected work item.

The component's internal controller invokes the [BPMWorkItemDetailsService.populateWorkItemData](#) operation to provide the data that is displayed from the [WorkItemDetailsModel](#) object.

If you want to customize the contents of the <work-item-data> business component, see [Customizing a Business Component](#) for more information.

### Usage

```
<work-item-data
 selected-work-item="object"
 [template-path="string"]>
</work-item-data>
```

### Attributes

Name	Type	Description
selected-work-item	<a href="#">WorkItem</a> object	Currently selected work item
template-path	String	(Optional) Relative pathname of a custom template file to use in place of the internal template.

## Example

The `<work-item-data>` business component is used:

- on the [WORK ITEM DETAILS](#) tab of the Component Showcase application.
- in the My Work App application, when you select a work item in a Work View, then click **DETAILS** in the footer menu, then click **Data**.

In each case, `<work-item-data>` is called from the [work-item-detail-nav](#) business component.

## work-item-detail-nav

The `<work-item-detail-nav>` business component provides detailed information about a selected work item, using different tabs to display different types of information.

By default, the following tabs are provided, each of which invokes a further business component to display the required information about the selected work item.

Tab	Invokes Business component
Summary	<a href="#">bpm-summary-view</a>
Data	<a href="#">work-item-data</a>
Attributes	<a href="#">bpm-attributes-view</a>
Processes	<a href="#">bpm-process-view</a>
Documents	<a href="#">documents-view</a>



If you want to change which of these tabs are displayed, or the order in which they are displayed, use the custom template instead of the internal template.

If you want to customize the contents of the `<work-item-detail-nav>` business component, see [Customizing a Business Component](#) for more information.

## Usage

```
<work-item-detail-nav
 tabs="object"
 selected-work-item="object"
 [template-path="string"]>
</work-item-detail-nav>
```

## Attributes

Name	Description
tabs	An object that contains the title of each tab to be rendered.
selected-work-item	An object containing the work item for which details are to be shown.
template-path	(Optional) A string containing the relative pathname of a custom template file to use in place of the internal template.

### Example

The `<work-item-detail-nav>` business component is used:

- on the [WORK ITEM DETAILS](#) tab of the Component Showcase application.
- in the My Work App application, when you select a work item in a Work View, then click **DETAILS** in the footer menu.

## workListData Object

`workListData` is the object used to provide work list data to the `<work-list>` business component.

### Attributes

Name	Type	Description
<code>guid</code>	String	GUID of the BPM user whose work list is to be displayed
<code>workViewId</code>	String	(Optional) ID of the work view that is to be displayed.
<code>pageSize</code>	Number	Number of work items to be displayed in each page of the work list.
<code>workItemType</code>	String	Type of layout to be used to display the work list - either card, list or table.
<code>workItems</code>	Array	List of work items in the specified work list. This array is populated by either the <a href="#">getWorklistItems</a> or <a href="#">getWorklistItemsForView</a> BPMWorklistService operations.
<code>start</code>	String	Start position in the <code>workItems</code> array.
<code>formDiv</code>	String	Name of the <code>&lt;div&gt;</code> element that is to be used to render the contents of the work list.
<code>caseReference</code>	String	Case reference for which work items should be displayed.

See [work-list](#) for more information about how the `workListData` object is used.

## workViewData Object

`workViewData` is the object used to provide work list data to the `<work-views>` business component.

### Attributes

Name	Type	Description
<code>workList.name</code>	String	
<code>hideLeftMenu</code>	String	
<code>start</code>	String	Start position in the <code>workItems</code> array

Name	Type	Description
pageSize	Number	

See [work-list](#) for more information about how the workListData object is used.

# Business Component Services

Component services are used to perform BPM-related functions that would require several calls to the lower-level objectAPI. They can be used to extend or change existing business components, or to build new custom components.

Component services are fully functional AngularJS services that can be consumed by AngularJS clients. They make it much easier to perform certain BPM functions that require the use of multiple calls when done directly from the objectAPI - for example, opening a work item and displaying a form from a pageflow:

- Using the objectAPI, you must perform [this sequence of calls](#).
- Using the [BPMWorklistService](#), you can do the same thing with a single call:

```
BPMWorklistService.openWorkItem(openWorkitemServiceModel, callback)
```

where openWorkitemServiceModel is an object containing the ID of the work item to be opened.

Component services:

- are supplied as part of the TIBCO Component Framework library (tcf.nocache.js). They must be included in your application, but cannot be modified
- are compiled for optimal load time and best performance.
- only provide a subset of the functionality that is available from using the objectAPI directly.

## Invoking a Business Component Service Operation From Your Application

The HTML document from which you want to invoke the component service operation must:

- include the following libraries. The tcf.nocache.js library contains the controllers and underlying framework logic used by each component service. The other libraries provide additional core functions used by component services.

```
<script type="text/javascript" src="/apps/app-cdn/tcf/tcf.nocache.js"></script>
<script type="text/javascript" src="/apps/app-cdn/jquery/jquery-3.1.1.js"></script>
<script src="/apps/app-cdn/angular/1.6.0/angular.min.js"></script>
<script src="/apps/app-cdn/angular-ui-grid/ui-grid.min.js"></script>
```



A lightweight version of the TIBCO Component Framework library is available, which can be used to improve resource load times. For more information, see [Using the Lightweight TIBCO Component Framework](#).

- include any additional JavaScript or CSS files that need to be loaded in the application. They must be loaded using the collectDependencies() method inside a script tag in the HTML HEAD section. For example:

```
function collectDependencies() {
 // any additional CSS or JS files that need to be downloaded can be provided
 here as a key, url map
 cap.depend("MyWorkCtrl", "MyWorkCtrl.js", false, false);
}
```

- include the angular module to be loaded by the application. It is specified using the onOCFAppLoad() method inside a script tag in the HTML HEAD section. For example:

```
function onOCFAppLoad(moduleName) {
 // which module to be used for bootstrapping
 return ["MyWork"];
}
```

- attach an application controller to the DOM.

- provide a UI element to invoke the required operation.

The associated application controller must:

- specify a dependency on the required component service, that is, a dependency on the Angular module 'TibcoFramework'.
- define and set the parameters required by the operation.
- invoke the operation, passing in any required parameters, including a callback function to handle the operation response.

### Example - Opening a Work Item

The **OPEN WORK ITEM** menu option in the Component Showcase application demonstrates how to invoke the `BPMWorklistService.openWorkItem` operation.

**OPEN WORK ITEM** provides a two-pane user interface:

- In the left-hand pane, you can enter the ID and, optionally, version number of a work item, then **OPEN** it.
- The right-hand pane provides a tabbed display in which you can view each opened work item's form, and **Cancel**, **Close** or **Submit** it.

The `component_showcase/component-demos/open-workitem/OpenWorkItem.html` file:

- includes the required libraries and the source for the application controller.  

```
<script type="text/javascript" src="/apps/app-cdn/tcf/tcf.nocache.js"></script>
<script type="text/javascript" src="/apps/app-cdn/jquery/jquery-3.1.1.js"></script>
<script src="/apps/app-cdn/angular/1.6.0/angular.min.js"></script>
<script src="/apps/app-cdn/angular-ui-grid/ui-grid.min.js"></script>
<script src="js/OpenWorkItemSample.js"></script>
```
- attaches the `OpenWorkItemSampleCtrl` controller to the DOM.  

```
<body layout="column" ng-controller="OpenWorkItemSampleCtrl" ng-cloak >
```
- provides an **OPEN** button that will be used to invoke the `BPMWorklistService.openWorkItem` operation.  

```
<md-button ng-disabled="sampleForm.$invalid" type="submit" class="md-raised md-primary">Open</md-button>
```

The `component_showcase/component-demos/open-workitem/js/OpenWorkItemSample.js` file defines the `OpenWorkItemSampleCtrl` controller, which has dependencies on the `BPMWorklistService` and `BPMLoginService` service.

```
(function() {
 var componentShowcase = angular.module("componentShowcase",
 ['TibcoFramework']);
 componentShowcase.controller("OpenWorkItemSampleCtrl",
 function($rootScope, $scope, BPMLoginService, BPMWorklistService) {
```



The `OpenWorkItemSampleCtrl` controller defines an open function that is invoked when the user clicks the OPEN button.

```
$scope.open = function() {
 for (var i = 0; i < $scope.workItemFormTabs.length; i++) {
 if ($scope.workItemFormTabs[i].id == $scope.id)
 return;
 }
}
```

The open function:

- defines a config object to set the parameters required by the `openWorkItem` operation.

```
}
 var config = {};
 config.id = $scope.id;
 config.version = $scope.version;

 config.guid = $rootScope.guid;

 config.formDiv = "formWorkDiv#" + $scope.id;

 $scope.workItemFormTabs.push(config);
```

- invokes the `openWorkItem` operation, passing in the config object and a *callback* function that displays the work item form.

```
if (BPMWorklistService) {
 BPMWorklistService.openWorkItem(config, {
 onSuccess : function() {

 $scope.workItemFormTabs.splice($scope.workItemFormTabs.indexOf(config), 1);

 $scope.$apply("");
 },
 onFailure : function() {

 $scope.workItemFormTabs.splice($scope.workItemFormTabs.indexOf(config), 1);

 $scope.$apply("");
 }
 });
}
```

The GUID of the currently logged in user is one of the required parameters for `openWorkItem`. The controller obtains this GUID by using a separate invocation of the `BPMLoginService.getExtendedUserInfo` operation. This is why the controller also has a dependency on the `BPMLoginService`.



```
BPMLoginService.getExtendedUserInfo({
 onSuccess : function(userInfo) {

 $rootScope.guid = userInfo.guid;
 }
});
```

## Suppressing Errors Displayed by Business Component Services

Invoking a component service operation may result in an error message being displayed to the end-user.

You can, however, prevent an error message from being displayed by the component service operation so that you can handle the exception yourself. This is done by passing `useDefault=false` in the callback. For example:

```
CaseInformationService.getCaseInformation(summary,
{
 useDefault: false,
 onSuccess: function () {
```

•  
•  
•

## Using the Lightweight TIBCO Component Framework

A *lightweight* version of the TIBCO Component Framework is available.

The lightweight version:

- supports the use of [business component services](#) and the [objectAPI](#).
- does NOT support the use of [business components](#).
- removes the dependency on the Angular Material library. The business components in TIBCO Component Framework are based on Angular Material. The service's methods also use Angular Material to display alert dialogs and confirmation messages. Therefore, since the lightweight framework does not load Angular Material, the default error messages that appear when you use TIBCO Component Framework are not available with the lightweight version. If you want to show the error messages, implement the `onFailure()` callback method for the service calls and show your own alert dialogs.

Applications that do not use business components can use the lightweight TIBCO Component Framework (`tcfservices.nocache.js`) instead of the original framework (`tcf.nocache.js`), to benefit from improved resource load times. (Applications that use business components must continue to use the original version of the TIBCO Component Framework (`tcf.nocache.js`)).

### Procedure

1. In Application Management, edit the `index.html` file (or other host file) of your existing deployed application (where you have used `tcf.nocache.js`):
  - a) Add the following statement at the end of the `<script>` tags in the HTML HEAD section:
 

```
<script type="text/javascript" src="/apps/app-cdn/tcfservices/tcfservices.nocache.js"></script>
```
  - b) Comment out the line:
 

```
<script type="text/javascript" src="/apps/app-cdn/tcf/tcf.nocache.js"></script>
```
2. Save the changes.
3. Analyze your service calls and make sure to report error messages to the user using the `onFailure()` method.  
As explained above, default error dialogs are not displayed to the user since Angular Material is not used.
4. Publish the modified application.

## Business Component Services Reference

### BPMBusinessActionsService

## BizActionHandler

BizActionHandler is the object model used by the BPMBusinessActionsService.startBizAction operation to provide the calling application with details of any event that occurs while processing the business action.

### Attributes

Name	Type	Description
response	Object	An object containing the data for the started business action. The object type depends on the event that has occurred, as shown below.
form	Object	Current form instance.
formContainerId	String	ID of the <code>Div</code> element containing the currently opened form.

### Response Object Types

Event Name	Response Object Type
bs.START	<a href="#">PageResponse</a>
bs.FORM_LOAD	<a href="#">BusinessServiceType</a>
bs.UPDATE	<a href="#">PageResponse</a>
bs.CANCEL	<a href="#">CancelBusinessServiceResponseType</a>
bs.COMPLETE	<a href="#">PageResponse</a>

### Methods

Method Name	Description
cancelPageflow()	Cancel the processing pageflow and destroy the associated form.

See [startBizAction](#) for specific information on how the startBizAction operation uses the BizActionHandler object model as part of its callback function .

## BusinessActionsServiceModel

BusinessActionsServiceModel is the object model used by BPMBusinessActionsService operations to pass data in service requests and responses.

### Attributes

Name	Type	Description
actionsCount	Number	Number of business actions available, populated by either a <a href="#">fetchTopCategories</a> or <a href="#">fetchChildCategories</a> call.
actionGroups	Array	<p>List of objects containing the available categories and their business actions. This attribute is:</p> <ul style="list-style-type: none"> <li>populated by a <a href="#">fetchTopCategories</a> call with the list of all available categories and, for each top-level category, the business actions in that category.</li> <li>passed as a parameter to a <a href="#">fetchChildCategories</a> call, defining the child categories for which the list of available business actions should be returned.</li> <li>populated by a <a href="#">fetchChildCategories</a> call with the list of business actions available in the specified child category.</li> </ul>

See the following topics for specific information on how BPMBusinessActionsService operations use the BusinessActionsServiceModel object model:

- [fetchTopCategories](#)
- [fetchChildCategories](#)

## StartBusinessActionServiceModel

BusinessActionsServiceModel is the object model used by the BPMBusinessActionsService.startBizAction operation to pass data in its request and response when starting a selected business action.

### Attributes

Name	Type	Description
businessActionItem Data	<a href="#">BusinessServiceTemplate</a> object	Details of the business action. This object can be obtained from <a href="#">BusinessActionsServiceModel.actionGroups</a> populated by a <a href="#">fetchTopCategories</a> or <a href="#">fetchChildCategories</a> call.
channelId	String	<p>ID of the presentation channel to be used to display the business action form.</p> <p>Default value = <code>openspaceGWTPull_DefaultChannel</code></p>

Name	Type	Description
channelType	String	Enumerated value defining the channel type (technology) associated with the specified channelId. See <a href="#">Identifying the Client Channel in a Service Call</a> for more information.  Default value = openspaceChannel.
formDiv	String	ID of the Div element in which to render the business action form.
localeKey	String	Locale to be used to display the business action form.  Default value = en_US
payload	String	Payload for the business action.

## fetchChildCategories

Get a list of the business actions available in the specified child categories.

You can use [fetchTopCategories](#) to get a list of all available categories and, for each top-level category, the business actions in that category.


### Required System Actions

listBusinessServices

### Usage

```
BPMBusinessActionsService.fetchChildCategories(actionGroups, callback)
```

### Parameters

Parameter	Type	Description
actionGroups	<a href="#">BusinessActionsServiceModel.actionGroups</a> object	<p>Data model to be used by the service request. The response populates:</p> <ul style="list-style-type: none"> <li><i>BusinessActionsServiceModel.actionsCount</i> with the total number of available business actions in the specified categories.</li> <li><i>BusinessActionsServiceModel.actionGroups</i> with the details of the available business actions in the specified categories.</li> </ul> <div>  <p><i>BusinessActionsServiceModel.actionGroups</i> is an array of objects that maps each category (<a href="#">Category</a>) to its actions (<a href="#">BusinessServiceTemplate</a>).</p> </div>
callback	Function	Callback function to handle success or failure results from the request.

## fetchTopCategories

Get a list of all available categories and, for each top-level category, the business actions in that category. You can use [fetchChildCategories](#) to obtain the list of available business actions in each child category.


### Required System Actions

listBusinessServices

### Usage

```
BPMBusinessActionsService.fetchTopCategories(BusinessActionsServiceModel, callback)
```

### Parameters

Parameter	Type	Description
<i>BusinessActionsServiceModel</i>	<a href="#">BusinessActionsServiceModel</a> object	<p>Data model to be used by the service request. The response populates:</p> <ul style="list-style-type: none"> <li><i>BusinessActionsServiceModel.actionsCount</i> with the total number of available business actions.</li> <li><i>BusinessActionsServiceModel.actionGroups</i> with the details of the available categories.</li> </ul> <div>  <p><i>BusinessActionsServiceModel.actionGroups</i> is an array of objects that maps each category (<a href="#">Category</a>) to its actions (<a href="#">BusinessServiceTemplate</a>).</p> </div>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## startBizAction

Start a business action and render its form in a specified container element.

### Required System Actions

executeBusinessService

### Usage

```
BPMBusinessActionsService.startBizAction(StartBusinessActionServiceModel, callback)
```

### Parameters

Parameter	Type	Description
<i>StartBusinessActionServiceModel</i>	<a href="#">StartBusinessActionServiceModel</a> object	Data model to be used by the service request. The response renders the business action form in the specified <i>StartBusinessActionServiceModel.formDiv</i> element.
<i>callback</i>	Function	Callback function to handle success results, failure results and events returned by the request.

## Callback Function

The callback function of the `BPMBusinessActionsService.startBizAction` operation supports the following methods:

- `onSuccess: function(result)`: returns the business action data (in *result*) if the call succeeds.
- `onFailure: function(exception)`: returns the exception thrown (in *exception*) if the call fails.
- `onEvent: function(eventName, bizActionHandler)`: notifies the calling application of any event that occurs while processing the business action. `onEvent` takes the following parameters:
  - *eventName* identifies the event that has occurred. One of the following strings is returned.
    - `bs.START`
    - `bs.FORM_LOAD`
    - `bs.UPDATE`
    - `bs.CANCEL`
    - `bs.COMPLETE`
  - *BizActionHandler* is an object that provides the data for the started business action and the form data for the current page. See [BizActionHandler](#).

For example:

```
if (BPMBusinessActionsService){
 BPMBusinessActionsService.startBizAction(businessActionTemplate,
 {onSuccess: function(result){
 },
 onFailure: function(exception){
 },
 onEvent: function(eventName, bizActionHandler){
 console.log("onEvent:"+ eventName);

 // Add any desired event handling code here. You can cancel the
 // processing pageflow and destroy the associated form using
 // the cancelPageflow method on the bizActionHandlerobject:
 //
 // bizActionHandler.cancelPageflow();
 }
 });
}
```

## BPMCaseEventViewService

The `BPMCaseEventViewService` provides an operation that retrieves case events.

### CaseEventViewModel

`CaseEventViewModel` is the object model used by `BPMCaseEventViewService` operations to pass data in service requests and responses about case events.

#### Attributes

Name	Type	Description
<code>eventDataList</code>	<code>Array&lt;<a href="#">CaseAuditEntry</a>&gt;</code>	Case events returned by the <a href="#">getSearchCaseReport</a> operation.
<code>caseReference</code>	<code>String</code>	Case reference.

See the following topic for specific information on how [BPMCaseEventViewService](#) operations use the `CaseEventViewModel` object model:

- [getSearchCaseReport](#)

## getSearchCaseReport

Lists all events associated with the specified case.

### Required System Actions

readGlobalData

### Usage

```
BPMCaseEventViewService.getSearchCaseReport(model, callback)
```

### Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CaseEventViewModel</a>	<p>The case model to be passed to the \$scope object.</p> <p>You must provide a case reference to identify the case for which the case events are to be returned.</p> <p>The response populates the <a href="#">CaseEventViewModel.eventDataList</a> object with the case events.</p>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

### Sample Usage

```
// Create a model with required properties
var model = {};
model.caseReference = "BDS-1-com.example.claimbom.Policy-3-0"; // case reference

//Actual Service call
BPMCaseEventViewService.getSearchCaseReport(model, {
 onSuccess: function(model)
 {
 console.log("Events for caseRef " + model.caseReference + " are " +
model.eventDataList);
 }
});
```

## BPMChartService

The `BPMChartService` provides operations that are used to display charts. These operations return chart data in a tabular format, which can be used by charting libraries like Highcharts.

`BPMChartService` operations are used by the [<bpm-highcharts>](#) business component.



## ChartModel

ChartModel is the object model used by the BPMChartService operations to pass data in their request and response when obtaining the requested audit data.

### Attributes

Name	Type	Description
data	ChartData object	Audit data returned by BPMChartService operations.
dataPayload	DataPayload Object	The payload passed into the BPMChartService operations.

## getChart

Aggregates the data in an Event Collector *area* (work items, cases, process templates, process instances, or events) to obtain audit information about the area.

This operation aggregates, then returns, the audit information in a tabular format, which is consumable by charting libraries such as Highcharts. (The [<bpm-highcharts>](#) business component, which uses this service operation, uses the Highcharts library to display charts.)

### Usage

```
BPMChartService.getChart(chartModel, callback)
```

### Parameters

Parameter	Type	Description
<i>chartModel</i>	ChartModel object	Data model to be used by the service request. The response populates <i>ChartModel.data</i> with the audit data requested.
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## getChartData

Returns audit information for a specified Event Collector *area* (work items, cases, process templates, process instances, or events) based on specified filter criteria. For example, you can request all outstanding work items whose status is OFFERED.

This operation returns the audit information in a tabular format, which is consumable by charting libraries such as Highcharts.

### Usage

```
BPMChartService.getChartData(ChartModel, callback)
```

### Parameters

Parameter	Type	Description
<i>chartModel</i>	ChartModel object	Data model to be used by the service request. The response populates <i>ChartModel.data</i> with the audit data requested.

Parameter	Type	Description
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## BPMCommentService

The BPMCommentService provides operations that are used to add or retrieve comments for work items, process instances, or cases.

BPMCommentService operations are used by the [<bpm-comments>](#) business component.

## CommentModel

CommentModel is the object model used by BPMCommentService operations to pass data in their request and response when adding or retrieving comments.

### Attributes

Name	Type	Description
data	data object	The data object is used by BPMCommentService operations to store the result from 'get' and 'add' comment operations.  For 'get' operations, the data is of type <a href="#">CommentAudit</a> . For 'add' operations, the data is of type <a href="#">DoubleResponse</a> .
comment	comment object	The comment object contains the comment being added to the work item, process instance, or case using the 'add' comment operations.  The data in the comment object is of type <a href="#">baseComment</a> .
parentId	String	Identifies a parent comment. This can be used by the 'add' comment operations to "reply" to a previous comment. This allows for a hierarchical structure of comments.
<b>For work item comments:</b>		
workItemId	String	Identifies a work item for which comments are being added or retrieved.
procId	String	A process instance ID, which can be used in conjunction with a workItemId and appActId to add a comment to a work item using <a href="#">BPMCommentService.addWorkItemComment</a> .
appActId	String	An application activity ID, which can be used in conjunction with a workItemId and procId to add a comment to a work item using <a href="#">BPMCommentService.addWorkItemComment</a> .
<b>For process instance comments:</b>		

Name	Type	Description
procId	String	A process instance ID, which identifies the process instance for which a comment is being added using <a href="#">BPMCommentService.addProcessInstanceComment</a> .
reference	String	A process instance ID, which identifies the process instance for which comments being retrieved using <a href="#">BPMCommentService.getProcInstComments</a> .
<b>For case comments:</b>		
caseReference	String	Identifies a case for which comments are being added or retrieved.
className	String	The name of the case class for which comments are being added or retrieved.
majorVersion	String	Version number of the case class.
managedObjectId	String	The managed object ID for the case.

## addCaseComment

Adds a comment to a case.

### Required System Actions

queryAudit

### Usage

```
BPMCommentService.addCaseComment(model, callback)
```

### Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CommentModel</a>	<p>The data model to be passed to the \$scope object.</p> <p>The comment to add is passed in <a href="#">CommentModel.comment</a>.</p> <p>Identify the case by providing either:</p> <ul style="list-style-type: none"> <li><a href="#">CommentModel.caseReference</a></li> </ul> <p>or</p> <ul style="list-style-type: none"> <li><a href="#">CommentModel.className</a>, <a href="#">CommentModel.majorVersion</a>, and <a href="#">CommentModel.managedObjectId</a></li> </ul> <p>To add a comment that is subordinate to a previous comment, also provide the parent ID for the previous comment in <a href="#">CommentModel.parentId</a>.</p>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## addProcessInstanceComment

Adds a comment to a process instance.

### Required System Actions

queryAudit

### Usage

```
BPMCommentService.addProcessInstanceComment(model, callback)
```

### Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CommentModel</a>	<p>The data model to be passed to the \$scope object.</p> <p>The comment to add is passed in <a href="#">CommentModel.comment</a>.</p> <p>Identify the process instance by providing a process instance ID in <a href="#">CommentModel.procId</a>.</p> <p>To add a comment that is subordinate to a previous comment, also provide the parent ID for the previous comment in <a href="#">CommentModel.parentId</a>.</p>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## addWorkItemComment

Adds a comment to a work item.

### Required System Actions

queryAudit

### Usage

```
BPMCommentService.addWorkItemComment(model, callback)
```

## Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CommentModel</a>	<p>The data model to be passed to the \$scope object.</p> <p>The comment to add is passed in <a href="#">CommentModel.comment</a>.</p> <p>Also pass either:</p> <ul style="list-style-type: none"> <li>• <a href="#">CommentModel.workItemId</a> to identify the work item for which a comment is being added.</li> </ul> <p>or</p> <ul style="list-style-type: none"> <li>• <a href="#">CommentModel.workItemId</a>, <a href="#">CommentModel.procId</a>, and <a href="#">CommentModel.appActId</a> to identify the work item, process instance, and application activity ID. This is preferable for performance reasons, as passing only the work item ID requires a SELECT on the ec_event view to add the comment.</li> </ul> <p>To add a comment that is subordinate to a previous comment, also provide the parent ID for the previous comment in <a href="#">CommentModel.parentId</a>. The parent ID is returned in <a href="#">CommentModel.parentId</a> when you add or get a comment for an entity.</p>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## getCaseComments

Retrieves comments that had been previously added to a case.

### Required System Actions

queryAudit

### Usage

```
BPMCommentService.getCaseComments(model, callback)
```

## Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CommentModel</a>	<p>The data model to be passed to the \$scope object.</p> <p>Identify the case by providing either:</p> <ul style="list-style-type: none"> <li>• <a href="#">CommentModel.caseReference</a></li> </ul> <p>or</p> <ul style="list-style-type: none"> <li>• <a href="#">CommentModel.className</a>, <a href="#">CommentModel.majorVersion</a>, and <a href="#">CommentModel.managedObjectId</a></li> </ul> <p>The response populates <a href="#">CommentModel.data</a>.</p>

Parameter	Type	Description
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

### getProcInstComments

Retrieves comments that had been previously added to a process instance.

#### Required System Actions

queryAudit

#### Usage

```
BPMCommentService.getProcInstComments(model, callback)
```

#### Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CommentModel</a>	The data model to be passed to the \$scope object. Identify the process instance by providing a process instance ID in <a href="#">CommentModel.reference</a> . The response populates <a href="#">CommentModel.data</a> .
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

### getWorkItemComments

Retrieves comments that had been previously added to a work item.

#### Required System Actions

queryAudit

#### Usage

```
BPMCommentService.WorkItemComments(model, callback)
```

#### Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CommentModel</a>	The data model to be passed to the \$scope object. Identify the work item by providing a work item ID in <a href="#">CommentModel.workItemId</a> . The response populates <a href="#">CommentModel.data</a> .
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## BPMEventViewService

The BPMEventViewService provides a single operation that allows you to obtain and display the audit trail for the process instance associated with the currently selected work item.

The BPMEventViewService.[getEventView](#) operation is used by the [bpm-event-view](#) business component.

### EventViewModel

EventViewModel is the object model used by the BPMEventViewService.getEventView operation to pass data in its request and response when obtaining the audit trail for the process instance associated with the currently selected work item.

#### Attributes

Name	Type	Description
eventDataList	Array	List of audit entries for the current processInstanceId, populated by a <a href="#">getEventView</a> call.
processInstanceId	String	ID of the process instance associated with the currently selected work item.

### getEventView

Get the audit trail for the process instance associated with the currently selected work item.


#### Required System Actions

None

#### Usage

```
BPMEventViewService.getEventView(EventViewModel, callback)
```

#### Parameters

Parameter	Type	Description
<i>EventViewModel</i>	<a href="#">EventViewModel</a> object	Data model to be used by the service request. The response populates <i>EventViewModel.eventDataList</i> with the audit trail for the current process instance.  <i>EventViewModel.eventDataList</i> is an array of type <a href="#">ProcessInstanceAuditEntry</a> .
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## BPMLoginService

The BPMLoginService provides user login related functions.

## LoginServiceModel

LoginServiceModel is the object model used by BPMLoginService operations to pass data in service requests and responses.

### Attributes

Name	Type	Description
cookie	String	Cookie value
authenticated	Boolean	Whether the user is authenticated or not. This attribute is populated by the response from a BPMLoginService. <a href="#">login</a> call. Clients should only read this value to check if the user is authenticated.
username	String	User's name
password	String	User's password
loginRedirect	String	URL to be redirected after successful authentication
logoutRedirect	String	User is redirected to this URL after logout. If not specified, the standard login page is displayed after logout.
doNotRedirect	Boolean	After BPMLoginService. <a href="#">login</a> is called, the service typically redirects to the sample Workapp application or another application, as configured using the loginRedirect attribute. If the doNotRedirect attribute is set to true, the service will not do a redirection. This is typically used when the login component is embedded within the custom application where no redirection is required after login. Default = false

See the following topics for specific information on how each BPMLoginService operation uses the LoginServiceModel object model:

- [login](#)
- [logout](#)

### getExtendedUserInfo

Get detailed information about the currently logged in BPM user.

#### Required System Actions

resolveResource

#### Usage

```
BPMLoginService.getExtendedUserInfo(callback)
```



## Parameters

Parameter	Type	Description
<i>callback</i>	Function	<p>Callback function to handle success or failure results from the request.</p> <p>The response returns an object of type <a href="#">XmlResourceDetail</a>, which provides extended information about the currently logged in BPM user.</p>

## getUserInfo

Get basic information about the currently logged in user.

### Usage

```
BPMLoginService.getUserInfo(callback)
```

## Parameters

Parameter	Type	Description
<i>callback</i>	Function	<p>Callback function to handle success or failure results from the request.</p> <p>The response returns information about the currently logged in user in an object containing the following attributes:</p> <ul style="list-style-type: none"> <li>• version</li> <li>• userName</li> <li>• cookie</li> <li>• digest</li> </ul>

## login

Log in to the custom client application.

### Required System Actions

None

### Usage

```
BPMLoginService.login(LoginServiceModel, callback)
```

## Parameters

Parameter	Type	Description
<i>LoginServiceModel</i>	<a href="#">LoginServiceModel</a> object	Data model to be used by the service request. The response populates <i>LoginServiceModel.authenticated</i> with the result.

Parameter	Type	Description
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## logout

Log out of the custom client application.

### Required System Actions

None

### Usage

```
BPMLoginService.logout(LoginServiceModel, callback)
```

### Parameters

Parameter	Type	Description
<i>LoginServiceModel</i>	<a href="#">LoginServiceModel</a> object	Data model to be used by the service request.
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## BPMWorkItemDetailsService

The BPMWorkItemDetailsService provides operations that allow you to display different types of information about the currently selected work item.

### WorkItemDetailsModel

WorkItemDetailsModel is the object model used by BPMWorkItemDetailsService operations to pass data in service requests and responses about work items.

### Attributes

Name	Type
<i>attributes</i>	Array
<i>bomJSPath</i>	String
<i>description</i>	String
<i>distributionStrategy</i>	String
<i>endDate</i>	String
<i>guid</i>	String
<i>id</i>	String

Name	Type
name	String
previewData	String
priority	Number
processDescription	String
processInstanceId	String
processName	String
selectedWorkItem	WorkItem
startDate	String
state	String
version	String
workItemAttributes	Array

See the following topics for specific information on how BPMWorkItemDetailsService operations use the WorkItemDetailsModel object model:

- [fetchWorkItem](#)
- [populateWorkItemAttributes](#)
- [populateWorkItemData](#)
- [populateWorkItemDetails](#)
- [populateWorkItemProcessDetails](#)

## fetchWorkItem

Get details of a specific work item.

### Required System Actions

viewWorkList

### Usage

```
BPMWorkItemDetailsService.fetchWorkItem(WorkItemDetailsModel, WorkItemID, WorkItemVersion, callback)
```

### Parameters

Parameter	Type	Description
<i>WorkItemDetailsModel</i>	<a href="#">WorkItemDetailsModel</a> object	Data model to be used by the service request.

Parameter	Type	Description
<i>WorkItemID</i>	String	ID of the work item to be fetched.
<i>WorkItemVersion</i>	String	Version of the work item to be fetched.
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## populateWorkItemAttributes

### Required System Actions

viewWorkList

### Usage

```
BPMWorkItemDetailsService.populateWorkItemAttributes(WorkItemDetailsModel, callback)
```

### Parameters

Parameter	Type	Description
<i>WorkItemDetailsModel</i>	<a href="#">WorkItemDetailsModel</a> object	Data model to be used by the service request.
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## populateWorkItemData

### Required System Actions

None

### Usage

```
BPMWorkItemDetailsService.populateWorkItemData(WorkItemDetailsModel, callback, previewElementID)
```

### Parameters

Parameter	Type	Description
<i>WorkItemDetailsModel</i>	<a href="#">WorkItemDetailsModel</a> object	Data model to be used by the service request.
<i>callback</i>	Function	Callback function to handle success or failure results from the request.
<i>previewElementID</i>	String	

## populateWorkItemDetails

### Required System Actions

None

### Usage

```
BPMWorkItemDetailsService.populateWorkItemDetails(WorkItemDetailsModel)
```

### Parameters

Parameter	Type	Description
<i>WorkItemDetailsModel</i>	<a href="#">WorkItemDetailsModel</a> object	Data model to be used by the service request.

## populateWorkItemProcessDetails

### Required System Actions

None

### Usage

```
BPMWorkItemDetailsService.populateWorkItemProcessDetails(WorkItemDetailsModel)
```

### Parameters

Parameter	Type	Description
<i>WorkItemDetailsModel</i>	<a href="#">WorkItemDetailsModel</a> object	Data model to be used by the service request.

## BPMWorkItemHistoryService

The BPMWorkItemHistoryService provides a single operation that allows you to obtain and display the audit trail for the currently selected work item.

## WorkItemHistoryModel

WorkItemHistoryModel is the object model used by the BPMWorkItemHistoryService.getWorkItemHistory operation to pass data in its request and response when obtaining the audit trail for the currently selected work item.

### Attributes

Name	Type	Description
historyDataList	Array	List of audit entries for the current workItemId, populated by a <a href="#">getWorkItemHistory</a> call.
workItemId	String	ID of the currently selected work item.


## getWorkItemHistory

Get the audit trail for the currently selected work item.

### Usage

```
BPMWorkItemHistoryService.getWorkItemHistory(WorkItemHistoryModel, callback)
```

### Parameters

Parameter	Type	Description
<i>WorkItemHistoryModel</i>	<a href="#">WorkItemHistoryModel</a> object	Data model to be used by the service request. The response populates <i>WorkItemHistoryModel.historyDataList</i> with the audit trail for the currently selected work item.  <i>WorkItemHistoryModel.historyDataList</i> is an array of type <a href="#">WorkItemAudit</a> .
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## BPMWorklistService

The BPMWorklistService provides operations that allow you to display the contents of a user's work list or work view; to allocate, re-offer or open work items from the user's work list; or list all work items associated with a specified case.

BPMWorklistService operations are used by the [work-list](#) business component.


## OpenWorkitemServiceModel

OpenWorkitemServiceModel is the object model used by BPMWorkListService operations to pass data in service requests and responses when opening a work item.

### Required Attributes

Name	Type	Description
formDiv	String	ID of the Div element in which to render the work item form.
guid	String	GUID of the logged in user, used to authenticate and authorize the service request. The GUID can be obtained by using the BPMLoginService. <a href="#">getExtendedUserInfo</a> operation.
id	String	ID of the work item to be opened

### Optional Attributes

Name	Type	Description
channelId	String	ID of the presentation channel to be used to display the work item form. Default value = openspaceGWTPull_DefaultChannel
channelType	String	Enumerated value defining the channel type (technology) associated with the specified channelId. See <a href="#">Identifying the Client Channel in a Service Call</a> for more information. Default value = openspaceChannel.
localeKey	String	Locale to be used to display the work item form. Default value = en_US
version	String	Version of the work item. Default value = -1 (which means use the latest version).
workListViewId	String	ID of the work list view associated with this work item. ID of the work list view This is used to determine the work item to open next.  This attribute is used to determine the work item to open next, so can only be used with <a href="#">openNextWorkItem</a> . Default value = -1 (which means use the default work view).

See the following topics for specific information on how BPMWorkListService operations use the OpenWorkitemServiceModel object model:

- [openNextWorkItem](#)
- [openWorkItem](#)

## WorklistServiceModel

WorklistServiceModel is the object model used by BPMWorkListService operations to pass data in service requests and responses about work lists.

### Attributes

Name	Type	Description
templateData	Object	Object used to obtain data from and return data to the calling template.
templateData.workListViews	Array	List of work view objects, populated by a <a href="#">getViewsForResource</a> call.
templateData.workItems	Array	List of work item objects, populated by either a <a href="#">getWorklistItems</a> or <a href="#">getWorkListItemsForView</a> call.
templateData.startPosition	Number	Index of the first item in the list.
templateData.endPosition	Number	Index of the last item in the list.
templateData.totalItems	Number	Total number of items in the list.
templateData.customData	Object	

See the following topics for specific information on how BPMWorkListService operations use the WorklistServiceModel object model:

- [allocateWorkItem](#)
- [getViewsForResource](#)
- [getWorklistItems](#)
- [getWorkListItemsForGlobalData](#)
- [getWorkListItemsForView](#)
- [unallocateWorkItem](#)

## allocateWorkItem

Allocates one or more work items to specific users.

Use this operation to allocate work items from a list that has been previously populated using [getWorklistItems](#) or [getWorkListItemsForView](#).

This operation can only be used on work items that are in a state from which they can be allocated. See [Work Item State Transitions](#).

### Required System Actions

workItemAllocation



## Usage

```
BPMWorklistService.allocateWorkItem(idArray,resourceArray,worklistServiceModel,callback)
```

## Parameters

Parameter	Type	Description
<i>idArray</i>	Array	IDs of the work items to be allocated.
<i>resourceArray</i>	Array	GUIDS of the users to whom each work item should be allocated.  Each work item specified in an <i>idArray</i> element is allocated to the user specified in the <i>resourceArray</i> element with the matching index number. For example, a work item specified in <i>ids[4]</i> is allocated to the GUID specified in <i>guids[4]</i> .
<i>worklistServiceModel</i>	<a href="#">WorklistServiceModel</a> object	Data model to be used by the service request. The response updates successfully allocated work items in <i>worklistServiceModel.workItems</i> .
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## getViewsForResource

Return the list of work views to which the calling user has access.

## Required System Actions


viewWorkList

## Usage

```
BPMWorklistService.getViewsForResource(startPosition,numberOfItems,worklistServiceModel,callback)
```

## Parameters

Parameter	Type	Description
<i>startPosition</i>	Number	Position in the list of available work views (zero-based) from which to start the results list.
<i>numberOfItems</i>	Number	Number of work views to include in the results list.

Parameter	Type	Description
<i>worklistServiceModel</i>	<a href="#">WorklistServiceModel</a> object	<p>Data model to be used by the service request. The response populates <i>worklistServiceModel</i>.<i>templateData</i>.<i>workListView</i>s with the details of each work view that the calling user is permitted to access.</p> <p> <i>worklistServiceModel</i>.<i>templateData</i>.<i>workListView</i>s is an array of type <a href="#">GetViewsForResourceResponseType</a>.</p>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

### getWorklistItems

Gets the work item list for an organization model entity.


#### Required System Actions

viewWorkList

#### Usage

```
BPMWorklistService.getWorklistItems(entityType, entityGuid, start, count,
worklistServiceModel, callback)
```

#### Parameters

Parameter	Type	Description
<i>entityType</i>	String	One of these <a href="#">enumerated values</a> , defining the type of organization model entity for which data should be returned.
<i>entityGuid</i>	String	GUID of the organization model entity for which data should be returned.
<i>start</i>	Number	Position in the work item list (zero-based) from which to start the results list.
<i>count</i>	Number	Number of work items to include in the results list.
<i>worklistServiceModel</i>	<a href="#">WorklistServiceModel</a> object	<p>Data model to be used by the service request. The response populates <i>worklistServiceModel</i>.<i>templateData</i>.<i>workItems</i> with the contents of the work list for the specified <i>entityGuid</i>.</p> <p> <i>worklistServiceModel</i>.<i>templateData</i>.<i>workItems</i> is an array of type <a href="#">GetWorkListItemsResponseType</a>.</p>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## getWorkListItemsForGlobalData

Lists all work items associated with the specified case.


### Required System Actions

readGlobalData

### Usage

```
BPMWorklistService.getWorkListItemsForGlobalData(startPosition, numberOfItems, globalDataRef, wlsvcModel, callback)
```

### Parameters

Parameter	Type	Description
<i>startPosition</i>	String	Position in the work item list (zero-based) from which to start the results list.
<i>numberOfItems</i>	String	Number of items from the work view to include in the results list.
<i>globalDataRef</i>	String	Case reference for which associated work items are to be returned.
<i>wlsvcModel</i>	<a href="#">WorklistServiceModel</a> object	<p>Data model to be used by the service request. The response populates <i>wlsvcModel.templateData.workItems</i> with the work items associated with the specified <i>globalDataRef</i>.</p> <div>  <i>wlsvcModel.templateData.workItems</i> is an array of type <a href="#">GetWorkListItemsForGlobalDataResponseType</a>. </div>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

### Sample Usage

```
// Create a model with required properties
var model = {};
model.templateData = {};
var globalDataRef = "BDS-1-com.example.claimbom.Policy-3-0"; //case reference
var startPosition = "0"; // start index
var numberOfItems = "20"; // number of items to return

//Actual Service call
BPMWorklistService.getWorkListItemsForGlobalData(startPosition, numberOfItems,
globalDataRef, model, {
 onSuccess: function (model) {
 console.log("Workitems for caseRef "+ globalDataRef+" are "+
model.templateData.workItems); //Service call assigns the work items to
model.templateData.workItems
 }
});
```

## getWorkListItemsForView

Gets the work item list for a specific work view.

This operation must be called as a user who has access to the specified work view. See [Work List View Access](#).


### Required System Actions

viewWorkList

### Usage

```
BPMWorklistService.getWorkListItemsForView(viewId, startPosition, numberOfItems, worklistServiceModel, callback)
```

### Parameters

Parameter	Type	Description
<i>viewId</i>	Number	ID of the work view from which to get the work item list.
<i>startPosition</i>	Number	Position in the work item list (zero-based) from which to start the results list.
<i>numberOfItems</i>	Number	Number of items from the work view to include in the results list.
<i>worklistServiceModel</i>	<a href="#">WorklistServiceModel</a> object	<p>Data model to be used by the service request. The response populates <i>worklistServiceModel</i>.<i>templateData</i>. <i>workItems</i> with the contents of the work list for the specified <i>viewId</i>.</p> <div>  <i>worklistServiceModel</i>.<i>templateData</i>. <i>workItems</i> is an array of type <a href="#">GetWorkListItemsForViewResponseType</a>. </div>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## openNextWorkItem

Open the next work item in the work list and render its form in a specified container element.

### Usage

```
BPMWorklistService.openNextWorkItem(openWorkitemServiceModel, callback)
```

### Parameters

Parameter	Type	Description
<i>openWorkitemServiceModel</i>	<a href="#">OpenWorkitemServiceModel</a> object	Data model to be used in the request. The response renders the work item form in the specified <i>openWorkitemServiceModel</i> . <i>formDiv</i> element.

Parameter	Type	Description
<i>callback</i>	Function	A callback function to handle success or failure results from the request.

## openWorkItem

Open a work item and render its form in a specified container element.

### Required System Actions

None

### Usage

```
BPMWorklistService.openWorkItem(openWorkitemServiceModel, callback)
```

### Parameters

Parameter	Type	Description
<i>openWorkitemServiceModel</i>	<a href="#">OpenWorkitemServiceModel</a> object	Data model to be used in the request. The response renders the work item form in the specified <i>BusinessActionsServiceModel.formDiv</i> element.
<i>callback</i>	Function	A callback function to handle success or failure results from the request.

## unallocateWorkItem

Re-offers one or more currently allocated work items to their original offer sets.

Use this operation to re-offer work items from a list that has been previously populated using [getWorklistItems](#) or [getWorkListItemsForView](#).

This function can only be used on work items that are in a state from which they can be unallocated. See [Work Item State Transitions](#).

### Required System Actions

workItemAllocation

### Usage

```
BPMWorklistService.unallocateWorkItem(idArray, worklistServiceModel, callback)
```

### Parameters

Parameter	Type	Description
<i>idArray</i>	Array	IDs of the work items to be re-offered.
<i>worklistServiceModel</i>	<a href="#">WorklistServiceModel</a> object	Data model to be used by the service request. The response updates successfully re-offered work items in <i>worklistServiceModel.workItems</i> .

Parameter	Type	Description
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## CaseDocumentService

The CaseDocumentService provides operations that allow you to obtain and display a list of case documents associated with a case reference or work item; delete, download or open a selected document from the previously obtained list; or delete orphaned case folders.

CaseDocumentService operations are used by the [documents-view](#) business component.

## CaseDocumentServiceModel

CaseDocumentServiceModel is the object model used by CaseDocumentService operations to pass data in service requests and responses.

### Attributes

Name	Type	Description
caseRef	String	Case reference
downloadDocumentDivId	String	ID of the Div element in which to download a document
guid	String	GUID of the logged in user, used to authenticate and authorize the service request. The GUID can be obtained by using the BPMLoginService. <a href="#">getExtendedUserInfo</a> operation.
item	Array	List of case documents
openDocumentDivId	String	ID of the Div element in which to open a document
selectedDate	String	Date used to delete orphaned folders.
selectedDocument	DocumentItem	Currently selected case document
selectedWorkItem	WorkItem	Currently selected work item

See the following topics for specific information on how each CaseDocumentService operation uses the CaseDocumentServiceModel object model:

- [deleteDocument](#)
- [deleteOrphanedFolders](#)
- [downloadDocument](#)
- [fetchDocuments](#)
- [openDocument](#)

## deleteDocument

Delete the currently selected case document.

Use this operation to delete a selected case document from a list that has been previously populated using [fetchDocuments](#).

### Required System Actions

cmisAdmin

### Usage

```
CaseDocumentService.deleteDocument(model, callback)
```

### Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CaseDocumentServiceModel</a> object	Data model to be used by the service request. <i>model.selectedDocument</i> identifies the case document to be deleted.
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## deleteOrphanedFolders

Deletes orphaned case folders, that is, case folders whose case objects have been deleted.



Orphaned case folders may be deleted automatically when case objects are deleted, depending on the setting in the **deleteFolderOnUndeploymentOrCaseObjectDeletion** configuration parameter. For more information, see the "Case Folders" section in the *TIBCO ActiveMatrix BPM Case Data User's Guide*.

### Required System Actions

cmisAdmin

### Usage

```
CaseDocumentService.deleteOrphanedFolders(model, callback)
```

### Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CaseDocumentServiceModel</a> object	Data model to be used by the service request. The request must include <i>model.selectedDate</i> - Case folders associated with cases that were deleted on or before this date, are deleted.
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## Sample Usage

```
$scope.selectedDate = $scope.acceptDateFromUser;

//Actual Service call for deleteOrphanedFolders
CaseDocumentService.deleteOrphanedFolders($scope,{onSuccess: function()

{
 console.log("Orphaned folders before the date"+$scope.selectedDate+" deleted
successfully"); $scope.$apply("");
}
});
```

## downloadDocument

Download the currently selected case document.

Use this operation to download a selected case document from a list that has been previously populated using [fetchDocuments](#).

### Required System Actions

cmisAdmin

### Usage

```
CaseDocumentService.downloadDocument(model, callback)
```

### Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CaseDocumentServiceModel</a> object	Data model to be used by the service request. <ul style="list-style-type: none"> <li><i>model.selectedDocument</i> identifies the case document to be downloaded.</li> <li><i>model.downloadDocumentDivId</i> identifies the Div element to which the iframe for the document download should be attached.</li> </ul>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## fetchDocuments

Fetch the list of case documents associated with either the currently selected work item or a specified case reference.

### Required System Actions

cmisUser

### Usage

```
CaseDocumentService.fetchDocuments(model, callback)
```



## Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CaseDocumentServiceModel</a> object	Data model to be used by the service request. <ul style="list-style-type: none"> <li><i>model.caseRef</i> identifies the case reference for which to obtain the list of documents.</li> <li><i>model.item</i> identifies the array to be populated with the list of documents.</li> <li><i>model.selectedWorkItem</i> identifies the work item for which to obtain the list of documents</li> </ul>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## openDocument

Open the currently selected case document.

Use this operation to open a selected case document from a list that has been previously populated using [fetchDocuments](#).

### Required System Actions

None

### Usage

```
CaseDocumentService.openDocument(model, callback)
```

## Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CaseDocumentServiceModel</a> object	Data model to be used by the service request. <ul style="list-style-type: none"> <li><i>model.selectedDocument</i> identifies the case document to be opened.</li> <li><i>model.openDocumentDivId</i> identifies the Div element in which to open the document.</li> </ul>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## CaseInformationService

The CaseInformationService provides operations used to manage and obtain information about cases.



The CaseInformationService was introduced in ActiveMatrix BPM 4.0. However, the CaseInformationService was superseded by the [CaseManagementService](#) in ActiveMatrix BPM 4.1. CaseInformationService is still supported, but may be deprecated in the future. Therefore, you should use the operations in [CaseManagementService](#) instead, unless you have a specific requirement to use CaseInformationService.

## CaseInformationServiceModel

CaseInformationServiceModel is the object model used by CaseInformationService operations to pass data in service requests and responses about cases.

### Attributes

Name	Type	Description
caseActions	Array< <a href="#">CaseCategory</a> >	Case actions. A case action defines an action a user can perform on a case.
caseData	Array< <a href="#">CaseDataType</a> >	Case data. Case data is business data that is centrally managed by ActiveMatrix BPM and can therefore be accessed and updated by multiple BPM process applications.
caseRef	String	Case reference.
caseRefInformation	Array< <a href="#">CaseReferenceWithSummaryDetailsType</a> >	Case reference and its summary details.
guid	String	GUID of the logged in user, used to authenticate and authorize the service request. The GUID can be obtained by using the <a href="#">BPMLoginService.getExtendedUserInfo</a> operation.
selectedWorkItem	<a href="#">WorkItem</a>	Work item whose case information is to be retrieved.

See the following topics for specific information on how [CaseInformationService](#) operations use the CaseManagementModel object model:

- [getCaseData](#)
- [getCaseInformation](#)
- [listCaseActions](#)

### getCaseData

Retrieves case data for the given case reference.



The CaseInformationService.getCaseData operation was introduced in ActiveMatrix BPM 4.0, but has been superseded by the [CaseManagementService.getCaseData](#) operation, which was introduced in ActiveMatrix BPM 4.1. The CaseInformationService.getCaseData operation is still supported, but may be deprecated in the future. Therefore, you should use the [CaseManagementService.getCaseData](#) operation instead, unless you have a specific requirement to continue to use CaseInformationService.getCaseData.

### Required System Actions

readGlobalData

## Usage

```
CaseInformationService.getCaseData(model, caseRef, callback)
```

## Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CaseManagement Model</a>	The case model to be passed to the \$scope object. The response populates the <a href="#">CaseInformationServiceModel.caseDataTree</a> object with the case data.
<i>caseRef</i>	String	Identifies the case whose data is to be returned.
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## Sample Usage

```
scope.caseRef = "BDS-1-com.example.claimbom.Policy-3-0"; //accept a case reference
from user for which we need the case data

//Actual Service call
CaseInformationService.getCaseData(scope,scope.caseRef, {onSuccess: function()
{
 console.log("case data for the given case reference"+scope.caseRef+" is
"+scope.caseDataTree); //case data is returned by the service and assigned to
caseDataTree
}
});
```

## getCaseInformation

Retrieves a summary of the case information for a given work item or case reference.



The `CaseInformationService.getCaseInformation` operation was introduced in ActiveMatrix BPM 4.0, but has been superseded by the [CaseManagementService.getCaseInformation](#) operation, which was introduced in ActiveMatrix BPM 4.1. The `CaseInformationService.getCaseInformation` operation is still supported, but may be deprecated in the future. Therefore, you should use the [CaseManagementService.getCaseInformation](#) operation instead, unless you have a specific requirement to continue to use `CaseInformationService.getCaseInformation`.

## Required System Actions

readGlobalData

## Usage

```
CaseInformationService.getCaseInformation(model, callback)
```

## Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CaseManagement Model</a>	<p>The case model to be passed to the \$scope object.</p> <p>You must provide either one of the following:</p> <ul style="list-style-type: none"> <li>a <a href="#">CaseManagementServiceModel.WorkItem</a> object for the work item whose associated cases and case information are to be retrieved. The <a href="#">CaseManagementServiceModel.WorkItem</a> must be assigned to an object named <a href="#">selectedWorkItem</a> on the scope, then the scope must be passed to the service.</li> <li>a <a href="#">CaseManagementServiceModel.caseRef</a> for the case whose information is to be retrieved.</li> </ul> <p>The response populates the <a href="#">CaseManagementServiceModel.caseRefInformation</a> object with an array of <a href="#">CaseReferenceWithSummaryDetailsType</a> objects.</p> <p>The sample usage below shows how each of these can be passed.</p>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## Sample Usage

```
scope.selectedWorkItem = scope.acceptWorkItemFromUser; //accept an workItem from
user for which we will find all the cases and their case summary
//OR
scope.caseRef = "BDS-1-com.example.claimbom.Policy-3-0"; //accept a case reference
from user for which we need the case summary

//Actual Service call
CaseInformationService.getCaseInformation(scope, {onSuccess: function()
{
 console.log("Case Reference Summary for caseRef "+scope.caseRef+" is
"+scope.caseRefInformation); //Service call assigns the case summary to
scope.caseRefInformation
}
});
```

## listCaseActions

Lists all the case actions available for a given case reference.



The [CaseInformationService.listCaseActions](#) operation was introduced in ActiveMatrix BPM 4.0, but has been superseded by the [CaseManagementService.listCaseActions](#) operation, which was introduced in ActiveMatrix BPM 4.1. The [CaseInformationService.listCaseActions](#) operation is still supported, but may be deprecated in the future. Therefore, you should use the [CaseManagementService.listCaseActions](#) operation instead, unless you have a specific requirement to continue to use [CaseInformationService.listCaseActions](#).

## Required System Actions

readGlobalData

## Usage

```
CaseInformationService.listCaseActions(model, caseRef, callback)
```

## Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CaseManagement Model</a>	The case model to be passed to the \$scope object. The response populates the <a href="#">CaseManagementModel.caseActions</a> object with the case actions.
<i>caseRef</i>	String	Identifies the case for which case actions are to be returned.
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## Sample Usage

```
scope.caseRef = "BDS-1-com.example.claimbom.Policy-3-0"; //accept a case reference
from user for which we need the case actions

//Actual Service call
CaseInformationService.listCaseActions(scope,scope.caseRef, {onSuccess: function()
 { console.log("case actions for the given case reference"+scope.caseRef+" are
"+scope.caseActions); //case actions are returned by the service and assigned to
caseActions
 }
});
```

## CaseManagementService

The CaseManagementService provides operations used to manage and obtain information about cases.

## CaseManagementModel

CaseManagementModel is the object model used by CaseManagementService operations to pass data in service requests and to receive data responses.

## Attributes

Name	Type	Description
<i>caseActions</i>	Array< <a href="#">CaseCategory</a> >	Case actions. A case action defines an action a user can perform on a case.
<i>caseData</i>	Array< <a href="#">CaseDataType</a> >	Case data. Case data is business data that is centrally managed by ActiveMatrix BPM and can therefore be accessed and updated by multiple BPM process applications.
<i>caseDataTree</i>	Array< <a href="#">CaseDataTree</a> >	The case data for a case.

Name	Type	Description
caseRef	String	Case reference.
caseRefArray	Array<String>	Case references.
caseReferences Array	Array<String>	Case references.
caseReferences FromAdHocView	Array<String>	Case references.
caseRefInforma tion	Array< <a href="#">CaseReferenceWithSummaryDetailsType</a> >	Case reference and its summary details.
caseStates	Array< <a href="#">CaseState</a> >	Current and allowed state of the cases.
caseTypeAttrib utes	Array< <a href="#">CaseClassInfo</a> >	Case attributes.
caseTypes	Array< <a href="#">CaseClassInfo</a> >	Information about the case types, such as name and attribute details.
guid	String	GUID of the logged in user, used to authenticate and authorize the service request. The GUID can be obtained by using the <a href="#">BPMLoginService.getExtendedUserInfo</a> operation.
relatedCaseRef erences	Array< <a href="#">CaseObjectDetails</a> >	Details of the cases that are related to the given case reference supplied in the <a href="#">getRelatedCases</a> operation.
selectedCaseMo del	<a href="#">CaseModelBasicInfo</a>	Identifies the case model.
selectedCaseTy pe	<a href="#">CaseClassInfo</a>	Information about the case type, such as name and attribute details.
selectedWorkIt em	<a href="#">WorkItem</a>	Work item whose case information is to be retrieved.

See the following topics for specific information on how [CaseManagementService](#) operations use the [CaseManagementModel](#) object model:

- [findAllCases](#)
- [getCaseData](#)
- [getCaseInformation](#)
- [getCaseStates](#)
- [getRelatedCases](#)
- [listCaseActions](#)
- [listCaseTypes](#)

## findAllCases

Find all cases for a given case type.

### Required System Actions

readGlobalData

### Usage

```
CaseManagementService.findAllCases(model, callback)
```

### Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CaseManagement Model</a>	<p>The case model to be passed to the \$scope object in a <a href="#">selectedCaseType</a> object.</p> <p>You must provide a case model that provides a case type and the major version of the case model to which it belongs, then assign it to an object named <a href="#">selectedCaseType</a> on the scope and pass the scope to the service.</p> <p>This service requires:</p> <ul style="list-style-type: none"> <li>• <a href="#">CaseManagementModel.selectedCaseType.name</a></li> <li>• <a href="#">CaseManagementModel.selectedCaseModel.majorVersion</a></li> </ul> <p>The response populates the <a href="#">CaseManagementModel.caseReferencesArray</a> object with the references of the cases that were found.</p>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

### Sample Usage

```
$scope.selectedCaseType={}; //accept selected case type from user / assign
it from a selected casetype
$scope.selectedCaseType.name=$scope.obj.id;
$scope.selectedCaseModel={}; //accept casemodel information from user / assign it
from a selected caseModel
$scope.selectedCaseModel.majorVersion=$scope.obj.version; //accept majorVersion
from user / assign it from a selected caseModel
CaseManagementService.findAllCases($scope,{onSuccess: function()
{
 console.log("Cases of "+$scope.selectedCaseType.name+" are : "+
$scope.caseReferencesArray); //Service call assigns the list of case references to
$scope.caseReferencesArray
}
});
```

## getCaseData

Retrieves case data for the given case reference.

### Required System Actions

readGlobalData

### Usage

```
CaseManagementService.getCaseData(model, caseRef, callback)
```

### Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CaseManagement Model</a>	The case model to be passed to the \$scope object. The response populates the CaseManagementModel.caseDataTree object with the case data.
<i>caseRef</i>	String	Identifies the case whose data is to be returned.
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

### Sample Usage

```
scope.caseRef = "BDS-1-com.example.claimbom.Policy-3-0"; //accept a case reference
from user for which we need the case data

//Actual Service call
CaseManagementService.getCaseData(scope,scope.caseRef, {onSuccess: function()
{
 console.log("case data for the given case reference"+scope.caseRef+" is
"+scope.caseDataTree); //case data is returned by the service and assigned to
caseDataTree
}
});
```

## getCaseInformation

Retrieves a summary of the case information for a given work item or case reference.

### Required System Actions

readGlobalData

### Usage

```
CaseManagementService.getCaseInformation(model, callback)
```



## Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CaseManagement Model</a>	<p>The case model to be passed to the \$scope object.</p> <p>You must provide one of the following:</p> <ul style="list-style-type: none"> <li>a <a href="#">CaseManagementModel.WorkItem</a> object for the work item whose associated cases and case information are to be retrieved. The <a href="#">CaseManagementModel.WorkItem</a> must be assigned to an object named <a href="#">selectedWorkItem</a> on the scope, then the scope must be passed to the service.</li> <li>a <a href="#">CaseManagementModel.caseRef</a> for the case whose information is to be retrieved.</li> <li>a <a href="#">CaseManagementModel.caseReferencesArray</a> of the cases whose information is to be retrieved.</li> </ul> <p>The sample usage below shows how each of these can be passed.</p> <p>The response populates the <a href="#">CaseManagementModel.caseRefInformation</a> object with an array of <a href="#">CaseReferenceWithSummaryDetailsType</a> objects.</p>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## Sample Usage

```

BPMLoginService.getExtendedUserInfo({
 onSuccess : function(userInfo) {
 $scope.guid = userInfo.guid;
 $scope.selectedWorkItem = $scope.acceptWorkItemFromUser;

 //OR

 $scope.caseRef = "BDS-2 com.example.claimmodel.Claim-132-0"; //accept a case
 reference from user for which we need the case summary

 //OR

 $scope.caseReferencesArray= [];
 $scope.caseReferencesArray.push('BDS-2-com.example.claimmodel.Claim-132-0','BDS-2-
 com.example.claimmodel.Claim-132-0'); //assign multiple case references for which
 we need the case summary

 //Actual Service call
 CaseManagementService.getCaseInformation($scope, {onSuccess: function()
 {
 console.log("Case Reference Summary is
 "+JSON.stringify($scope.caseRefInformation)); //Service call assigns the case
 summary to scope.caseRefInformation
 }
 });
 }
});

```

## getCaseStates

Retrieves the case states for one or more given case references.

### Required System Actions

readGlobalData

### Usage

```
CaseManagementService.getCaseStates(model, callback)
```

### Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CaseManagementModel</a>	<p>The case model to be passed to the \$scope object.</p> <p>You must provide an array of case references (<a href="#">CaseManagementModel.caseRefArray</a>) for which the case states are to be returned.</p> <p>The response populates the <a href="#">CaseManagementModel.caseStates</a> object with the current state of the case(s).</p>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

### Sample Usage

```
scope.caseRef = "BDS-1-com.example.claimbom.Policy-3-0"; //accept a case reference
from user for which we need case states
scope.caseRefArray = [];
scope.caseRefArray.push(scope.caseRef); //user can assign one or multiple case
references to find their case states

//Actual Service call
CaseManagementService.getCaseStates(scope, {onSuccess: function()
{
 console.log("Case States for the given case references"+scope.caseRefArray
+" are "+scope.caseStates); //Case states are returned and assigned to caseStates
}
});
```

## getRelatedCases

Retrieves all of the cases that are linked or related to the given case reference.

### Required System Actions

readGlobalData

### Usage

```
CaseManagementService.getRelatedCases(model, caseRef, callback)
```

## Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CaseManagement Model</a>	The case model to be passed to the \$scope object. The response populates the <a href="#">CaseManagementModel.relatedCaseReferences</a> object with the case data.
<i>caseRef</i>	String	Identifies the case for which related cases are to be returned.
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## Sample Usage

```
scope.caseRef = "BDS-1-com.example.claimbom.Policy-3-0"; //accept a case reference
from user for which we need the related cases

//Actual Service call
CaseManagementService.getRelatedCases(scope,scope.caseRef, {onSuccess: function()
 { console.log("case data for the given case reference"+scope.caseRef+" is
"+scope.relatedCaseReferences); //related cases are returned and assigned to
relatedCaseReferences
 }
});
```

## listCaseActions

Lists all the case actions available for a given case reference.

## Required System Actions

readGlobalData

## Usage

```
CaseManagementService.listCaseActions(model, caseRef, callback)
```

## Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CaseManagement Model</a>	The case model to be passed to the \$scope object. The response populates the <a href="#">CaseManagementModel.caseActions</a> object with the case actions.
<i>caseRef</i>	String	Identifies the case for which case actions are to be returned.
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## Sample Usage

```
scope.caseRef = "BDS-1-com.example.claimbom.Policy-3-0"; //accept a case reference
from user for which we need the case actions

//Actual Service call
CaseManagementService.listCaseActions(scope,scope.caseRef, {onSuccess: function()
 { console.log("case actions for the given case reference"+scope.caseRef+" are
"+scope.caseActions); //case actions are returned by the service and assigned to
caseActions
 }
});
```

## listCaseTypes

Retrieve the list of all case types for the provided case model.

### Required System Actions

readGlobalData

### Usage

```
CaseManagementService.listCaseTypes(model, callback)
```

### Parameters

Parameter	Type	Description
<i>model</i>	<a href="#">CaseManagement Model</a>	<p>The caseModel to be passed to the \$scope object in a <a href="#">selectedCaseModel</a> object.</p> <p>You must provide a caseModel and assign it to an object named <a href="#">selectedCaseModel</a> on the scope and pass the scope to the service.</p> <p>This service requires:</p> <ul style="list-style-type: none"> <li>• CaseManagementModel.selectedCaseModel.caseModel ID</li> <li>• CaseManagementModel.selectedCaseModel.appName</li> <li>• CaseManagementModel.selectedCaseModel.majorVersion</li> </ul> <p>The response returns an array <a href="#">CaseManagementModel.CaseClassInfo</a> objects. This array is assigned to an object named <a href="#">caseTypes</a> on the scope.</p>
<i>callback</i>	Function	Callback function to handle success or failure results from the request.

## Sample Usage

```
$scope.selectedCaseModel={};
$scope.selectedCaseModel.caseModelID=$scope.obj.id; //accept id from user /
assign it from a selected caseModel
$scope.selectedCaseModel.appName=$scope.obj.appName; //accept appName from user /
assign it from a selected caseModel
$scope.selectedCaseModel.majorVersion=$scope.obj.version; //accept majorVersion
from user / assign it from a selected caseModel
```

```
//Actual Service call
CaseManagementService.listCaseTypes($scope,{onSuccess: function() {
 console.log("Case Types of case Model "+$scope.selectedCaseModel.appName+" are :
"+$scope.caseTypes); //Service call assigns caseTypes to a $scope.caseTypes object
}
});
```

# objectAPI

The objectAPI provides services that perform BPM-related functionality. The API is provided in the form of a JavaScript library. It supports both JavaScript and AngularJS.

Each objectAPI service performs a specific type of BPM functionality. For instance, the `AuditService` provides functions for managing audit information, the `WorkListService` is used to display and manage work lists, and so on. For a list of all available objectAPI services, see [objectAPI Reference](#).

Internally, the objectAPI services invoke the ActiveMatrix BPM REST API. Some services return JavaScript objects corresponding to the returned JSON value.

## Invoking the objectAPI

To invoke the objectAPI, you must include the following libraries:

```
<script type="text/javascript" src="/apps/app-cdn/tcf/tcf.nocache.js"></script>
<script src="/apps/app-cdn/angular/1.6.0/angular.min.js"></script>
```

The `tcf.nocache.js` library contains the underlying logic used by the objectAPI. The Angular library is needed even if you are using pure JavaScript.

If you are using JQuery in your application, you must also include the following library:

```
<script type="text/javascript" src="/apps/app-cdn/jquery/jquery-3.1.1.js"></script>
```

The single point of entry for all objectAPI services is the [RestApiService](#) object.

In AngularJS, it can be injected as a service named `RestApiService`.

A function in the service is executed as follows:

```
RestApiService.getWorkListService().getWorkListItemsForGlobalData(...)
```

All request objects are in the package:

```
tibco.objectapi.service.request
```

All functions are asynchronous and have a callback object:

```
var request= ...
var callback=...
RestApiService.getWorkListService().getWorkListItemsForGlobalData(request,new
callback())
```

## Example of using the objectAPI -- getting a work list

1. Get access to the service for work lists:

```
RestApiService.getWorkListService ()
```

2. Create a request object:

```
var request = new tibco.objectapi.service.request.GetWorkListItemsRequest
("RESOURCE", " tibco-admin", 0, 100);
```

3. Create a response callback:

```
var callback = function(deferred) {
 this.onSuccess = function(result) {
 // success you got worklistResponse
 var workitems=result.workItems;
 },
 this.onFailure = function(message) {
 //handle failure
 };
};
```

4. Make the service call:

```
RestApiService.getWorkListService().getWorkListItems(request, new callback());
```

## objectAPI Configuration

The objectAPI can be configured using the `restServiceConfig` object.

The objectAPI configuration is set by passing the `restServiceConfig` object to the `RestApiService.setRestServiceConfig` function.

The `restServiceConfig` object is used to configure the following:

- **baseUrl** (String) - The REST API endpoint for all ActiveMatrix BPM services, with the exception of the L10NService and the RoleService (see the **appsBaseUrl** configuration below).
- **appsBaseUrl** (String) - The REST API endpoint for WBPM services, which include the L10NService and the RoleService.
- **headers** (JavaScript object) - These specify operating parameters of the service. There is a property name and value for each header. Some common headers are:
  - Authorization - The credentials for the call. For example:  
`'Authorization': "Basic dGliY28tYWRTaW46c2VjcmV0"`
  - Content-Type - The media type for the request body (for calls that use POST and PUT in the underlying REST API). For example:  
`'Content-Type': "application/json"`
  - Accept - The media type for the response from a call. For example:  
`'Accept': "application/json"`
- **corsEnabled** (boolean) - This is currently not used.

The `restServiceConfig` object values can be set in one of two ways:

As a function that returns the value:

```
{
 getBaseUrl : getBaseUrlFunction,
 getAppsBaseUrl : getAppsBaseUrlFunction,
 getHeaders : getHeadersFunction,
 isCorsEnabled : isCorsEnabledFunction
}
```

Or, as a property that is set to the value:

```
{
 baseUrl : baseUrlString,
 appsBaseUrl : appsBaseUrlString,
 headers : headersObject,
 corsEnabled : corsEnabledBoolean
}
```

If both the function and value are set, the value returned from the function is used.

### Examples

The following are examples of setting configuration values:

```
var getBaseUrl = function () {
 var baseUrl = 'http://<hostname>:<port>/bpm/rest';
 return baseUrl;
};

var getHeaders = function() {
 return {'Authorization': "Basic dGliY28tYWRTaW46c2VjcmV0"};
};

var isCorsEnabled = function() {
 return false;
};
```

## objectAPI Reference

The objectAPI services are divided into categories that describe the functions in the service.

The available objectAPI services are:

### **AuditService**

Functions to to manage purges of audit entries, submit events for audit by Event Collector, and get audit information from the Event Collector database tables.

### **BusinessDeadlineService**

A function to calculate the deadline for completing a specified task, based on a given calendar.

### **BusinessServiceService**

Functions to get information about and interact with deployed business services.

### **CaseActionService**

A function that lists the case actions for the specified case reference.

### **DirectoryService**

Functions to manage LDAP containers and to get information about LDAP connections.

### **DocumentService**

Use this service to perform functions with case folders, which are used to store documents associated with a case.

### **DynamicResourceQueryService**

Functions to reset dynamic resource queries.

### **EntityResolverService**

A function to locate a resource by LDAP DN.

### **ExporterService**

Functions to export and import defined LDAP containers, resource mappings, and push destinations.

### **GlobaldataAdminService**

Functions to get information about and administer case models.

### **GlobaldataService**

Functions to create, read, update and delete case data. You can also perform searches on case data based on specific criteria.

### **OrgEntityConfigService**

Functions to manage resource configuration attributes and to get or set them for individual resources.

### **OrgModelService**

Functions to get information about organization models, as well as get or set information for calendars, push destinations, extension points, and candidate queries that are tied to a particular organization model.

### **OrgResourceService**

Functions to manage resources, such as creating, deleting, finding, and updating resources.

### **PageFlowService**

Functions to get information about and interact with deployed PageFlows.

### **ProcessManagementService**

Functions to manage process templates and process instances.

### **ResourceQueryService**



Functions to execute a Resource Query Language (RQL) query to find a set of resources that match specific criteria.

### [RoleService](#)

Functions related to roles, which are used to control access to applications, as well as access to specific functionality within an application.

### [SecurityService](#)

Functions to manage user authorization.

### [StatisticsService](#)

Function to request a process template measure from the Event Collector database tables.

### [UserSettingsService](#)

Functions to manage user settings.

### [WorkCalService](#)

Functions to create, delete, edit, and get information about base and overlay calendars and the time zones and calendar entries that they use.

### [WorkItemManagementService](#)

Functions to perform actions on work items.

### [WorkListService](#)

Functions to get work item lists for organization model entities, and to get or to set sort/filter criteria for work item lists.

### [WorkPresentationService](#)

Functions to perform various operations on work items.

## RestApiService

The single point of entry for all objectAPI services is the [RestApiService](#) object. For information about using RestApiService to invoke objectAPI methods, see [objectAPI](#).


## AuditService



The AuditService contains functions to manage purges of audit entries, submit events for audit by Event Collector, and get audit information from the Event Collector database tables.

The table below lists the functions available from the [AuditService](#).


For more information about audit events, see [Events](#).


Function	Description	Returns
<a href="#">addCaseComment</a> --or-- <a href="#">addCaseCommentAlt</a>	Stores a comment against a case. You can use these comments to create a custom audit trail.  For more information, see <a href="#">Comments</a> .  <b>Required system action:</b> queryAudit	<a href="#">Comment</a>
<a href="#">addProcessInstanceComment</a>	Stores a comment against a process instance. You can use these comments to create a custom audit trail.  For more information, see <a href="#">Comments</a> .  <b>Required system action:</b> queryAudit	<a href="#">Comment</a>

Function	Description	Returns
<a href="#">addWorkItemComment</a> --or-- <a href="#">addWorkItemCommentAlt</a>	Stores a comment against a work item. You can use these comments to create a custom audit trail.  For more information, see <a href="#">Comments</a> .  <b>Required system action:</b> queryAudit	<a href="#">Comment</a>
<a href="#">checkAutoArchive</a>	Checks whether an archive is currently running. It returns the status of the last archive that ran. If the archive is currently running, it also returns the status of the archive that is currently running.  <b>Required system action:</b> none	<a href="#">CheckAutoArchiveResponseObject</a>
<a href="#">checkPurge</a>	Checks the status of a purge started with the doPurge function (see below).  <b>Required system action:</b> none	<a href="#">PurgeStatus</a>
<a href="#">doPurge</a>	Purges data related to completed, cancelled, and failed process instances from the Event Collector database tables.  You can specify the process templates to be purged either by ID or by name and version, as well as other options that limit the extent of the purge.  <div>  TIBCO recommends that you back up your database before running doPurge. For more information, see <a href="#">Purging Audit Data</a>. </div> <b>Required system action:</b> none	<a href="#">PurgeStatus</a>

Function	Description	Returns
<a href="#">event</a>	<p>Submits an event to the Event Collector database.</p>  <ul style="list-style-type: none"> <li>You submit a particular instance of an event to Event Collector if you want that event to be audited. See the "Audit" chapter in <i>TIBCO ActiveMatrix BPM Administration</i> for details of how to set up auditing.</li> <li>You can submit details of any of the event's attributes, provided that they have previously been registered using the <code>registerAttributes</code> function (see below).</li> <li>There is no response to this operation. If it is necessary to confirm whether the <code>eventRequest</code> has been correctly received, you can query the database. See "Working with Events and the <i>TIBCO ActiveMatrix BPM Event Collector Schema Reference</i> for information about querying the database.</li> </ul> <p><b>Required system action:</b> none</p>	Empty response (no JSON payload)
<a href="#">executeGenericQuery</a>	<p>Executes the specified query against the Event Collector database tables.</p> <p>If a query needs to be executed multiple times, better performance will be experienced by registering the query with <code>registerQuery</code>, then using the <code>executeRegisteredQuery</code> function.</p> <p><b>Required system action:</b> none</p>	<a href="#">ExecuteQueryResponseType</a>
<a href="#">executeRegisteredQuery</a>	<p>Executes a query that has been previously registered using the <code>registerQuery</code> function.</p>  <ul style="list-style-type: none"> <li>When queries are registered they are represented both by a GUID returned by the registration, and also by a tag which is passed in when registering.</li> <li>The request must specify the previously registered query that is to be executed, along with any options to be used on this particular execution of the query.</li> <li>The response returns details of the events that match the specified query, along with summary information about the executed query.</li> </ul> <p><b>Required system action:</b> none</p>	<a href="#">GenericResult</a>

Function	Description	Returns
<a href="#">getAttributes</a>	Lists all attributes in the Event Collector database tables that are registered for a specific component.  <b>Required system action:</b> none	<a href="#">AttributeDefinitions</a>
<a href="#">getCaseComments</a>	Retrieves comments that have been stored against a case object, by specifying a managedObjectId.  Also see <a href="#">getSearchCaseComments</a> , in which you can pass a case reference instead of a managedObjectId.  For more information, see <a href="#">Comments</a> .  <b>Required system action:</b> queryAudit	<a href="#">CommentAudit</a>
<a href="#">getCaseReport</a>	Generates an audit trail for the given case class.  <b>Required system action:</b> none	<a href="#">CaseAudit</a>
<a href="#">getChart</a>	Aggregates the data in an Event Collector <i>area</i> (work items, cases, process templates, process instances, or events) to obtain audit information about the area.  This operation aggregates, and returns, the audit information in a tabular format, which is consumable by charting libraries such as Highcharts.  Also see the <a href="#">BPMChartService</a> component service.  <b>Required system action:</b> queryAudit	<a href="#">ChartResult</a>
<a href="#">getChartData</a>	Returns audit information for a specified Event Collector <i>area</i> (work items, cases, process templates, process instances, or events) based on specified filter criteria. For example, you can request all outstanding work items whose status is OFFERED.  This operation returns the audit information in a tabular format, which is consumable by charting libraries such as Highcharts.  Also see the <a href="#">BPMChartService</a> component service.  <b>Required system action:</b> queryAudit	<a href="#">ChartDataResult</a>
<a href="#">getComponents</a>	Lists the components that are registered on the system.  <b>Required system action:</b> none	<a href="#">getComponentsResponseObject</a>
<a href="#">getProcessInstanceReport</a>	Generates an audit trail for a given process instance. The request contains the ID of the process instance. Optionally, you can specify how the information is displayed.  <b>Required system action:</b> none	<a href="#">ProcessInstanceAudit</a>

Function	Description	Returns
<a href="#">getProcInstComments</a>	Retrieves comments stored against a specified process instance.  For more information, see <a href="#">Comments</a> . <b>Required system action:</b> queryAudit	<a href="#">CommentAudit</a>
<a href="#">getSearchCaseComments</a>	Retrieves comments that have been stored against a case, by specifying a case reference.  Also see <a href="#">getCaseComments</a> , in which you can pass a managedObjectId instead of a case reference.  For more information, see <a href="#">Comments</a> . <b>Required system action:</b> queryAudit	<a href="#">CommentAudit</a>
<a href="#">getSearchCaseReport</a>	Generates an audit trail for the given case data. <b>Required system action:</b> none	<a href="#">CaseAudit</a>
<a href="#">getWorkItemComments</a>	Retrieves comments stored against a specified work item.  For more information, see <a href="#">Comments</a> . <b>Required system action:</b> queryAudit	<a href="#">CommentAudit</a>
<a href="#">getWorkItemReport</a>	Generates an audit trail for a given work item.  The request contains the ID of the work item. Optionally, you can specify how the information is displayed. <b>Required system action:</b> none	<a href="#">WorkItemAudit</a>
<a href="#">isQueryRegistered</a>	Tests whether a query is registered with Event Collector. (This can be useful for determining whether a frequently used query tag is already present on the system.) <b>Required system action:</b> none	<a href="#">QueryRegisteredResponseObject</a>
<a href="#">lookupQueryByTag</a>	Lists the details for a particular query (identified by its tag). This function can be used to obtain the GUID for a registered query.   The <a href="#">executeRegisteredQuery</a> function (see above) executes faster if the query is identified using its GUID rather than its tag. <b>Required system action:</b> none	<a href="#">lookupQueryByTagResponseObject</a>

Function	Description	Returns
<a href="#">registerAttributes</a>	<p>Registers attributes with the Event Collector database.</p> <p>Before you can audit an event, you must have registered with Event Collector any attributes used. You can register more than one attribute with the registerAttributes function. When registering an attribute, you must specify the component it is being registered against.</p> <div>  <p><b>Do not</b> register any attributes against existing TIBCO components. Doing so will cause TIBCO ActiveMatrix BPM to fail the next time you attempt to start it. You should only register attributes against custom components.</p> <p>The attributes delivered as part of TIBCO ActiveMatrix BPM are already registered.</p> </div> <p><b>Required system action:</b> none</p>	<a href="#">RegisterAttributesResponseObject</a>
<a href="#">registerComponent</a>	<p>Registers a component with the Event Collector database.</p> <p>You need to register a component if you have added a custom component to TIBCO ActiveMatrix BPM, and you intend to audit messages produced by that component. The TIBCO components that make up TIBCO ActiveMatrix BPM (such as BRM or Event Collector) are already registered.</p> <p>Use the getComponents function (see above) to see what components are registered.</p> <p>Attributes are registered against a registered component, so you must register a component first before you can use the registerAttributes function (see above) to register attributes against that component, or use the event function (see above) to audit messages that relate to that component.</p> <p><b>Required system action:</b> none</p>	<a href="#">registerComponentResponseObject</a>
<a href="#">registerQuery</a>	<p>Registers a query with Event Collector, so that it can be subsequently executed using the executeRegisteredQuery function (see above).</p> <p><b>Required system action:</b> none</p>	<a href="#">registerQueryResponseObject</a>
<a href="#">stopAutoArchive</a>	<p>Stops an archive operation that is currently running. The archive operation may not be able to stop immediately, but will stop as soon as possible.</p> <p><b>Required system action:</b> none</p>	<a href="#">StopAutoArchiveResponseObject</a>

Function	Description	Returns
<a href="#">stopPurge</a>	<p>Stops a purge started with the <a href="#">doPurge</a> function (see above).</p> <p>The request can either call for a simple status message, or can specify options that specify the data to be returned as fields in the <a href="#">purgeStatus</a> object.</p> <p><b>Required system action:</b> none</p>	<a href="#">PurgeStatus</a>
<a href="#">triggerAutoArchive</a>	<p>Starts an archive, if it is not currently running. If the archive overlaps the time that the next archive is scheduled to run, the archive will not run and the next archive will take place at the next scheduled time.</p> <p>If you have disabled automatic archiving by specifying <code>autoArchiveEnabled=false</code> in the <code>ec.properties</code> file, then try and trigger an archive using <code>triggerAutoArchive</code>, the archive operation will not be triggered. No warning or error is returned to inform you that the archive operation has not been triggered.</p> <p><b>Required system action:</b> none</p>	<a href="#">AutoArchiveOptions</a>
<a href="#">unregisterQuery</a>	<p>Unregisters a previously registered query.</p> <p><b>Required system action:</b> none</p>	<a href="#">UnRegisterQueryResponseObject</a>

## Purge Audit Status

The purge functions in the `AuditService` return a status.

The response from the [doPurge](#), [checkPurge](#), and [stopPurge](#) functions may be one of the following:

- A status message. This is the response if extended data is not requested.
- The status message plus a series of fields in the [purgeStatus](#) object. This response is given if you specify either of:
  - `requireAllExtendedData=true`
  - `requireExtendedData=<field;field...>`

## Merged Status Message

The format of the status message included in the response to [doPurge](#), [checkPurge](#), and [stopPurge](#) depends on the status returned. The table below shows the message formats.

PURGE_IN_PROGRESS	Purge Operation [ <i>purge_name</i> ] started at <i>start_time</i> is still in progress! <i>deleted</i> instances of total <i>total</i> deleted!
PURGE_COMPLETED	Purge Operation [ <i>purge_name</i> ] started at <i>start_time</i> finished at <i>end_time</i> with success! <i>deleted</i> instances of total <i>total</i> deleted!

PURGE_FAILED	Purge Operation [ <i>purge_name</i> ] started at <i>start_time</i> finished at <i>end_time</i> with error! <i>deleted</i> instances of total <i>total</i> deleted!  . ace <i>failure_stack_trace</i>
PURGE_STOPPED	Purge operation [ <i>purge_name</i> ] stopped successfully!

Where:

purge_name	=	A unique automatically-generated identifier for the purge. For example, [EC-PURGE-1351517760674].
start_time	=	The start time of the purge.
end_time	=	The time when the purge completed.
deleted	=	The number of instances deleted by the purge operation so far.
total	=	The total number of instances identified as matching the purge criteria, and therefore eligible to be deleted.
failure_stack_trace	=	The stack trace for the failure, if the purge resulted in a failure.

### PurgeStatus Object

You can configure any of the [doPurge](#), [checkPurge](#), and [stopPurge](#) functions to use **requiredExtendedData**. This causes status data to be returned via fields in the [purgeStatus](#) object. Possible values are:

- **requireAllExtendedData**: If `true`, return all the available extended data for this operation. Not all fields will be available to be returned for all circumstances. For example, if a purge operation is currently in progress the **END** field is not returned.
- **requiredExtendedData**: This enables you to specify one or more specific items of extended data. Possible values are:

Field	Meaning
START	The start date of a purge operation.
END	The end date of purge operation.
DETAILS	Any additional details about the operation. (This is currently only used for stack traces when errors occur).
NAME	The automatically-generated unique name for the purge operation.
CURRENT	The number of items purged so far.
TOTAL	The total number of items found that need to be purged.



## BusinessDeadlineService

The BusinessDeadlineService is used to calculate the deadline for completing a specified task, based on a given calendar.

The table below lists the functions available from the [BusinessDeadlineService](#).

Function	Description	Returns
<a href="#">calcDeadline</a>	<p>Calculates the earliest date and time that a deadline can be completed. A starting time (start-date-time) and a duration for the task must be given.</p> <p>If the duration does not specify hours, minutes, or seconds, then it is assumed to be in <b>working days</b>. See <a href="#">Duration Definitions</a>. Days with too little working time do not count as <b>working days</b>; see <a href="#">Minimum Hours in Working Day Calculations</a>.</p> <p>The <code>calendarLookAhead</code> property in the <code>dac.properties</code> file determines how far ahead in the calendar the calculation algorithm looks to take account of entries that would affect the duration of a task. See "Configuring TIBCO ActiveMatrix BPM Deadline and Calendar Properties" in <i>TIBCO ActiveMatrix BPM Administration</i> for details of this property.</p> <p>The request may optionally also include a collection of identifiers from which a calendar can be resolved. There can be any number of these, each line being one of:</p> <ul style="list-style-type: none"> <li>• A calendar reference.</li> <li>• Information identifying an entity within an organization model which is the participant for the task. An organizational entity may be associated at design time with a particular calendar.</li> </ul> <p>If these identifiers do not define a calendar to be used, then the default Base calendar is used to calculate the deadline.</p> <p>Once the operation has determined which base and overlay calendars are to be used for the deadline calculation, it determines the available working hours in those calendars. Using that information, the operation calculates and returns the earliest time at which work of the given duration can be completed.</p> <p><b>Required system action:</b> readCalendars</p>	<a href="#">XmlCalcDeadline Response</a>

## Duration Definitions

Depending on how a **duration** parameter is expressed, it may be interpreted either as elapsed time, or as a number of **working days**.

- If the **duration** parameter is specified only in years, months, weeks, and/or days, but does not specify hours, minutes, or seconds, then it is assumed to be in **working days**. For example:
  - the duration P3D is taken to be three working days, not 72 hours.

- P1M is interpreted as 31 working days (regardless of the number of days in the actual month in which the period occurs).
- Similarly, P1Y is 365 working days.
- When a deadline calculation is made in terms of working days, the calculation takes account of the **start-date-time** to check whether the current day has enough working hours remaining to count as a working day (see [Minimum Hours in Working Day Calculations](#)). If it does not, the next available business day is used as the first working day in the calculation. The final result will then be the *start* of the first available day following the requested number of working days.

For example, if the working day is defined as being from 08:00 to 17:00, and minimum working hours value is defined as 5, a calculation with a **start-date-time** before 12:00 noon on the current day will treat the current day as a working day. If the duration is P1D, the calculation would return a deadline of the next business day at 08:00.

However if the **start-date-time** is after 12:00 noon on the current day, there are less than five hours remaining in the business day, and so there are no longer enough hours to count the current day as a working day. The calculation then treats the next business day as the first available working day, and the start of the working day after that would be returned as the deadline.

- If the **duration** parameter is specified only in hours, minutes, or seconds, then it is assumed to indicate elapsed time. So if the duration were PT72H then it would be taken to be 72 elapsed hours, not three working days.
- If days (or months, or years) is specified *in combination* with any of hours, minutes, or seconds, then the duration is taken to be elapsed time and a day is taken as 24 hours. For example, P3D1H would be 73 hours. P1D represents one working day; P1D1H is 25 hours.

### Minimum Hours in Working Day Calculations

Each base calendar holds a value that defines the minimum number of hours in a working day. This may differ from the normal business hours defined by the **day-of-week** and **time-slot** entries in the same base calendar.

When a deadline is being calculated in **working days**, if the number of available hours in a day being considered is less than the defined minimum hours, that day is not included as a working day, and the calculation moves on to the next available day; where, again, the number of available hours is compared. The final result will be the start of the available day following the requested number of working days.

For example, if the normal working day is defined as eight hours for Mondays to Fridays, and **min-hours** (for details about **min-hours**, see [saveBaseCalendar](#)) is defined as four hours:

- Saturdays and Sundays are ignored for deadline calculation because they have fewer working hours (0) than the **min-hours** value. The deadline calculation moves on to the Monday, where the number of available hours is compared with the minimum.
- A day with a six-hour exclusion defined for a training session will be ignored, as this brings the available working hours down below the **min-hours** value. The deadline calculation moves on to the next available day, where again the number of available hours is compared with the minimum.

### BusinessServiceService

The BusinessServiceService contains functions to get information about and interact with deployed business services.

The table below lists the functions available from the [BusinessServiceService](#).

Function	Description	Returns
<a href="#">cancelBusinessService</a>	<p>Cancels a given business service instance.</p> <p><b>Required system action:</b> executeBusinessService</p>	<a href="#">CancelBusinessServiceResponse</a>
<a href="#">getBusinessServiceDetailsByModule</a>	<p>Retrieves all the information about a business service, identified by module.</p> <p><b>Required system action:</b> listBusinessServices</p>	<a href="#">BusinessServiceType</a>
<a href="#">injectBusinessService</a>	<p>Injects data into a business service whose execution state is IN_PROGRESS.</p> <p>The request identifies an intermediate event that is used to inject data into the given business service and specifies the formal parameter values that are to be injected.</p> <p>The response returns complete information about the injected business service instance.</p> <p><b>Required system action:</b> executeBusinessService</p>	<a href="#">injectBusinessServiceEventResponseType</a>
<a href="#">listBusinessServices</a>	<p>Lists deployed business services.</p> <p>The request specifies any filter criteria to be applied to the request.</p> <p><b>Required system action:</b> listBusinessServices</p>	<a href="#">ListBusinessServicesResponse</a>
<a href="#">listCategories</a>	<p>Lists categories of all deployed business services.</p> <p>The request requests a list of the categories of all deployed business services, for either a specific channel or for all channels.</p> <p><b>Required system action:</b> listBusinessServices</p>	<a href="#">ListCategoriesResponse</a>
<a href="#">queryBusinessServices</a>	<p>Returns the deployed business services that satisfy the specified filter criteria.</p> <p><b>Required system action:</b> listBusinessServices</p>	<a href="#">QueryBusinessServicesResponse</a>
<a href="#">queryCategories</a>	<p>Returns categories of the deployed business services that satisfy the specified filter criteria.</p> <p><b>Required system action:</b> listBusinessServices</p>	<a href="#">QueryCategoriesResponse</a>
<a href="#">startBusinessService</a>	<p>Starts an instance of a deployed business service.</p> <p><b>Required system action:</b> executeBusinessService</p>	<a href="#">startBusinessServiceResponse</a>
<a href="#">updateBusinessService</a>	<p>Updates the specified business service instance.</p> <p><b>Required system action:</b> executeBusinessService</p>	<a href="#">updateBusinessServiceResponse</a>

## CaseActionService

The CaseActionService contains a function that lists the case actions for the specified case reference.

The table below lists the functions available from the [CaseActionService](#).

Function	Description	Returns
<a href="#">listCaseAction</a>	Lists the case actions for the specified case reference. <b>Required system action:</b> listBusinessServices	<a href="#">listCaseActionResponse</a>


## DirectoryService

The DirectoryService contains functions to manage LDAP containers and to get information about LDAP connections.

The table below lists the functions available from the [DirectoryService](#).

Function	Description	Returns
<a href="#">deleteContainer</a>	Deletes the configuration of the identified LDAP container. If the LDAP container has any resources associated with it, you must use the <b>delete-resources</b> parameter to also delete the associated resources, otherwise the deletion will fail. <b>Required system action:</b> deleteLDAPAdmin If you are deleting resources with this function (using the <b>deleteResources</b> parameter), you must also have either the resourceAdmin or deleteResourceAdmin system action.	<a href="#">DeleteContainerResponse</a>
<a href="#">executeLargeLdapQuery</a>	The same as the executeLdapQuery function (see below), except this function is intended for complex or large LDAP queries. <b>Required system action:</b> LDAPAdmin	<a href="#">ExecuteLdapQueryResponse</a>
<a href="#">executeLdapQuery</a>	Returns the Distinguished Names (DNs) for LDAP entries that match the specified query. The query traverses all subtree nodes starting from the node identified by the <b>basedn</b> parameter. <b>Required system action:</b> LDAPAdmin	<a href="#">ExecuteLdapQueryResponse</a>
<a href="#">executePagedLdapQuery</a>	Returns the Distinguished Names (DNs) for LDAP entries that match the specified query, in a paged format. The response returns the following two values: <ul style="list-style-type: none"> <li><b>bookmark:</b> This value is passed in subsequent calls to get the next page of results. If the bookmark returns empty in the response, it means the end of the result set has been reached.</li> <li><b>estimated-size:</b> The estimated number of DN's that satisfy the query. Also note that this is not supported by all LDAP servers (the Apache LDAP server does not support it); when not supported, it returns 0.</li> </ul> <b>Required system action:</b> LDAPAdmin	<a href="#">ExecuteLdapQueryResponse</a>

Function	Description	Returns
<a href="#">getCandidateDetail</a>	<p>Retrieves additional information about one candidate returned by the <a href="#">listCandidateResources</a> function (see below).</p> <p>If the candidate is one for which a resource already exists, the GUID and the name of that existing resource are returned in the response. If not, the name that will be assigned (by default) to the resource created from this candidate entry is returned.</p> <p><b>Required system action:</b> LDAPAdmin</p>	<a href="#">GetCandidateDetailResponseType</a>
<a href="#">getLdapEntry</a>	<p>Gets details of a specified LDAP entry.</p> <p>The request identifies an LDAP entry by its alias and DN, and specifies the attributes required. If no attribute names are listed, it is taken as a request for all attributes.</p> <p><b>Required system action:</b> LDAPAdmin</p>	<a href="#">GetLdapEntryResponseType</a>
<a href="#">listAttributeNames</a>	<p>Lists the names of those attributes that are available from the LDAP entries associated with the named LDAP connection.</p> <p>The request identifies the LDAP connection by its alias. The search can be further limited by specifying a base-dn, an LDAP query as a filter, and a sample size.</p> <p><b>Required system action:</b> LDAPAdmin</p>	<a href="#">ListAttributeNamesResponseType</a>
<a href="#">listAttributeNamesLargeQuery</a>	<p>Same as the <a href="#">listAttributeNames</a> function (see above), except this function is intended for complex or large LDAP queries, where putting the query in the URL would be impractical.</p> <p><b>Required system action:</b> LDAPAdmin</p>	<a href="#">ListAttributeNamesResponseType</a>
<a href="#">listCandidateResources</a>	<p>Lists all candidate resources (including existing resources) from the identified LDAP container. The entries in the response are suitable to be passed to the <a href="#">createResource</a> function in <a href="#">OrgResourceService</a>.</p> <p>Also see the <a href="#">listPagedCandidatesResources</a> function (below) to list a page of candidate resources.</p> <p><b>Required system action:</b> LDAPAdmin or resourceAdmin <sup>1</sup></p>	<a href="#">ListCandidateResourcesResponseType</a>
<a href="#">listContainers</a>	<p>Lists the configuration detail of all LDAP containers to which the user has access permission. Access can be restricted by the organizations to which the calling user is associated.</p> <p><b>Required system action:</b> LDAPAdmin or resourceAdmin <sup>1</sup></p>	<a href="#">ListContainersResponseType</a>

Function	Description	Returns
<a href="#">listLdapConnections</a>	<p>Lists details of all the LDAP connection shared resources available to Directory Engine.</p> <p>The response lists those LDAP connections named with the prefix <code>ldap/de/</code>, indicating that they are available to Directory Engine. Each connection is identified by its alias (which is the name minus the <code>ldap/de/</code> prefix). The response also includes the URL that the connection uses.</p> <div>  <p>You must use TIBCO ActiveMatrix Administrator to create LDAP connections. Note that the <b>Name</b> used within Administrator corresponds to the alias used here; that is, it does not have the <code>ldap/de/</code> prefix.</p> </div> <p><b>Required system action:</b> LDAPAdmin</p>	<a href="#">ListLdapConnectionsResponseType</a>
<a href="#">listPagedCandidateResources</a>	<p>Returns a page of candidate resources (including existing resources) from the identified LDAP container. The entries in the response are suitable to be passed to the <a href="#">createResource</a> function in <code>OrgResourceService</code>.</p> <p>The response returns the following two values:</p> <ul style="list-style-type: none"> <li>• <b>bookmark:</b> This value is passed in subsequent calls to get the next page of results. If the bookmark returns empty in the response, it means the end of the result set has been reached.</li> <li>• <b>estimated-size:</b> The estimated number of candidate resources. Also note that this is not supported by all LDAP servers (the Apache LDAP server does not support it); when not supported, it returns 0.</li> </ul> <p><b>Required system action:</b> LDAPAdmin or resourceAdmin <sup>1</sup></p>	<a href="#">ListCandidateResourcesResponseType</a>
<a href="#">saveContainer</a>	<p>Saves the given LDAP container configuration. Use this function to create a new LDAP container or to update an existing one.</p> <p>The request parameters depend on which of the following type of LDAP source is used to identify candidate resources:</p> <ul style="list-style-type: none"> <li>• <b>LDAP Query Source</b> - An LDAP query is used to identify the directory entries that will be candidate resources.</li> <li>• <b>LDAP Group Source</b> - A Group DN is used to identify the directory entry that is the group. When a Group DN is specified, a member attribute is also specified, which holds the collection of member identifiers, that is, their DN's. This provides the list of candidate resources.</li> </ul> <p>The response returns the unique ID of the newly-created LDAP container.</p> <p><b>Required system action:</b> LDAPAdmin</p>	<a href="#">SaveContainerResponseType</a>

Function	Description	Returns
<a href="#">updateContainer</a>	<p>Saves the given LDAP container configuration. Use this function to update an existing container.</p> <p>The request parameters depend on which of the following type of LDAP source is used to identify candidate resources:</p> <ul style="list-style-type: none"> <li>• <b>LDAP Query Source</b> - An LDAP query is used to identify the directory entries that will be candidate resources.</li> <li>• <b>LDAP Group Source</b> - A Group DN is used to identify the directory entry that is the group. When a Group DN is specified, a member attribute is also specified, which holds the collection of member identifiers, that is, their DNs. This provides the list of candidate resources.</li> </ul> <p>The response returns the unique ID of the updated LDAP container.</p> <p><b>Required system action:</b> LDAPAdmin</p>	<a href="#">SaveContainerResponseType</a>


<sup>1</sup> Users possessing either the LDAPAdmin or the resourceAdmin system action can call this function. But having the LDAPAdmin system action gives the user additional access to organizations that are restricted due to organization relationships. For more information, see [Overriding Organization Relationships](#).

## DocumentService

The DocumentService is used to manage case folders, which are used to store documents associated with a case.

The table below lists the functions available from the [DocumentService](#).


System actions required: deleteDocument, deleteOrphanedFolders, and unlinkDocument require the **cmisAdmin** system action. All other DocumentService functions require the **cmisUser** system action.

Function	Description	Returns
<a href="#">createDocument</a>	Creates a document in a case folder.	<a href="#">CreateDocumentResponseType</a>
<a href="#">deleteDocument</a> --or-- <a href="#">deleteDocumentAlt</a>	<p>Deletes the specified document from the CMS.</p> <p>You can either pass a document reference string, or a <a href="#">DeleteDocumentRequestType</a> element, to identify the document.**</p> <div>  <p>If a document is linked to multiple cases, and that document is deleted, it is deleted from all case folders.</p> </div>	<a href="#">DeleteDocumentResponseType</a>



Function	Description	Returns
<a href="#">deleteOrphanedFolders</a>	<p>Deletes orphaned case folders, that is, case folders whose case objects have been deleted.</p> <ul style="list-style-type: none"> <li>The request can specify that details for the deleted folders be returned, as well as a date that specifies that folders whose case objects were deleted on or before the date be deleted.</li> <li>The response can optionally return the paths to folders that failed to be deleted.</li> </ul> <p>Orphaned case folders may be deleted automatically when case objects are deleted, depending on the setting in the <b>deleteFolderOnUndeploymentOrCaseObjectDeletion</b> configuration parameter. For more information, see the "Case Folders" section in the <i>TIBCO ActiveMatrix BPM Case Data User's Guide</i>.</p>	<a href="#">DeleteOrphanedFoldersResponseType</a>
<a href="#">findDocuments</a> --or-- <a href="#">findDocumentsAlt</a>	<p>Returns references to documents that are linked to a specified case.</p> <p>You can either pass a case reference string, or a <a href="#">FindDocumentsRequestType</a> element, to identify the case.</p> <p>For more information about queries used with the findDocuments function, see <a href="#">Queries for findDocuments</a>.</p>	<a href="#">FindDocumentsResponseType</a>
<a href="#">getDocumentContent</a> --or-- <a href="#">getDocumentContentAlt</a>	<p>Returns the content of the specified document.</p> <p>You can either pass a document reference string, or a <a href="#">GetDocumentContentRequestType</a> element, to identify the document.**</p> <p>The response returns the document content, in base64Binary, as well as the mime type of the content.</p>	<a href="#">GetDocumentContentResponseType</a>
<a href="#">getDocumentMetadata</a> --or-- <a href="#">getDocumentMetadataAlt</a>	<p>Returns the metadata for the specified document.</p> <p>You can either pass a document reference string, or a <a href="#">GetDocumentMetadataRequestType</a> element, to identify the document.**</p>	<a href="#">GetDocumentMetadataResponseType</a>



Function	Description	Returns
<a href="#">getFolderContent</a> --or-- <a href="#">getFolderContentAlt</a>	<p>Returns the content of the specified case folder, and optionally, sub-folders.</p> <p>You can either pass a case reference string, or a <a href="#">GetFolderContentRequestType</a> element, to identify the case.**</p>	<a href="#">GetFolderContentResponseType</a>
<a href="#">getRepositoryInfo</a>	<p>Returns information about the CMIS repository, including which "versioning states" the repository supports.</p> <p>Some repositories support versioning (in which case, MAJOR and MINOR are typically allowed) and others don't support versioning (in which case, this operation returns "NONE" for "versioningStateSupported"). This information can be used by the application to offer only the valid choices when users attempt to import documents.</p>	<a href="#">GetRepositoryInfoResponseType</a>
<a href="#">linkDocument</a> --or-- <a href="#">linkDocumentAlt</a>	<p>Links a document to a specified case.</p> <p>You can either pass the document reference (or document specifier) and target case reference as strings, or a <a href="#">LinkDocumentRequestType</a> element, to identify the document and case.**</p> <p>The response returns either a document reference or an empty response, as follows:</p> <ul style="list-style-type: none"> <li>• If a document specifier is passed in the request (as opposed to a document reference), a document reference is returned.</li> <li>• If a document reference is passed in the request, an empty response is returned.</li> </ul> <div>  <p>If a document is linked to multiple cases, and that document is deleted, it is deleted from all case folders. Not all repositories allow you to link documents to multiple cases.</p> </div>	<a href="#">LinkDocumentResponseType</a>

Function	Description	Returns
<a href="#">moveDocument</a> --or-- <a href="#">moveDocumentAlt</a>	<p>Moves the specified document from one case folder to another case folder.</p> <p>You can either pass the document reference and source and target case references as strings, or a <a href="#">MoveDocumentRequestType</a> element, to identify the document and case.**</p> <p>The response returns the document reference of the document that was moved. Note that moving the document may cause its reference to change.</p>	<a href="#">MoveDocumentResponseType</a>
<a href="#">unlinkDocument</a> --or-- <a href="#">unlinkDocumentAlt</a>	<p>Unlinks a document from a case, which removes the document from the case folder.</p> <p>You can either pass the document and case references as strings, or an <a href="#">UnlinkDocumentRequestType</a> element, to identify the document and case.**</p>	<a href="#">UnlinkDocumentResponseType</a>

\*\* Some CMIS servers use characters in document and case references that need escaping when included in URLs. If this is an issue, you can URL-encode the references, or you can use the alternative function that passes an element.

## Queries for findDocuments

The query syntax that is allowed for use with the findDocuments function depends on the CMS you are using.

Also note that the specific queries allowed may need to be configured on your CMS before they can be used in the [DocumentService](#) functions. Check your vendor-specific CMS documentation concerning configuring query options.

The following lists the typical query constructs that can be used to find documents in a CMS.

### CMIS Properties

The CMIS properties that can be queried are:

- cmis:name
- cmis:creationDate
- cmis:createdBy
- cmis:lastModifiedBy
- cmis:lastModificationDate

Note that some CMS repositories may allow additional properties to be queried.

### Unsupported Constructs

The following are CMIS query constructs that are not currently supported:

- Join

- orderBy
- IN\_TREE
- IN\_FOLDER

### Wildcard Characters

The specific wildcard characters that can be used in queries depends on the predicate you are using, as well as the CMS being used. Consult the documentation for your CMS. The following are common wildcard characters (depending predicate and CMS):

- % - Matches any numbers of characters.
- \* - Matches any numbers of characters.
- \_ - Matches a single character.
- ? - Matches a single character.

### Examples

The following shows example queries. Consult the documentation for your CMS for specific allowable constructs.

- cmis:name LIKE 'P%'
- cmis:name LIKE 'errors\_.jpg'
- cmis:name LIKE '%readme%'
- cmis:createdBy LIKE 'admin'
- cmis:createdBy NOT LIKE 'susieq'
- cmis:lastModifiedBy LIKE 'admin'
- cmis:creationDate = '2014-10-06T00:00:00'
- cmis:creationDate < '2014-10-06T00:00:00'
- cmis:name LIKE 'P%' AND cmis:creationDate < '2014-10-06T00:00:00'
- (cmis:name LIKE 'P%' OR cmis:name LIKE 'D%') AND cmis:createdBy LIKE 'admin'

## DynamicResourceQueryService

The DynamicResourceQueryService contains functions to reset dynamic resource queries.

The table below lists the functions available from the [DynamicResourceQueryService](#).

Function	Description	Returns
<a href="#">resetQueries</a>	Resets the dynamic resource queries for all model versions.	<a href="#">ExecuteQueryResponse</a>
<a href="#">resetQueryWithVersion</a>	Resets the dynamic resource queries for the specified model version.	<a href="#">ExecuteQueryResponse</a>

If automatic resource queries have been generated as a result of a change to an organization model or resource, but you are experiencing issues such as work items not being offered/allocated to correct people, or not being distributed because queries are being re-evaluated, you can use these functions to reset the dynamic resource queries.




- These functions should only be used with advice from TIBCO Support.
- Using these functions may result in a temporary inconsistency in the allocation of work items that use resource queries until the reset operation is completed.
- These function operate asynchronously, so receiving a successful response does not indicate that the reset operation is complete.

## EntityResolverService

The EntityResolverService contains a function to locate a resource by LDAP DN.

The table below lists the functions available from the [EntityResolverService](#).

Function	Description	Returns
<a href="#">findByDN</a>	Locates a Resource by their LDAP DN. The caller may supply the LDAP Alias in order to reduce the search results. The caller may optionally specify that the response should include detailed information on those users that match the criteria (the default is to only return a count of the users that match the criteria).  <div>  <p>If you want to search by entity or name, use the <a href="#">findByEntity</a> or <a href="#">findByName</a> function.</p> <p><b>Required system action:</b> resolveResource</p> </div>	<a href="#">LookupUserResponseType</a>

## ExporterService

The ExporterService contains functions to export and import defined LDAP containers, resource mappings, and push destinations.

The table below lists the functions available from the [ExporterService](#).

Function	Description	Returns
<a href="#">exportResources</a>	Exports defined LDAP containers, resource mappings and push destinations, either for backup or for subsequent import to other environments.  <b>Required system action:</b> exportLDAPAdmin	<a href="#">ExportResource</a>
<a href="#">importResources</a>	Imports a previously exported object containing details of the exported LDAP containers, resource mappings and push destinations.  <b>Required system action:</b> importLDAPAdmin	boolean

## GlobaldataAdminService

The GlobaldataAdminService includes functions to get information about and administer case models.

The table below lists the functions available from the [GlobaldataAdminService](#).

For more information about administering case models, see [Case Data Models](#).

Function	Description	Returns
<a href="#">getAuditInfo</a>	Returns all the CREATE/UPDATE/DROP scripts that have been edited for a case model, including, who performed the operation (for example, the system or the user) and when. It also returns any previously generated scripts. For example, you may have five previous update scripts if you have deployed five different versions of the same case model major version.  <b>Required system action:</b> accessGlobalDataScripts	<a href="#">GetAuditInfoResponseType</a>
<a href="#">getCaseClassInfo</a> --or-- <a href="#">getCaseClassInfo Alt1</a> --or-- <a href="#">getCaseClassInfo Alt2</a>	Lists all the case classes and their attributes for a specific case model. This information can then be used when searching for case data.  The request can specify any of the following: <ul style="list-style-type: none"> <li>the name of the application that generated the case model, as well as the major version number of the application, specified in TIBCO Business Studio at design-time. You can get the application name and major version number by running <a href="#">getCaseModelBasicInfo</a> (see below).</li> <li>--or--</li> <li>the case model ID. The case model ID can be obtained using the <a href="#">getCaseModel</a> or <a href="#">getCaseModelBasicInfo</a> function (see below).</li> <li>--or--</li> <li>neither the application name nor the case model ID. This results in all case classes on the system being returned.</li> </ul> The response returns all the case classes and attribute information for the specified case model.  <b>Required system action:</b> readGlobalData	<a href="#">GetCaseClassInfoResponseType</a>
<a href="#">getCaseModel</a>	Lists the CREATE/UPDATE/DROP database scripts for all deployed case models. Use this to retrieve the database scripts for the case models.  <b>Required system action:</b> accessGlobalDataScripts	<a href="#">GetCaseModelResponseType</a>
<a href="#">getCaseModelBasicInfo</a>	Lists basic information about all deployed case models. Use this function to discover all case models.  <b>Required system action:</b> readGlobalData	<a href="#">GetCaseModelBasicInfoResponseType</a>
<a href="#">notifyCleaned</a>	Notifies the system when the case model database has been manually cleaned up after auto remove of tables failed.  <b>Required system action:</b> administerGlobalDataScripts	<a href="#">OperationDetailsType</a>

Function	Description	Returns
<a href="#">notifyCreated</a>	<p>Notifies the system that the CREATE script has been executed on the database and the database schema for the case model has been generated.</p> <p><b>Required system action:</b> administerGlobalDataScripts</p>	<a href="#">OperationDetailsType</a>
<a href="#">notifyDropped</a>	<p>Notifies the system that the DROP script has been executed on the database and the database schema for the case model has been dropped.</p> <p><b>Required system action:</b> administerGlobalDataScripts</p>	<a href="#">OperationDetailsType</a>
<a href="#">notifyFreeze</a>	<p>Freezes the case model.</p> <p>Freezing a case model means that no further updates can be made to any of the database scripts and that no newer minor version of this case model can be deployed to update the frozen model.</p> <p><b>Required system action:</b> administerGlobalDataScripts</p>	<a href="#">OperationDetailsType</a>
<a href="#">notifyUnfreeze</a>	<p>Unfreezes the case model.</p> <p><b>Required system action:</b> administerGlobalDataScripts</p>	<a href="#">OperationDetailsType</a>
<a href="#">notifyUpdated</a>	<p>Notifies the system that the UPDATE script has been executed on the database the database schema for the case model has been updated.</p> <p><b>Required system action:</b> administerGlobalDataScripts</p>	<a href="#">OperationDetailsType</a>

Function	Description	Returns
<a href="#">updateDBScripts</a>	<p>Updates the existing CREATE/UPDATE/DROP database scripts with any required changes.</p> <p>Although editing the scripts is possible, TIBCO does not recommend this unless it is essential. There are only a few changes that you are allowed to make.</p> <p>The request must specify the case model ID whose database scripts need updating and the new scripts that you want to use. You must include the entire script.</p> <div>  <ul style="list-style-type: none"> <li>No validation is performed on the scripts when you edit them in Data Admin. This means it is possible to make invalid changes to your scripts.</li> <li>If the CREATE script is changed (for example, index or foreign key names), you must edit the UPDATE/DROP scripts, to reflect the changes.</li> <li>If you have edited an UPDATE script and the business object model is updated, the new UPDATE script that is generated restores the original values and settings. This means you must manually edit the newly generated UPDATE script with your original changes.</li> </ul> </div> <p><b>Required system action:</b> administerGlobalDataScripts</p>	<a href="#">OperationDetailsType</a>

## GlobaldataService

The GlobaldataService includes functions to create, read, update and delete case data. You can also perform searches on case data based on specific criteria.

The table below lists the functions available from the [GlobaldataService](#).

Function	Description	Returns
<a href="#">createCase</a>	<p>Creates new case data for a given case class in the case data store.</p> <p>The request contains the case type, model version and the case data to be created.</p> <p>The response lists the case references of the newly created case data.</p> <p><b>Required system action:</b> createGlobalData</p>	<a href="#">CreateCaseResponseType</a>

Function	Description	Returns
<a href="#">createDataView</a>	<p>Used to specify a search query to be stored as a data view that can be retrieved later using the <a href="#">getCaseReferencesForDataViewByID</a> function (see below).</p> <p>The request specifies the search query to be used in the data view. It identifies the case class and the search condition.</p> <p>The response returns the data view ID.</p> <p><b>Required system action:</b> <a href="#">manageDataViews</a></p>	<a href="#">CreateDataViewResponseType</a>
<a href="#">deleteCaseByCID</a>	<p>Deletes existing case data from the case data store for a given case identifier (CID), for example, a customer ID.</p> <p>A case identifier is a special type of attribute that can be used to uniquely identify a case class. A case class must have at least one case identifier and may have many, depending on the nature of the case data. Case identifiers are specified at design-time when modeling your case data in TIBCO Business Studio.</p> <p>The request must contain the case identifier for the case class. It also specifies the case type, case model version, and the case data to be deleted. You can specify more than one case data to be deleted.</p> <p><b>Required system action:</b> <a href="#">deleteGlobalData</a></p>	Empty response (no JSON payload)
<a href="#">deleteCaseByRef</a> --or-- <a href="#">deleteCaseByRefAlt1</a> --or-- <a href="#">deleteCaseByRefAlt2</a>	<p>Deletes existing case data for a given case reference from the case data store.</p> <p>A case object is accessed by reference. Whenever a case object is created, a unique case reference is also created. Providing the case reference, provides access to the case data object.</p> <p>The request contains the case references of the case data to be deleted. You can specify more than one case reference. You can obtain case references using <a href="#">findAllCases</a> (see below).</p> <p><b>Required system action:</b> <a href="#">deleteGlobalData</a></p>	Empty response (no JSON payload)
<a href="#">deleteDataViewByID</a>	<p>Deletes a previously saved data view using the view ID.</p> <p>The request contains the data view to delete by specifying a data view ID.</p> <p><b>Required system action:</b> <a href="#">manageDataViews</a></p>	Empty response (no JSON payload)



Function	Description	Returns
<a href="#">deleteDataViewByName</a>	<p>Deletes a previously saved data view using the view name.</p> <p>The request contains the data view to delete by specifying a data view name.</p> <p><b>Required system action:</b> manageDataViews</p>	Empty response (no JSON payload)
<a href="#">editDataView</a>	<p>Edits a previously saved data view.</p> <p>The request contains a data view ID or name of the data view to edit, as well as the changes to make to the data view.</p> <p><b>Required system action:</b> manageDataViews</p>	Empty response (no JSON payload)
<a href="#">findAllCases</a>	<p>Finds all of the case references for a given case class. A case object is accessed by reference. Whenever a case object is created, a unique case reference is also created. Providing the case reference, provides access to the case data object.</p> <p><b>Required system action:</b> readGlobalData</p>	<a href="#">SearchResults</a>
<a href="#">findCaseByCID</a>	<p>Finds the case reference of the case data using a case identifier (CID), for example, a customer ID.</p> <p>A case identifier is a special type of attribute that can be used to uniquely identify a case class. A case class must have at least one case identifier and may have many, depending on the nature of the case data. Case identifiers are specified at design-time when modeling your case data in TIBCO Business Studio.</p> <p>The request contains the case type, case model version, and the case identifier of the case data to be found. Your request must contain only one case identifier, unless you have set any CIDs to Composite. In this case, you must include all of the CIDs set as Composite in the request.</p> <p>The response lists the case references of the case data for the specified case identifier.</p> <p><b>Required system action:</b> readGlobalData</p>	<a href="#">CaseReferenceType</a>

Function	Description	Returns
<a href="#">findCaseByCriteria</a>	<p>Finds case data that matches specified criteria. For the valid syntax of the criteria, see the <i>TIBCO Business Data Services Guide</i>.</p> <p>The request contains the case type (the fully qualified name of the case class, for example, com.example.gddemo.Car), case model version and the criteria the case data must match.</p> <p>The response lists the case references of the case data that matches the criteria.</p> <p><b>Required system action:</b> readGlobalData</p>	<a href="#">FindCaseByCIDResponse</a>
<a href="#">findCaseByExample</a>	<p>Finds the case references of case data using a searchable case attribute. Searchable attributes can be used to find case data that match specified attribute values. Case attributes are specified as searchable at design-time when modeling your case data in TIBCO Business Studio.</p> <p><b>Required system action:</b> readGlobalData</p>	<a href="#">SearchResults</a>
<a href="#">getCaseReferencesForDataViewByAdhocView</a>	<p>Returns the case references that match the specified case data search query.</p> <p>You can use this function to provide ad-hoc searches of all case data or to search data views that have been created and stored previously. The search query enables you to specify sophisticated search conditions based on a search schema.</p> <p><b>Required system action:</b> readGlobalData</p>	<a href="#">SearchResults</a>
<a href="#">getCaseReferencesForDataViewByID</a> --or-- <a href="#">getCaseReferencesForDataViewByIDAlt</a>	<p>Returns the case references that match the specified data view ID.</p> <p><b>Required system action:</b> readGlobalData</p>	<a href="#">SearchResults</a>
<a href="#">getCaseReferencesForDataViewByName</a> --or-- <a href="#">getCaseReferencesForDataViewByNameAlt</a>	<p>Returns the case references that match the specified data view name.</p> <p><b>Required system action:</b> readGlobalData</p>	<a href="#">SearchResults</a>
<a href="#">getCaseState</a>	<p>Returns the state of one or more specified cases. You can also specify that all allowable case state values be returned.</p> <p><b>Required system action:</b> readGlobalData</p>	<a href="#">GetCaseStateResponseType</a>

Function	Description	Returns
<a href="#">getCaseSummary</a> --or-- <a href="#">getCaseSummaryAlt</a>	Returns a summary of a case object, including information such as the case identifier and state.  <b>Required system action:</b> readGlobalData	<a href="#">GetCaseSummaryResponseType</a>
<a href="#">getDataViewCategories</a>	Returns the data view categories that are defined.  The request specifies whether to list all of the categories or the categories at a specific level in the hierarchy.  <b>Required system action:</b> readGlobalData	<a href="#">GetDataViewCategoriesResponseType</a>
<a href="#">getDataViewDetailsByApp</a>	Retrieves the data view for the specified application, for example, com.example.gddemo-2.  <b>Required system action:</b> readGlobalData	<a href="#">GetDataViewDetailsResponseType</a>
<a href="#">getDataViewDetailsByCaseClass</a>	Retrieves the data view for the specified case class.  You must pass the fully qualified name of the case class, for example, com.example.gddemo.Car. You can obtain class names by using <a href="#">getCaseClassInfo</a> .  <b>Required system action:</b> readGlobalData	<a href="#">GetDataViewDetailsResponseType</a>
<a href="#">getDataViewDetailsByCategory</a>	Retrieves the data view for the specified category.  <b>Required system action:</b> readGlobalData	<a href="#">GetDataViewDetailsResponseType</a>
<a href="#">getDataViewDetailsByID</a>	Retrieves the data view for the specified view ID.  <b>Required system action:</b> readGlobalData	<a href="#">GetDataViewDetailsResponseType</a>
<a href="#">getDataViewDetailsByName</a>	Retrieves the data view for the specified view name.  The following wildcards can be used to search for data views by name: <ul style="list-style-type: none"> <li>• '?' or '_' matches any single character.</li> <li>• '*' or '%' matches zero or more characters.</li> </ul> Wildcards can be escaped using the "\" character. For example, "\%". Escape characters can be escaped using "\\" which evaluates to matching "\".  <b>Required system action:</b> readGlobalData	<a href="#">GetDataViewDetailsResponseType</a>

Function	Description	Returns
<a href="#">getDataViewDetailsByUncategorized</a>	<p>Returns all the data views that have been created using the API without specifying a category.</p> <p>If you create a data view in the Openspace or Workspace client without specifying a category, they are placed in a category called UNCATEGORIZED. This function does <i>not</i> return any data views created without a category in Openspace or Workspace.</p> <p><b>Required system action:</b> readGlobalData</p>	<a href="#">GetDataViewDetailsResponse</a>
<a href="#">linkCase</a> --or-- <a href="#">linkCaseAlt</a>	<p>Creates an association between case data. This is necessary when navigating between case data. For example, in an insurance company, you may want to link a customer's case reference number with their orders.</p> <p>You can then use either the <code>navigateCase</code> or the <code>navigateCaseByCriteria</code> function (see below) to search for orders belonging to that particular customer.</p> <p>The request identifies the case reference of the case data to associate and the name of the association.</p> <p>The response returns a new case reference for the association. The case reference is incremented by one. For example, if you have <code>BDS-2-com.example.gddemo.Customer-1-0</code>, and you link it with some case data, the new case reference would be <code>BDS-2-com.example.gddemo.Customer-1-1</code>.</p> <p><b>Required system action:</b> updateGlobalData</p>	<a href="#">LinkResponseType</a>

Function	Description	Returns
<a href="#">navigateCase</a>	<p>Used to perform searches on case data that have been associated using the linkCase function (see above).</p> <p>For example, in an insurance company, you may want to link a customer's case reference number with their claims. You can use this function to search for claims belonging to that particular customer. You can use the navigateCaseByCriteria function (see below) to perform more sophisticated search queries. For example, you could search for all claims for the customer where the quantity was greater than 100.</p> <p>The request identifies the case reference of the source case data and the name of the association that was specified when the association was created using the linkCase function (see above).</p> <p>The response returns the case references of the case data that matches the search.</p> <p><b>Required system action:</b> readGlobalData</p>	<a href="#">SearchResults</a>
<a href="#">navigateCaseByCriteria</a>	<p>Used to perform searches, using the specified criteria, on case data that have been associated using the linkCase function (see above).</p> <p>The request identifies the case reference of the source case data and the name of the association that was specified when the association was created using the linkCase function (see above).</p> <p>The response returns the case references of the case data that matches the search criteria.</p> <p><b>Required system action:</b> readGlobalData</p>	<a href="#">SearchResults</a>
<a href="#">navigateCaseDetailed</a> --or-- <a href="#">navigateCaseDetailedAlt</a>	<p>Used to perform searches on case data that have been associated using the linkCase function (see above). This function differs from navigateCase (see above), as this function navigates association links from the given source object for all roles, or a specific role, with the ability to include case summary in the response.</p> <p><b>Required system action:</b> readGlobalData</p>	<a href="#">NavigateCaseDetailedResponseType</a>

Function	Description	Returns
<a href="#">readCase</a> --or-- <a href="#">readCaseAlt1</a> --or-- <a href="#">readCaseAlt2</a>	<p>Reads existing case data from the case data store.</p> <p>The request contains the case reference to the case data object you want to read. Whenever a case object is created, a unique case reference is also created. Providing the case reference, provides access to the case data object, for example, BDS-2-com.example.gddemo.Customer-1-0. You can specify more than one. You can obtain case references using the <code>findAllCases</code>, <code>findCaseByCID</code>, <code>findCaseByCriteria</code>, and <code>findCaseByExample</code> functions (see above).</p> <p><b>Required system action:</b> <code>readGlobalData</code></p>	<a href="#">ReadCaseResponseType</a>
<a href="#">unlinkCase</a> --or-- <a href="#">unlinkCaseAlt</a>	<p>Removes an existing association between case data.</p> <p>The request identifies the case references of the case data whose association you want to remove.</p> <p>The response returns a case reference for the association. The new case reference version is incremented by one. For example, if you have BDS-2-com.example.gddemo.Customer-1-2, and you unlink it from some case data, the new case reference would be BDS-2-com.example.gddemo.Customer-1-3.</p> <p><b>Required system action:</b> <code>updateGlobalData</code></p>	<a href="#">LinkResponseType</a>
<a href="#">updateCase</a> --or-- <a href="#">updateCaseAlt</a>	<p>Updates existing case data in the case data store.</p> <p>The request contains the pairs of case references and case data to be updated. It is important that the case reference used is the case reference returned by the <code>readCase</code> function (see above), so that the optimistic locking works correctly. You can update as many case data as you want.</p> <p>The response lists the new case reference of the updated case data. The case reference version is incremented by one. For example, for BDS-2-com.example.gddemo.Customer-1-0, the new case reference would be BDS-2-com.example.gddemo.Customer-1-1.</p> <p><b>Required system action:</b> <code>updateGlobalData</code></p>	<a href="#">UpdateCaseResponseType</a>

## OrgEntityConfigService

The `OrgEntityConfigService` contains functions to manage resource configuration attributes and to get or set them for individual resources.

The table below lists the functions available from the [OrgEntityConfigService](#).

Function	Description	Returns
<a href="#">deleteOrgEntityConfigAttributes</a>	Deletes one or more configuration attributes for a resource.  <b>System action required:</b> autoOpenNextWorkItem**	<a href="#">DeleteOrgEntityConfigAttributesResponseType</a>
<a href="#">getOrgEntityConfigAttributes</a>	Gets the configuration attribute information currently defined for a resource.  <b>System action required:</b> None	<a href="#">GetOrgEntityConfigAttributesResponseType</a>
<a href="#">getOrgEntityConfigAttributesAvailable</a>	Returns all available attributes that can be used.  This returns a subset of the attributes returned by calling <a href="#">getOrgEntityConfigAttributes</a> (see above), including the name, description and whether it is read only.  <b>System action required:</b> None	<a href="#">GetOrgEntityConfigAttributesAvailableResponseType</a>
<a href="#">setOrgEntityConfigAttributes</a>	Sets one or more configuration attributes for a resource.  This can be used to either create a new attribute or update an existing one.  <b>System action required:</b> autoOpenNextWorkItem**	<a href="#">SetOrgEntityConfigAttributesResponseType</a>

\*\* This system action is required to delete or set the **WorkItemAutoOpen** attribute, otherwise, no system action is required for [deleteOrgEntityConfigAttributes](#) or [setOrgEntityConfigAttributes](#).

## OrgModelService

The OrgModelService contains functions to get information about organization models, as well as get or set information for calendars, push destinations, extension points, and candidate queries that are tied to a particular organization model.

The table below lists the functions available from the [OrgModelService](#).

Function	Description	Returns
<a href="#">deleteCalendarReference</a>	Deletes any calendar reference from the identified organization model entity.  Also see: <a href="#">deleteCalendarReferences</a> in <a href="#">WorkCalService</a>  <b>System action required:</b> browseModel	Empty response (no JSON payload)
<a href="#">deleteGroupCandidateQuery</a>	Deletes any candidate query assignment from the identified group.  <b>System action required:</b> resourceAdmin or LDAPAdmin*	Empty response (no JSON payload)
<a href="#">deletePositionCandidateQuery</a>	Deletes any candidate query assignment from the identified position.  <b>System action required:</b> resourceAdmin or LDAPAdmin*	Empty response (no JSON payload)

Function	Description	Returns
<a href="#">deletePushDestination</a>	<p>Deletes the identified push destination assignment from the identified organization model entity.</p> <p>See also: <a href="#">updatePushDestinations</a> (see below)</p> <p><b>System action required:</b> <a href="#">writePushDestinations</a></p>	Empty response (no JSON payload)
<a href="#">getCalendarReferences</a>	<p>Gets the calendar references for the identified organization model entity.</p> <p>The response is an ordered collection of calendar references obtained by navigating the organization model hierarchy to which the identified model entity belongs. Each entry identifies the organization model entity that holds the calendar reference.</p> <p><b>System action required:</b> <a href="#">browseModel</a> or <a href="#">readCalendars</a></p>	<a href="#">GetCalendarReferencesResponseType</a>
<a href="#">getOrgModel</a>	<p>Requests the details of the entities that form the organization model identified by the given major version.</p> <p><b>System action required:</b> <a href="#">browseModel</a></p>	<a href="#">GetOrgModelResponseType</a>
<a href="#">getOrgModelEntity</a>	<p>Requests the details of the specified organization model entity.</p> <p>This operation can be used in conjunction with the <a href="#">getOrgModelRoots</a> function (see below) to "drill down" into an organization model structure to obtain details about one or more organizational entities.</p> <p><b>System action required:</b> <a href="#">browseModel</a></p>	<a href="#">GetOrgModelEntityResponseType</a>
<a href="#">getOrgModelRoots</a>	<p>Requests the details of the entities that form the root elements of the organization model identified by the given major version. These include organizations, groups, locations, capabilities, privileges, resource attributes, and model templates.</p> <p>If the organization model is very large, you can use this function instead of <a href="#">getOrgModel</a> (see above) to improve performance. This allows you to "drill down" into the organization model structure to retrieve specific fragments of the model. You can then use <a href="#">getOrgModelEntity</a> (see above) to retrieve details about specific organizational entities.</p> <p><b>System action required:</b> <a href="#">browseModel</a></p>	<a href="#">GetOrgModelRootsResponseType</a>
<a href="#">listOrgModelVersions</a>	<p>Lists all major versions of an organization model. For each major version, the organization model deployments that make up that version are detailed, giving their full version number, name and date deployed.</p> <p><b>System action required:</b> <a href="#">browseModel</a></p>	<a href="#">ListOrgModelVersionsResponseType</a>



Function	Description	Returns
<a href="#">setCalendarReference</a>	<p>Assigns a calendar reference to the identified organization model entity. The assignments overwrite any existing assignments to the same organization model entities.</p> <p><b>System action required:</b> writeCalendars</p>	Empty response (no JSON payload)
<a href="#">setCalendarReferences</a>	<p>Assigns calendar references to the identified organization model entities.</p> <p>The assignments overwrite any existing assignments to the same organization model entities.</p> <p><b>System action required:</b> writeCalendars</p>	Empty response (no JSON payload)
<a href="#">setExtensionPoint</a> --or-- <a href="#">setExtensionPointAlt</a>	<p>There are two signatures for this function: In one sets or updates the extension point configuration for the identified organization model version, and the other removes the extension points for the identified organization model version.</p> <p>When setting or updating extension points:</p> <ul style="list-style-type: none"> <li>• The request must specify a reference to an organization model template that will be used to dynamically generate new organizational fragments directly below it.</li> <li>• Any existing configuration for the identified entities are overwritten. Attempting to overwrite an existing configuration for which model template instances have been generated causes a <code>DirectoryEngineFault</code>. Attempting to set the configuration for an organization model entity to which the caller does not have access causes a <code>SecurityFault</code>.</li> <li>• The new organizational fragments are generated at the date/time specified in the <code>ExtensionPointProcessStart</code> property in <code>DE.Properties</code>. For information about all extension point-related properties, see the <i>TIBCO ActiveMatrix BPM Administration</i> guide.</li> </ul> <p><b>System action required:</b> LDAPAdmin or organizationAdmin (both system actions allow equal access)</p>	Empty response (no JSON payload)
<a href="#">setGroupCandidateQuery</a>	<p>Sets or updates the candidate query assignments to the identified group.</p> <p>Each group may only hold one candidate query. Any existing assignment is overwritten by a new assignment. Candidate queries span all organization model versions in which the identified organization model exists. Therefore, the request does not include a model version.</p> <p><b>System action required:</b> resourceAdmin or LDAPAdmin*</p>	Empty response (no JSON payload)



Function	Description	Returns
<a href="#">setPositionCandidateQuery</a>	<p>Sets or updates the candidate query assignments to the identified position.</p> <p>Each position may only hold one candidate query. Any existing assignment is overwritten by a new assignment. Candidate queries span all organization model versions in which the identified organization model exists. Therefore, the request does not include a model version.</p> <p><b>System action required:</b> resourceAdmin or LDAPAdmin*</p>	Empty response (no JSON payload)
<a href="#">setPushDestination</a>	<p>Modifies the push destination for a specified organization model entity.</p> <p>The value(s) of the push destination may be specified as 'hard-coded' collection of values (such as email addresses), or they may be derived from the Resource Attribute identified in the request.</p> <p>Existing push destination values are overwritten.</p> <p>To get existing push destinations, use the <a href="#">getResource</a> function, for resources, or the <a href="#">getOrgModel</a> function (see above), for organization model entities other than resources.</p> <p><b>System action required:</b> writePushDestinations</p>	Empty response (no JSON payload)
<a href="#">updatePushDestinations</a>	<p>Updates the push destinations for the identified organization model entity.</p> <p>The update can include the deletion of existing push destinations. Each push destination is identified by its name, channel-type and channel-id.</p> <p><b>System action required:</b> writePushDestinations</p>	Empty response (no JSON payload)

\* LDAPAdmin allows access to all positions and LDAP containers, in the case where restrictions have been placed on access to organizations and LDAP containers. For more information, see [Organization Relationships](#).

## OrgResourceService

The OrgResourceService contains functions used to manage resources, such as creating, deleting, finding, and updating resources.

The table below lists the functions available from the [OrgResourceService](#).

Function	Description	Results
<a href="#">archivedResources</a>	<p>Retrieves archive records of resources that have been deleted.</p> <p>When resources are deleted, an archive of their records are maintained in order to allow process and work-item administration to be completed. This service allows those archive records to be retrieved. The results can be paged, or the entire result set can be returned in one call.</p> <p><b>System action required:</b> <code>resolveResource</code></p>	<a href="#">GetDeletedResourcesResponse</a>
<a href="#">createResource</a>	<p>Creates one or more resources, which causes a GUID to be assigned to each resource, and allows the resources to log into ActiveMatrix BPM applications.</p> <p>Typically, you will use the <a href="#">listCandidateResources</a> function to get a list of candidate resources in a particular LDAP container, then use this function to create the desired resources from those candidates. Once you have created resources, they can be mapped to the desired groups or positions using the <code>updateResource</code> function (see below).</p> <p>If the request identifies a resource that is already created, the already-present attribute in the response is set to "false".</p> <p><b>System action required:</b> <code>resourceAdmin</code> or <code>createResourceAdmin</code></p>	<a href="#">CreateResourceResponse</a>
<a href="#">deleteResource</a>	<p>Deletes one or more resources and their associated organization mappings (if any).</p> <div>  <p>If you want to delete a large number of resources, you must delete 83 resources or less at a time. If more than 83 resources are selected and deleted at one time, a <code>FULL</code> <code>headError</code> message displays and the request fails.</p> </div> <p><b>System action required:</b> <code>resourceAdmin</code> or <code>deleteResourceAdmin</code></p>	<a href="#">DeleteResourceResponse</a>
<a href="#">findByEntity</a>	<p>Finds all resources associated with one or more specified organization model entities.</p> <div>  <p>This function cannot be used to perform a user lookup using an LDAP DN. To do that, use the <a href="#">findByDN</a> function.</p> </div> <p>Also see the <code>findByName</code> function (below).</p> <p><b>System action required:</b> <code>resolveResource</code></p>	<a href="#">FindResourcesResponse</a>


Function	Description	Results
<a href="#">findByName</a>	<p>Finds all resources associated with one or more specified names.</p> <p>Also see the <a href="#">findByEntity</a> function (above).</p> <p><b>System action required:</b> <a href="#">resolveResource</a></p>	<a href="#">FindResourcesResponse</a>
<a href="#">getResource</a>	<p>Returns details about one or more specified resources, such as positions and groups in which the resource is a member, privileges and capabilities held by the resource, the LDAP entry from which the resource is derived, and so on.</p> <p><b>System action required:</b> <a href="#">resolveResource</a></p>	<a href="#">GetResourceResponseType</a>
<a href="#">purgeDeletedResources</a>	<p>Permanently deletes the archived resource records. The function allows selected records to be purged, or all records to be purged. The response states how many records were purged.</p> <p>When resources are deleted, a summary of their records are maintained in order to allow process and work-item administration to be completed. This function causes those deletion records to be permanently purged.</p> <p>After being purged, the user is still a candidate resource, and can be created again (assuming the user is still in the LDAP source).</p> <p><b>System action required:</b> <a href="#">resourceAdmin</a> or <a href="#">deleteResourceAdmin</a></p>	<a href="#">PurgeDeletedResourceResponseType</a>
<a href="#">updateResource</a>	<p>Updates the given properties of the specified resource or resources. This allows you to add or remove a resource's privileges, capabilities, and resource attributes, and so on. You can also add or remove a resource from positions and groups with this function.</p> <p>Only the properties passed in the request are updated; those not passed in the request are left unchanged.</p> <p>Multiple resources can be updated in one request, using a separate <code>&lt;resource&gt;</code> element for each.</p> <p><b>System action required:</b> <a href="#">resourceAdmin</a> or <a href="#">createResourceAdmin</a></p>	<a href="#">UpdateResourceResponseType</a>

## PageFlowService

The PageFlowService contains functions used to get information about, and interact with, deployed pageflows.

The table below lists the functions available from the [PageFlowService](#).

All PageFlowService functions require the [executeBusinessService](#) system action, except for [listPageFlows](#), which requires [listBusinessServices](#).


Function	Description	Returns
<a href="#">cancelPageFlow</a>	Cancels a given pageflow instance.	<a href="#">CancelPageFlowResponseType</a>
<a href="#">injectPageFlow</a>	Injects data into an in-progress pageflow.	<a href="#">InjectPageFlowEventResponseType</a>
<a href="#">listPageFlows</a>	<p>Lists all deployed pageflows.</p>  <p>Pageflow processes that have been deployed as business services are not included in the response.</p>	<a href="#">ListPageFlowsResponseType</a>
<a href="#">startPageFlow</a>	Starts an instance of a deployed pageflow.	<a href="#">StartPageFlowResponseType</a>
<a href="#">updatePageFlow</a>	Updates a specified pageflow instance.	<a href="#">UpdatePageFlowResponseType</a>

## ProcessManagementService

The ProcessManagementService contains functions to manage process templates and process instances.

The table below lists the functions available from the [ProcessManagementService](#).

For more information about processes, see [Processes](#).

Function	Description	Returns
<a href="#">cancelAdhocActivity</a>	<p>Cancels the currently active ad-hoc activity.</p>  <p>Cancelling a process instance also cancels all currently active ad-hoc activities associated with the cancelled process instance.</p> <p><b>Required system action:</b> StartAndCancelAdhocActivity</p>	<a href="#">GeneralStatusType</a>
<a href="#">cancelProcessInstance</a>	<p>Cancels a specific process instance.</p> <p>If successful, returns "OK" in the response body.</p> <p><b>Required system action:</b> cancelProcessInstance</p>	String
<a href="#">cancelProcessInstances</a>	<p>Cancels all process instances for the specified process templates.</p> <p>Returns the number of process instances that were cancelled.</p> <p><b>Required system action:</b> bulkCancelProcessInstances</p>	int

Function	Description	Returns
<a href="#">checkPurgeTerminatedProcessInstances</a>	<p>Checks the status of a purge job started with the <code>purgeTerminatedProcessInstances</code> function (see below).</p> <p><b>Required system action:</b> <code>bulkPurgeProcessInstances</code></p>	<a href="#">CheckPurgeTerminatedProcessInstancesResponseType</a>
<a href="#">clearMigrationRules</a>	<p>Clears all currently set process instance migration rules for the specified process templates.</p> <p>If successful, returns "OK" in the response body.</p> <p>For information about process migration, see <a href="#">Migrating a Process Instance to a Different Version</a>.</p> <p><b>Required system action:</b> <code>handleProcessMigration</code></p>	String
<a href="#">createProcessInstance</a>	<p>Creates (starts) a process instance using a starter operation.</p> <p>This function is used to create an instance of a process that has a start event with a trigger type of "None". If the process has a start event with a trigger type of "Message", it must be started either using a business service, if it was published as a business service, or by using the <code>startProcessIncomingReceiveTask</code> function (see below), if it was published as a REST Service.</p> <p>Returns the ID of the created process instance.</p> <p><b>Required system action:</b> <code>startprocess</code></p>	String
<a href="#">decodeProcessId</a>	<p>Decodes "process" IDs into "database" IDs.</p> <p>Process IDs written to the log and returned from Process Manager are in the form "pvm:nnnn". Whereas, PVM engine tables use database IDs (dbId). In order to debug PVM engine tables, you can use this function to correlate the process IDs with the database IDs.</p> <p><b>Required system action:</b> <code>queryProcessInstance</code></p>	<a href="#">ProcessIdDBInfo</a>
<a href="#">getActivityInstanceStatus</a>	<p>Gets the status of an activity for a particular process instance.</p> <p>The response returns status information for the specified activity. This includes the <code>activityID</code>, which is used when setting a deadline time on an activity - see the <code>setDeadlineExpiration</code> function (see below).</p> <p><b>Required system action:</b> <code>queryProcessInstance</code></p>	<a href="#">ActivityInstanceStatus</a>

Function	Description	Returns
<a href="#">getAvailableProcessInstanceVariables</a>	<p>Gets the details for all variables (or for a specific variable) that are available to the failed activity for a particular halted process instance.</p> <p>Also see the <a href="#">setAvailableProcessInstanceVariables</a> function (below).</p> <p><b>Required system action:</b> haltedProcessAdministration</p>	<a href="#">AvailableProcessInstanceVariable sType</a>
<a href="#">getMigrationPoints</a>	<p>Lists the permissible process instance migration points for the specified process templates. (Migration points are the points in the process from which a process instance can be migrated to a different version.)</p> <p>For information about process migration, see <a href="#">Migrating a Process Instance to a Different Version</a>.</p> <p><b>Required system action:</b> handleProcessMigration</p>	<a href="#">MigrationPointT ypes</a>
<a href="#">getParameterValue</a>	<p>Gets the value of a specific parameter (also referred to as "custom attributes") for a particular process instance.</p> <p><b>Required system action:</b> queryProcessInstance</p>	<a href="#">GetParameterVal ueOutputType</a>
<a href="#">getProcessInstanceStatus</a>	<p>Gets the status of a particular process instance.</p> <p><b>Required system action:</b> queryProcessInstance</p>	<a href="#">ProcessInstance</a>
<a href="#">getProcessInstanceSummary</a>	<p>Gets a full summary report on a specific process instance (this operation is most useful for debugging).</p> <p>The response returns (in a formatted string) all information relevant to the specified process instance, including a snapshot of all process variables' values.</p> <p><b>Required system action:</b> queryProcessInstance</p>	String
<a href="#">getStarterOperationInfo</a>	<p>Lists the parameter details for a particular starter operation. Starter operations refer to start events of processes.</p> <p>To be able to directly start an instance of a process template, the start event of the process must have a trigger type of "None" (as opposed to a trigger type of "Message", which requires that the process be started by a business service).</p> <p><b>Required system action:</b> queryProcessTemplate</p>	<a href="#">OperationInfo</a>


Function	Description	Returns
<a href="#">ignoreProcessInstance</a>	<p>Resumes execution of a process instance that has halted as a result of an activity failure, ignoring the failed activity. Execution continues from the point in the process after the failed activity, as if the failed activity had not been processed.</p> <p>Also see the <a href="#">cancelProcessInstances</a>, <a href="#">resumeHaltedProcessInstance</a>, <a href="#">retryProcessInstance</a>, and <a href="#">queryHaltedProcessInstances</a> functions in this topic.</p> <p><b>Required system action:</b> haltedProcessAdministration</p>	String
<a href="#">ignoreProcessInstances</a>	<p>Ignores failed activities and resume the execution of <i>all</i> halted process instances for the specified process templates. For each process instance, execution continues from the point in the process after the failed activity, as if the failed activity had not been processed.</p> <p>Also see the <a href="#">cancelProcessInstance</a>, <a href="#">resumeHaltedProcessInstance</a>, <a href="#">retryProcessInstance</a>, and <a href="#">queryHaltedProcessInstances</a> functions in this topic.</p> <p><b>Required system action:</b> haltedProcessAdministration</p>	int
<a href="#">isSetMigrationRule</a>	<p>Tests whether any process instance migration rules are set for a particular qualified task name.</p> <p>The request must specify the qualified task name that is to be tested. (The qualified task name includes the task name and its parent process template, module and template/module version.)</p> <p>For information about process migration, see <a href="#">Migrating a Process Instance to a Different Version</a>.</p> <p><b>Required system action:</b> handleProcessMigration</p>	boolean
<a href="#">listAdhocActivitiesByGlobalRef</a>	<p>Lists all ad-hoc activities associated with the specified global data reference.</p> <p>The response lists all ad-hoc activities for all process instances that reference the global data.</p> <p><b>Required system action:</b> None</p>	<a href="#">AdhocActivityTypes</a>
<a href="#">listAdhocActivitiesById</a>	<p>Lists all ad-hoc activities associated with the specified process ID.</p> <p>The response lists all ad-hoc activities associated with the identified process instance.</p> <p><b>Required system action:</b> None</p>	<a href="#">AdhocActivityTypes</a>



Function	Description	Returns
<a href="#">listMigrationRules</a>	<p>Lists the process instance migration rules that are set for the specified process templates.</p> <p>For information about process migration, see <a href="#">Migrating a Process Instance to a Different Version</a>.</p> <p><b>Required system action:</b> <code>handleProcessMigration</code></p>	<a href="#">MigrationRuleTypesList</a>
<a href="#">listProcessInstanceAttributes</a>	<p>Lists process instance attributes for the specified process templates (these attributes can be used in process instance queries).</p> <p>Also see the <code>listSetofProcessInstanceAttributes</code> function (below).</p> <p><b>Required system action:</b> <code>queryProcessInstance</code></p>	<a href="#">InstanceAttributesType</a>
<a href="#">listProcessInstances</a>	<p>Lists the process instances for the specified process template.</p> <p>This is a convenience function: the <code>queryProcessInstances</code> function (see below) can return the same information, but requires more input.</p> <p>The <code>listProcessInstances</code> response can contain a maximum of 500 process instances. This limit is fixed and cannot be configured. If you are expecting a response that exceeds 500 process instances, use the <code>queryProcessInstances</code> function (see below) with a positive <b>pageSize</b> parameter instead.</p> <p><b>Required system action:</b> <code>queryProcessInstance</code></p>	<a href="#">ProcessInstances</a>
<a href="#">listProcessTemplateAttributes</a>	<p>List process template attributes. These attributes, which are pre-defined variables and are the same for all process templates, can be used in queries for process templates.</p> <p>Also see the <code>queryProcessTemplates</code> and <code>queryProcessTemplatesAlt</code> functions in this topic.</p> <p><b>Required system action:</b> <code>queryProcessTemplate</code></p>	<a href="#">TemplateAttributes</a>
<a href="#">listProcessTemplates</a>	<p>Lists process templates that match the input criteria.</p> <p>This is a convenience function: the <code>queryProcessTemplates</code> function (see below) can return the same information, but requires more input.</p> <p><b>Required system action:</b> <code>queryProcessTemplate</code></p>	<a href="#">BasicProcessTemplatesType</a>


Function	Description	Returns
<a href="#">listServices</a>	<p>Lists the processes that have been "published as a REST service" in TIBCO Business Studio.</p> <p>The processes/services returned by this function can be started using the <code>startProcessIncomingReceiveTask</code> function (see below).</p> <p><b>Required system action:</b> <code>queryProcessTemplate</code></p>	<a href="#">QueryBusinessServicesResponse</a>
<a href="#">listSetofProcessInstanceAttributes</a>	<p>Lists process instance attributes for a set of process templates (these attributes can be used in process instance queries).</p> <p>Also see the <code>listProcessInstanceAttributes</code> function (above).</p> <p><b>Required system action:</b> <code>queryProcessInstance</code></p>	<a href="#">InstanceAttributesType</a>
<a href="#">listStarterOperations</a>	<p>Lists available starter operations for the specified process template.</p> <p>Also see the <code>createProcessInstance</code> function (see above).</p> <p>Starter operations refer to start events of processes. To be able to directly start an instance of a process template, the start event of the process must have a trigger type of "None" (as opposed to a trigger type of "Message", which requires that the process be started by a business service).</p> <p><b>Required system action:</b> <code>queryProcessTemplate</code></p>	<a href="#">StarterOperationsType</a>
<a href="#">purgeTerminatedProcessInstances</a>	<p>Purges saved process instances with terminal states that correspond to one or more of the specified marked process templates. There can only be one outstanding purge job at one time.<sup>1</sup></p> <p><b>Required system action:</b> <code>bulkPurgeProcessInstances</code></p>	<a href="#">PurgeStatus</a>
<a href="#">queryApplications</a>	<p>Returns all process template(s) belonging to the specified application(s).</p> <p><b>Required system action:</b> <code>queryProcessTemplate</code></p>	<a href="#">QueryApplicationsResponseType</a>


<sup>1</sup> The `ProcessManagementService` contains a `purgeProcessInstances` function that is a private API -- it is for internal use only.

Function	Description	Returns
<a href="#">queryDone</a>	<p>Releases all resources associated with a particular paged result set. An application must call this operation whenever it has finished with a particular paged result set.</p> <p>The request must include a paging ID, which can be obtained using <code>queryProcessInstances</code>, <code>queryProcessInstancesAlt</code>, <code>queryProcessTemplates</code>, or <code>queryProcessTemplatesAlt</code> (see below).</p> <p><b>Required system action:</b> <code>queryProcessInstance</code></p>	String
<a href="#">queryFirstPage</a>	<p>Lists the first page of data from the result set of a particular query operation.</p> <p>The request must include a paging ID, which can be obtained using <code>queryProcessInstances</code>, <code>queryProcessInstancesAlt</code>, <code>queryProcessTemplates</code>, or <code>queryProcessTemplatesAlt</code> (see below).</p> <p><b>Required system action:</b> <code>queryProcessInstance</code></p>	<a href="#">Page</a>
<a href="#">queryHaltedProcessInstances</a>	<p>Lists process instances that have halted (as a result of an activity failure) that match certain criteria.</p> <div>  <p>This function only queries HALTED process instances. To list process instances that are in a state other than HALTED, use <code>queryProcessInstances</code> or <code>queryProcessInstancesAlt</code> instead (see below).</p> </div> <p>Also see the <code>cancelProcessInstance</code>, <code>ignoreProcessInstance</code>, <code>resumeHaltedProcessInstance</code>, <code>retryProcessInstance</code>, <code>getAvailableProcessInstanceVariables</code>, and <code>setAvailableProcessInstanceVariables</code> functions in this topic.</p> <p><b>Required system action:</b> <code>queryProcessInstance</code></p>	<a href="#">QueryProcessInstancesOutputType</a>
<a href="#">queryLastPage</a>	<p>Lists the last page of data from the result set of a particular query operation.</p> <p>The request must include a paging ID, which can be obtained using <code>queryProcessInstances</code>, <code>queryProcessInstancesAlt</code>, <code>queryProcessTemplates</code>, or <code>queryProcessTemplatesAlt</code> (see below).</p> <p><b>Required system action:</b> <code>queryProcessInstance</code></p>	<a href="#">Page</a>

Function	Description	Returns
<a href="#">queryNextPage</a>	<p>Lists the next page of data from the result set of a particular query operation.</p> <p>The request must include a paging ID, which can be obtained using <a href="#">queryProcessInstances</a>, <a href="#">queryProcessInstancesAlt</a>, <a href="#">queryProcessTemplates</a>, or <a href="#">queryProcessTemplatesAlt</a> (see below).</p> <p><b>Required system action:</b> <a href="#">queryProcessInstance</a></p>	<a href="#">Page</a>
<a href="#">queryPreviousPage</a>	<p>Lists the previous page of data from the result set of a particular query operation.</p> <p>The request must include a paging ID, which can be obtained using <a href="#">queryProcessInstances</a>, <a href="#">queryProcessInstancesAlt</a>, <a href="#">queryProcessTemplates</a>, or <a href="#">queryProcessTemplatesAlt</a> (see below).</p> <p><b>Required system action:</b> <a href="#">queryProcessInstance</a></p>	<a href="#">Page</a>
<a href="#">queryProcessInstanceCount</a>	<p>Counts the number of process instances that match certain criteria.</p> <p><b>Required system action:</b> <a href="#">queryProcessInstance</a></p>	int
<a href="#">queryProcessInstanceCountAlt</a>	<p>Counts the number of process instances that match certain criteria. You can specify the status (ACTIVE, TERMINATED, ACTIVE_AND_TERMINATED) of the process instances to count.</p> <p><b>Required system action:</b> <a href="#">queryProcessInstance</a></p>	int
<a href="#">queryProcessInstances</a>	<p>Lists the process instances that match specified criteria.</p> <p>The response returns information from process instances according to the specified criteria, either as a complete list or as the first page of a paged list (in which case a paging ID is also returned).</p> <p>The paging ID can be used with the <a href="#">queryFirstPage</a>, <a href="#">queryNextPage</a>, <a href="#">queryPreviousPage</a>, <a href="#">queryLastPage</a>, and <a href="#">queryDone</a> functions (see above).</p> <p><b>Required system action:</b> <a href="#">queryProcessInstance</a></p>	<a href="#">QueryProcessInstancesOutputType</a>

Function	Description	Returns
<a href="#">queryProcessInstancesAlt</a>	<p>Queries information about process instances that match specified criteria.</p> <p>This is a variation of <a href="#">queryProcessInstances</a> (see above), in which the query string is divided into its constituent parts.</p> <p>The response returns information from process instances according to the specified criteria, either as a complete list or as the first page of a paged list (in which case a paging ID is also returned).</p> <p>To list process instances that are in a HALTED state, use <a href="#">queryHaltedProcessInstances</a> instead (see above).</p> <p><b>Required system action:</b> <a href="#">queryProcessInstance</a></p>	<a href="#">QueryProcessInstancesOutputType</a>
<a href="#">queryProcessTemplateCount</a>	<p>Counts the number of process templates that match certain criteria.</p> <p><b>Required system action:</b> <a href="#">queryProcessTemplate</a></p>	int
<a href="#">queryProcessTemplates</a>	<p>Queries and returns information about process templates that match specified criteria.</p> <p>The response returns information from process templates according to the specified criteria, either as a complete list or as the first page of a paged list (in which case a paging ID is also returned).</p> <p>The paging ID can be used with the <a href="#">queryFirstPage</a>, <a href="#">queryNextPage</a>, <a href="#">queryPreviousPage</a>, <a href="#">queryLastPage</a>, and <a href="#">queryDone</a> functions (see above).</p> <p>Also see the <a href="#">queryProcessTemplatesAlt</a> function (below) for an alternative method of querying process templates.</p> <p><b>Required system action:</b> <a href="#">queryProcessTemplate</a></p>	<a href="#">QueryProcessTemplatesOutputType</a>
<a href="#">queryProcessTemplatesAlt</a>	<p>Lists process templates that match certain criteria.</p> <p>This is a variation of <a href="#">queryProcessTemplates</a> (see above) in which the query string is divided into its constituent parts.</p> <p>The response returns the process templates that match the specified criteria, either as a complete list or as the first page of a paged list (in which case a paging ID is also returned).</p> <p>The paging ID can be used with the <a href="#">queryFirstPage</a>, <a href="#">queryNextPage</a>, <a href="#">queryPreviousPage</a>, <a href="#">queryLastPage</a>, and <a href="#">queryDone</a> functions (see above).</p> <p><b>Required system action:</b> <a href="#">queryProcessTemplate</a></p>	<a href="#">QueryProcessTemplatesOutputType</a>

Function	Description	Returns
<a href="#">replaceUser</a>	Accepts a list of process IDs or a user name as input, and uses the Subject on this request to replace the Subject associated with each of the referenced process instances or user. If the specified process instance has sub-processes, those Subjects of the sub-processes are replaced as well.	<a href="#">GeneralStatusType</a>
<a href="#">resumeHaltedProcessInstance</a>	<p>Resumes execution of a process instance that has halted as a result of an activity failure. Execution continues as if the activity had failed.</p> <div>  <p>The process instance immediately enters a FAILED state. This operation effectively allows the erroring process instance to resume and then fail as if it had never halted. It can be used when investigation of the halted process instance indicates that nothing can be done to retrieve it.</p> </div> <p>Also see the <code>cancelProcessInstance</code>, <code>ignoreProcessInstance</code>, <code>retryProcessInstance</code>, and <code>queryHaltedProcessInstances</code> functions in this topic.</p> <p>This function can only be used on a process instance that is in a HALTED state. To resume a process instance that is in a SUSPENDED state, use the <code>resumeProcessInstance</code> function instead (see below).</p> <p><b>Required system action:</b> haltedProcessAdministration</p>	String

Function	Description	Returns
<a href="#">resumeHaltedProcessInstances</a>	<p>Resumes execution of all process instances for the specified process templates, where those instances have halted as a result of an activity failure. Execution of each process instance continues as if the activity had failed.</p> <div>  <p>Each process instance immediately enters a FAILED state. This operation effectively allows the erroring process instances to resume and then fail as if they had never halted. It can be used when investigation of the halted process instances indicates that nothing can be done to retrieve them.</p> </div> <p>Returns an integer indicating the number of process instances that were resumed.</p> <p>Also see the <code>cancelProcessInstances</code>, <code>ignoreProcessInstances</code>, <code>retryProcessInstances</code>, and <code>queryHaltedProcessInstances</code> functions in this topic.</p> <p>This operation only affects process instance that are in a HALTED state. To resume process instances that are in a SUSPENDED state, use the <code>resumeProcessInstances</code> function instead (see below).</p> <p><b>Required system action:</b> haltedProcessAdministration</p>	int
<a href="#">resumeProcessInstance</a>	<p>Resumes a previously suspended process instance.</p> <p>Also see the <code>suspendProcessInstance</code> function (below).</p> <p><b>Required system action:</b> <code>resumeProcessInstance</code></p>	String
<a href="#">resumeProcessInstances</a>	<p>Resumes all process instances for the specified process templates.</p> <p>Returns an integer indicating the number of process instances that were resumed.</p> <p>Also see the <code>suspendProcessInstances</code> function (below).</p> <p><b>Required system action:</b> <code>bulkResumeProcessInstances</code></p>	int
<a href="#">retryProcessInstance</a>	<p>Resumes execution of a process instance that has halted as a result of an activity failure. Execution continues by retrying the failed activity.</p> <p>Also see the <code>cancelProcessInstance</code>, <code>ignoreProcessInstance</code>, and <code>resumeProcessInstance</code> functions in this topic.</p> <p><b>Required system action:</b> haltedProcessAdministration</p>	String

Function	Description	Returns
<a href="#">retryProcessInstances</a>	<p>Resumes the execution of all halted process instances for the specified process templates, in each case retrying the activity that caused the process instance to fail. For each process instance, execution continues by retrying the failed activity.</p> <p>Returns an integer indicating the number of process instances that were resumed.</p> <p>Also see the <code>cancelProcessInstances</code>, <code>ignoreProcessInstances</code>, and <code>resumeHaltedProcessInstances</code> functions in this topic.</p> <p><b>Required system action:</b> <code>haltedProcessAdministration</code></p>	int
<a href="#">runAdhocActivity</a>	<p>Invokes the identified ad-hoc activity. This function is applicable only to manual ad-hoc activities, not automatic ad-hoc activities (automatic ad-hoc activities are automatically invoked when they become enabled).</p> <p>The ad-hoc activity must be enabled before it can be invoked. An ad-hoc activity is enabled when:</p> <ul style="list-style-type: none"> <li>• any <i>initializer activity</i> specified in the ad-hoc activity definition has completed, and</li> <li>• a <i>pre-condition</i> specified in the ad-hoc activity definition is true.</li> </ul> <p>To determine if an ad-hoc activity is enabled, use the <code>listAdhocActivities</code> function (see above), and view the <code>enabled</code> element in the response.</p> <p>If an attempt is made to invoke an ad-hoc activity that is already active, the invocation is discarded, and an audit record is created for the request.</p> <p><b>Required system action:</b> <code>StartAndCancelAdhocActivity</code></p>	<a href="#">GeneralStatusType</a>
<a href="#">saveTerminatedProcessInstances</a> --or-- <a href="#">saveTerminatedProcessInstancesAlt1</a> --or-- <a href="#">saveTerminatedProcessInstancesAlt2</a>	<p>Specifies which process templates to save. This means all process instances started from the process template will be saved in the database when their status is complete.</p> <p><b>Required system action:</b> <code>bulkPurgeProcessInstances</code></p>	<a href="#">PurgeStatus</a>



Function	Description	Returns
<a href="#">setAvailableProcessInstanceVariables</a>	<p>Sets the value of one or more variables that are available to the failed activity for a particular halted process instance.</p> <p>Also see the <a href="#">getAvailableProcessInstanceVariables</a> function (above).</p> <p><b>Required system action:</b> haltedProcessAdministration</p>	String
<a href="#">setDeadlineExpiration</a>	<p>Changes the deadline value for an activity.</p> <p>An active deadline must be in use (that is, it cannot be expired) on an activity (using an intermediate timer event) to be able to set the expiration time with this function.</p> <p><b>Required system action:</b> setDeadlineExpiration</p>	String
<a href="#">setMigrationRules</a>	<p>Sets one or more process instance migration rules.</p> <p>Each migration rule consists of a unique set of module name, module version from which to migrate, process template name, task name from which to migrate, and module version to which to migrate.</p> <p>For information about process migration, see <a href="#">Migrating a Process Instance to a Different Version</a>.</p> <p><b>Required system action:</b> handleProcessMigration</p>	String
<a href="#">setPriority</a>	<p>Changes the scheduling priority for execution of a process instance.</p> <p><b>Required system action:</b> setPriority</p>	String
<a href="#">startProcessIncomingReceiveTask</a>	<p>Starts an instance of a process template that was "published as a REST Service" in TIBCO Business Studio.</p> <p>For details about publishing a process as a REST service, see the "Exposing the Web Service Operation as a REST Service" section in the <i>TIBCO Business Studio - BPM Implementation</i> guide.</p> <p>You can also start instances of process templates using the <a href="#">createProcessInstance</a> function (see above), or by starting a business service (see <a href="#">startBusinessService</a>).</p> <p><b>Required system action:</b> startprocess</p>	<a href="#">StartBusinessServiceResponse</a>

Function	Description	Returns
<a href="#">stopPurgeTerminatedProcessInstances</a>	<p>Stops the purge job started with the <a href="#">purgeTerminatedProcessInstances</a> function (see above). Only effective when purge job is run in the background.</p> <p>Also see the <a href="#">purgeTerminatedProcessInstances</a> function (above).</p> <p><b>Required system action:</b> <a href="#">bulkPurgeProcessInstances</a></p>	<a href="#">PurgeStatus</a>
<a href="#">suspendProcessInstance</a>	<p>Suspends a process instance from being scheduled for execution.</p> <p>Also see the <a href="#">resumeProcessInstance</a> function (above).</p> <p><b>Required system action:</b> <a href="#">suspendProcessInstance</a></p>	String
<a href="#">suspendProcessInstances</a>	<p>Suspends all process instances for the specified process templates from being scheduled for execution.</p> <p>Returns an integer indicating the number of process instances that were suspended.</p> <p>Also see the <a href="#">resumeProcessInstances</a> function (above).</p> <p><b>Required system action:</b> <a href="#">bulkSuspendProcessInstances</a></p>	int
<a href="#">unsetMigrationRules</a>	<p>Unsets one or more currently set process instance migration rules.</p> <p>For information about process migration, see <a href="#">Migrating a Process Instance to a Different Version</a>.</p> <p><b>Required system action:</b> <a href="#">handleProcessMigration</a></p>	String

## ResourceQueryService

The [ResourceQueryService](#) contains functions to execute a Resource Query Language (RQL) query to find a set of resources that match specific criteria.

The table below lists the functions available from the [ResourceQueryService](#).

For information about RQL, see the "Resource Query Language" topic in the *TIBCO ActiveMatrix BPM Implementation* guide.

The [ResourceQueryService](#) functions do not require any system actions.

Function	Description	Returns
<a href="#">executeComplexQuery</a>	Executes a Resource Query Language (RQL) query to find a set of resources that match specific criteria.	<a href="#">ExecuteQueryResponseType</a>
<a href="#">executeQuery</a>	Executes a Resource Query Language (RQL) query to find a set of resources that match specific criteria.	<a href="#">ExecuteQueryResponseType</a>

Note that the only difference between these two functions is that `executeQuery` uses an HTML GET with the query string passed as a query parameter, whereas `executeComplexQuery` uses an HTML POST with the query string passed as a POST payload.

## RoleService

The `RoleService` contains functions related to roles, which are used to control access to applications, as well as access to specific functionality within an application.

The table below lists the functions available from the [RoleService](#).

All `RoleService` functions require the `viewRoles` system action.

Function	Description	Returns
<a href="#">getRoleModel</a>	Retrieves the role model for the specified role.	<a href="#">RoleModel</a>
<a href="#">getRoleSettings</a>	Retrieves the settings for a role for the specified application. You can also optionally request the role settings for a specific component in the application.	<a href="#">RoleSettings</a>
<a href="#">listRolesExt</a>	Lists the available roles. You can optionally provide a list of targets (groups or positions). If specified, roles mapped to the targets are returned. If omitted, all roles are returned. The list should be passed in the body as a JSON array.	List< <a href="#">Role</a> >



Role administration functions (creating, mapping, and so on) are not available from the `objectAPI`. For information about performing those types of functions, see the *TIBCO ActiveMatrix BPM Client Application Management Guide*.

## SecurityService

The `SecurityService` contains functions to manage user authorization.

The table below lists the functions available from the [SecurityService](#).

The `SecurityService` functions do not require any system actions.

Function	Description	Returns
<a href="#">getUserPrivileges</a>	Lists the privileges currently assigned to a specific user.	<a href="#">GetUserPrivilegesResponseType</a>
<a href="#">isActionAuthorized</a>	Tests whether the caller is authorized to perform one or more system actions, either globally or within the scope of a particular organization model entity.	<a href="#">IsActionAuthorisedResponseType</a>
<a href="#">listActionAuthorizedEntities</a>	Lists the organization model entities on which the caller is authorized to perform a specific system action.	<a href="#">ListActionAuthorisedEntitiesResponseType</a>

Function	Description	Returns
<a href="#">listAuthorisedOrgs</a>	<p>Lists the organizations on which the caller is authorized to perform a specific system action.</p> <p>In addition to testing the named system action, this function limits the organizations to those to which the calling user has access (because of organization relationships). For more information, see <a href="#">Organization Relationships</a>.</p>	<a href="#">ListAuthorisedOrgsResponse</a>

## StatisticsService

The StatisticsService contains a function to request a process template measure from the Event Collector database tables.

The table below lists the functions available from the [StatisticsService](#).

Function	Description	Returns
<a href="#">getMeasure</a>	<p>Requests measures for processes or work items from the Event Collector database tables.</p> <p><b>System action required:</b> None</p>	<a href="#">GetMeasureResponse</a>

## UserSettingsService

The UserSettingsService contains functions to manage user settings.

The table below lists the functions available from the [UserSettingsService](#).

All of the UserSettingsService functions require the userAdmin system action.

Function	Description	Returns
<a href="#">deleteUserSettings</a>	Deletes all user settings (name/value pairs) currently defined for a specific storageKey.	<a href="#">DeleteUserSettingsResponseType</a>
<a href="#">getUserSettings</a>	Lists the user settings (name/value pairs) currently defined for a specific storageKey and settingID.	<a href="#">GetUserSettingsResponseType</a>
<a href="#">listUserSettingIds</a>	Lists all user settingIDs currently defined for a specific storageKey.	<a href="#">ListUserSettingIdsResponseType</a>
<a href="#">saveUserSettings</a>	<p>Saves one or more user settings (name/value pairs) for a specific storageKey and settingID.</p> <p>User settings allow the application to store any name/value pairs on the server for later retrieval (using the <a href="#">getUserSettings</a> function (see above)).</p>	<a href="#">SaveUserSettingsResponseType</a>

## WorkCalService

The WorkCalService contains functions to create, delete, edit, and get information about base and overlay calendars and the time zones and calendar entries that they use.

The table below lists the functions available from the [WorkCalService](#).

For more information about calendars, see [Calendars](#).



All start and end times passed are taken to be time-zone-neutral: that is, the API assumes that all times are in UTC format.

Function	Description	Returns
<a href="#">copyCalendar</a>	<p>Copies an existing calendar to create a new calendar of the same type.</p> <p>The request must identify the existing calendar and include a valid name and namespace for the new calendar. Note that &lt;blank&gt; is a valid namespace. The name specified must be unique within the given namespace.</p> <p>For base calendars, the properties to be copied from the original calendar will include the time zone and working hours. For both base and overlay calendars, all future calendar entries can also be copied if requested.</p> <p><b>Required system action:</b> writeCalendars</p>	<a href="#">XmlUpdatedCalendar</a>
<a href="#">createCalendarEntry</a>	<p>Creates a new working-day-exclusion calendar entry. (Note that in this context, "entry" is synonymous with "exclusion.")</p> <p>Date and time values in the request are expected to be time-zone neutral.</p> <p>The request must include a version number that is compared against the value held by the identified calendar entry. If the values don't match, the request fails with a "concurrency exception".</p> <p><b>Required system action:</b> writeCalendars</p>	<a href="#">XmlCalendarEntries</a>
<a href="#">createCalendarReferences</a>	<p>Associates a calendar with a calendar reference that was created in TIBCO Business Studio. A reference can refer to only one calendar, but a calendar can be referenced by any number of client references.</p> <p>If the request includes a calendar reference that is already associated with another calendar, that association is removed before adding the reference to the calendar identified in the request.</p> <p>The response identifies (by namespace and name) those calendars affected by the request, that is, those whose calendar references have been modified. It also lists the updated version number of each calendar.</p> <p>The request must include a version number that is compared against the value held by the identified calendar entry. If the values don't match, the request will fail with a "concurrency exception".</p> <p><b>Required system action:</b> writeCalendars</p>	<a href="#">SaveCalendarReferencesResponseType</a>

Function	Description	Returns
<a href="#">createRecurringEntry</a>	<p>Creates a new recurring exclusion calendar Entry.</p> <p>The request must include a version number that is compared against the value held by the identified calendar entry. If the values don't match, the request will fail with a "concurrency exception".</p> <p><b>Required system action:</b> writeCalendars</p>	<a href="#">XmlCalendarEntries</a>
<a href="#">deleteCalendar</a>	<p>Deletes the named calendar, and all the working day and calendar entries associated with it. The deleted calendar cannot be recovered.</p> <p>The association of any calendar references to the deleted calendar will also be deleted; leaving those calendar references unfulfilled.</p> <p>In the request, pass the version value returned in the earlier call to retrieve the calendar (getCalendar function). If the given version value does not match the current value held in the database, a conflicting update must have occurred since the initial request to retrieve the calendar, and the delete request is rejected. The caller must call the getCalendar function (see below) again.</p> <p>Amendments to existing calendars will not be reflected in those deadline calculations that have already been performed.</p> <p><b>Required system action:</b> None</p>	Empty response (no JSON payload)
<a href="#">deleteCalendarEntries</a>	<p>Deletes the calendar entries identified by the GUIDs in the request. (Note that in this context, "entries" is synonymous with "exclusions.")</p> <p>The version value returned in the earlier request to retrieve the calendar exclusion (getCalendarEntries function -- see below) should be passed in this deleteCalendarEntries call. If the given version value does not match the current value held in the database, a conflicting update must have occurred since the initial request to retrieve the calendar exclusion, and the delete request is rejected. The caller must then call the getCalendarEntries function again.</p> <p>Amendments to existing calendar exclusions will not be reflected in those deadline calculations that have already been performed.</p> <p><b>Required system action:</b> deleteCalendars</p>	Empty response (no JSON payload)

Function	Description	Returns
<a href="#">deleteCalendarReferences</a>	<p>Deletes the specified calendar references from the identified calendar.</p> <p>The request must include a version number that is compared against the value held by the identified calendar entry. If the values don't match, the request fails with a "concurrency exception".</p>	<a href="#">SaveCalendarReferencesResponseType</a>
<a href="#">getCalendar</a>	<p>Lists the details of a named calendar.</p> <p>The response also includes a version value that is used for optimistic locking purposes. The same value should be passed in a request to save the calendar.</p> <p>For a base calendar, the time zone and working day entries are included in the response. No other calendar entries are included in the response.</p> <p><b>Required system action:</b> readCalendars</p>	<a href="#">XmlCalendarResponse</a>
<a href="#">getCalendarEntries</a>	<p>Lists the calendar exclusions for a given date range. (Note that in this context, "entries" is synonymous with "exclusions.")</p> <p>Returns both working-day exclusions and recurring exclusions.</p> <p>The date and time values of the exclusions are specified and displayed as time-zone neutral. For base calendars, the time-zone identifier is included in the response.</p> <p>In the case of recurring exclusions, those occurrences that fall within the given date range are listed in the response.</p> <p><b>Required system action:</b> readCalendars</p>	<a href="#">XmlCalendarEntries</a>
<a href="#">listCalendars</a>	<p>Lists all known base and overlay calendars.</p> <p><b>Required system action:</b> readCalendars</p>	<a href="#">ListCalendarsResponseType</a>
<a href="#">listTimeZones</a>	<p>Lists all the time zones recognized by the system.</p> <p>These time zones are the possible values for the time-zone property of a base calendar.</p> <p>The calling operation may cache the response, and use the entries to allow the user to select a time zone when creating or editing a base calendar.</p> <p><b>Required system action:</b> None</p>	<a href="#">ListTimeZonesResponseType</a>

Function	Description	Returns
<a href="#">purgeCalendarEntries</a>	<p>Deletes all the calendar exclusions with an end date earlier than the specified to-date, for the named calendar. (Note that in this context, "entries" is synonymous with "exclusions.")</p> <p>If the to-date exceeds the current date, the current date is used.</p> <p>The deleted exclusions cannot be recovered. If any of the exclusions fails to be deleted, no exclusions are deleted.</p> <p><b>Required system action:</b> deleteCalendars</p>	Empty response (no JSON payload)
<a href="#">renameCalendar</a>	<p>Changes a calendar's name, namespace, or both.</p> <p>The change will not succeed if another calendar already exists with the new name and namespace combination.</p> <p>Any references to the calendar will not be affected by the change.</p> <p><b>Required system action:</b> writeCalendars</p>	Empty response (no JSON payload)
<a href="#">resolveReferences</a>	<p>Resolves a collection of specified calendar reference GUIDs, by returning the identifier of the calendar instance with which each reference is associated.</p> <p>More than one calendar reference may be associated with the same calendar.</p> <p>The response contains the same list of calendar references as requested, plus the identifier of the calendar instance that each reference is associated with.</p> <p>If any reference is not associated with a calendar instance, it is still included in the response but no calendar is identified for that reference.</p> <p><b>Required system action:</b> readCalendars</p>	<a href="#">ResolveReferencesResponse</a>



Function	Description	Returns
<a href="#">saveBaseCalendar</a>	<p>Updates the specified base calendar if it already exists, or creates it if it does not exist.</p> <ul style="list-style-type: none"> <li>• The request includes the calendar time zone and the working day entries.</li> <li>• Date and time values in the request are stored as UTC. However it is best practice to specify a time zone of UTC for each time entered, to prevent any misinterpretation that might result from the different time zones of client and server. For more information on how time zones are handled, see the <i>Time-zone and time-slot for Base Calendars</i> section below.</li> </ul> <p>The request does not include calendar entries; they are created using the <code>createCalendarEntry</code> function (see above).</p> <p>If this function call is to update an existing base calendar, the call should include the version value returned in the earlier request to retrieve the calendar (see the <code>getCalendar</code> function).</p> <p>If the given version value does not match the current value held in the database, a conflicting update must have occurred since the initial request to retrieve the calendar, and the save request is rejected. You must then call <code>getCalendar</code> again to obtain the current version.</p> <p>If the version value shows no conflict, the base calendar is updated, and the response includes a new version value.</p> <p>Amendments to existing base calendars are not reflected in deadline calculations that have already been performed.</p> <p><b>Required system action:</b> <code>writeCalendars</code></p>	<a href="#">XmlCalendarResponse</a>

Function	Description	Returns
<a href="#">saveOverlayCalendar</a>	<p>Updates the specified overlay calendar if it already exists, or creates it if it does not exist.</p> <p>The request does not include calendar entries; they are created using the createCalendarEntry function (see above).</p> <p>If the request is one to update an existing overlay calendar, the request should include the version value returned in the earlier request to retrieve the calendar (getCalendar).</p> <p>If the given version value does not match the current value held in the database, a conflicting update must have occurred since the initial request to retrieve the calendar, and the save request will be rejected; The caller must then perform another getCalendar request, to obtain the current version.</p> <p>If the version value shows no conflict, the calendar is updated, and the response includes a new version value.</p> <p>Amendments to existing calendars are not reflected in deadline calculations that have already been performed.</p> <p><b>Required system action:</b> writeCalendars</p>	<a href="#">XmlCalendarResponse</a>
<a href="#">updateCalendarEntry</a>	<p>Updates an existing calendar exclusion. (Note that in this context, "entry" is synonymous with "exclusion.")</p> <p>If the request is to update an existing calendar exclusion, the version value returned in the earlier request to retrieve the calendar exclusion (getCalEntries) should be passed in this function call. If the given version value does not match the current value held in the database, a conflicting update must have occurred since the initial request to retrieve the exclusion, and the update request is rejected. You must then call getCalendar again to obtain the current version.</p> <p>Amendments to existing calendar exclusions is not reflected in deadline calculations that have already been performed.</p> <p><b>Required system action:</b> writeCalendars</p>	<a href="#">XmlCalendarEntries</a>

Function	Description	Returns
<a href="#">updateRecurringEntry</a>	<p>Updates an existing recurring calendar exclusion. (Note that in this context, "entry" is synonymous with "exclusion.")</p> <p>A recurring exclusion is defined using the recurrence rule expression, not as a series of individual dates and times. You cannot edit an individual occurrence of a recurring exclusion as a separate entity, so any amendments made to a recurring exclusion are applied to all occurrences of that exclusion.</p> <p>If the request is to update an existing calendar exclusion, the version value returned in the earlier request to retrieve the calendar exclusion (<a href="#">getCalEntries</a>) should be passed in this function call. If the given version value does not match the current value held in the database, a conflicting update must have occurred since the initial request to retrieve the exclusion, and the update request is rejected. You must then call <a href="#">getCalendar</a> again to obtain the current version.</p> <p>Amendments to existing recurring calendar exclusions is not reflected in deadline calculations that have already been performed.</p> <p><b>Required system action:</b> writeCalendars</p>	<a href="#">XmlCalendarEntries</a>

### Time-zone and time-slot for Base Calendars

The following provides information about setting time zones and time slots when saving a base calendar using the [saveBaseCalendar](#) function (see above).

- **time-zone** - Use one of the values returned by the [listTimeZones](#) function (see above). If no time zone is specified, the time zone of the BPM server is used.
- **time-slot start** and **end** - For each **day-of-week** specified in a base calendar, there can be up to five entries, each entry giving the start and end times of a working period within the specified day. Morning and afternoon separated by a meal break, for example, would be two time slots. Specify times as hh:mm.SSSTZ, where:

- SSS is the 3-digit millisecond value
- TZ is the time zone; for example, UTC is represented by 'Z'.

TIBCO strongly recommends that you specify a time zone of Z for all entries. If no time zone is specified, some APIs used by TIBCO ActiveMatrix BPM will automatically convert the value to the time zone used by the server, regardless of the calendar's time-zone specification. This may not always be the time zone required.

BPM stores **time-slot start** and **end** times as UTC, regardless of any time zone specified. However, the base calendar's time-zone value is used to adjust these times for display purposes in the client application.


### WorkItemManagementService


The [WorkItemManagementService](#) contains functions to perform actions on work items.

The table below lists the functions available from the [WorkItemManagementService](#).

For more information about work items, see [Work Lists and Work List Views](#).

Function	Description	Returns
<a href="#">allocateAndOpenNextWorkItem</a>	<p>Allocates the next available work item for a specified resource to that resource and immediately opens the work item (to get the associated input and output data).</p> <p>The function fails if there is no next item in the resource's work list.</p> <p>This function can only be used if the work item is in a state from which it can be allocated and opened. See <a href="#">Work Item State Transitions</a>.</p> <p><b>Required system action:</b> workItemAllocation</p>	<a href="#">AllocateAndOpenNextWorkItemResponseType</a>
<a href="#">allocateAndOpenWorkItem</a>	<p>Allocates a work item to a single resource and immediately opens the work item (to get the associated input and output data).</p> <p>This function can only be used if the work item is in a state from which it can be allocated and opened. See <a href="#">Work Item State Transitions</a>.</p> <p><b>Required system action:</b> workItemAllocation</p>	<a href="#">AllocateAndOpenWorkItemResponseType</a>
<a href="#">allocateWorkItem</a>	<p>Allocates a lists of work items to a resource.</p> <p>This function can only be used if the work item is in a state from which it can be allocated and opened. See <a href="#">Work Item State Transitions</a>.</p> <p><b>Required system action:</b> workItemAllocation</p>	<a href="#">AllocateWorkItemResponseType</a>

Function	Description	Returns
<a href="#">closeWorkItem</a>	<p>Closes a work item (and updates the associated input and output data).</p> <p>There is a <code>closeWorkItem</code> function in two services:</p> <ul style="list-style-type: none"> <li>  <b>WorkItemManagementService</b> (the one described here) - Use this one if the user task did not open a form when the work item was opened. This would be used only in special use-case client applications. </li> <li> <b>WorkPresentationService</b> - Use this one to close a form that was opened when the work item was opened with the <a href="#">openWorkItem</a> function in the <code>WorkPresentationService</code>. This is typically called in response to a user clicking the <b>Close</b> button on a work item form. See <a href="#">closeWorkItem</a>. </li> </ul> <p>This function:</p> <ul style="list-style-type: none"> <li>• Puts the work item into a Pending state.</li> <li>• Can only be used if the work item is in a state from which it can be closed. See <a href="#">Work Item State Transitions</a>.</li> <li>• Can only be used if the work item is assigned to the user with whose credentials the operation is being invoked.</li> </ul> <p><b>Required system action:</b> Requires <code>closeOtherResourcesItems</code> in order to close a work item currently allocated to another user, for example, when a manager is closing an item in a supervised work list. Otherwise, none.</p>	<a href="#">CloseWorkItemResponseType</a>

Function	Description	Returns
<a href="#">completeWorkItem</a>	<p>Completes a work item (and updates the associated input and output data).</p> <p>Note that there is a <code>completeWorkItem</code> function in two services:</p> <ul style="list-style-type: none"> <li>• <b>WorkItemManagementService</b> (the one described here) - Use this function if the user task was designed to not open a form nor start a pageflow. This would typically be used only in special use-case client applications -- see <code>completeWorkItem</code>.</li> <li>• <b>WorkPresentationService</b> - Use this one if a form was opened when the work item was opened with the <a href="#">openWorkItem</a> function in the <code>WorkPresentationService</code>. This is typically called in response to a user clicking the <b>Submit</b> button on a work item form. See <a href="#">completeWorkItem</a>.</li> </ul> <p><b>Required system action:</b> None</p>	<a href="#">CompleteWorkItemResponseType</a>
<a href="#">getOfferSet</a>	<p>Gets the offer set for a work item (the offer set is the set of organization model entities to whom a work item is to be offered).</p> <p>You can choose whether to return only the GUIDs of the organization model entities, or whether to return version and type information as well.</p> <p><b>Required system action:</b> None</p>	<a href="#">GetOfferSetResponseType</a>
<a href="#">getWorkItemHeader</a>	<p>Gets the header information for a specific work item.</p> <p><b>Required system action:</b> None</p>	<a href="#">GetWorkItemHeaderResponseType</a>
<a href="#">openWorkItem</a>	<p>Opens a work item (to get the associated input and output data).</p> <div>  <p>There is an <code>openWorkItem</code> function in two services:</p> <ul style="list-style-type: none"> <li>• <b>WorkItemManagementService</b> (the one described here) - Use this one if the user task was designed to not open a form nor start a pageflow. This would typically be used only in special use-case client applications.</li> <li>• <b>WorkPresentationService</b> - Use this one if the user task was designed to either open a form or start a pageflow, which is typical for most client applications (see <code>openWorkItem</code>).</li> </ul> <p><b>Required system action:</b> None</p> </div>	<a href="#">OpenWorkItemResponseType</a>

Function	Description	Returns
<a href="#">pendWorkItem</a>	<p>Pends the specified work item, which causes the work item to be hidden in the work item list until a specified date/time, or period of time has expired. The work item becomes visible again when the date/time occurs, or the period of time expires.</p> <p>A hiddenPeriod of 0 cancels a hiddenPeriod set by a previous <code>pendWorkItem</code> function call on the same work item. See <a href="#">Accessing Hidden Work Items</a>.</p> <p><b>Required system action:</b> <code>pendWorkItem</code></p>	<a href="#">PendWorkItemResponseType</a>
<a href="#">reallocateWorkItem</a>	<p>Reallocates a work item with no new data to a different resource.</p> <p>Users to whom a work item can be reallocated is restricted by the system action held by the caller - see below. Also, to reallocate a work item to the original offer set, the work item can have a state of Offered, Allocated, or Pended; to reallocate a work item to the "the world", the work item can have a state of Allocated or Pended.</p> <p>If there is new data to be written to the work item, use the <code>reallocateWorkItemData</code> function instead (see below).</p> <p><b>Required system action:</b></p> <ul style="list-style-type: none"> <li>• <code>reallocateToOfferSet</code> - This system action allows you to reallocate work items to only the users in the original offer set.</li> <li>• <code>reallocateWorkItemToWorld</code> - This system action allows you to reallocate work items to any user in the organization model.</li> </ul>	<a href="#">ReallocateWorkItemResponseType</a>

Function	Description	Returns
<a href="#">reallocateWorkItemData</a>	<p>Reallocates a work item to a different resource and updates the work item with new data.</p> <p>Users to whom a work item can be reallocated is restricted by the system action held by the caller - see below. Also, to reallocate a work item to the original offer set, the work item can have a state of Offered, Allocated, or Pended; to reallocate a work item to the "the world", the work item can have a state of Allocated or Pended.</p> <p>If the work item has no new data associated with it use the <a href="#">reallocateWorkItem</a> function instead (see above).</p> <p><b>Required system action:</b></p> <ul style="list-style-type: none"> <li>• <a href="#">reallocateToOfferSet</a> - This system action allows you to reallocate work items to only the users in the original offer set.</li> <li>• <a href="#">reallocateWorkItemToWorld</a> - This system action allows you to reallocate work items to any user in the organization model.</li> </ul>	<a href="#">ReallocateWorkItemDataResponseType</a>



Function	Description	Returns
<a href="#">rescheduleWorkitem</a>	<p>Reschedules the work item and/or changes the work item data.</p> <p>Depending on your requirements, the request can specify the:</p> <ul style="list-style-type: none"> <li>• <b>itemSchedule</b> - This is the work item schedule period to be associated with the work item. If no object is passed then the item schedule period will not be changed. The parameters are: <ul style="list-style-type: none"> <li>– <b>startDate</b> (optional): Earliest date at which the work item can be started.</li> </ul> <p>And a choice of one of the following:</p> <li>– <b>maxDuration</b> (optional): Duration of the work item</li> <li>– <b>targetDate</b> (optional): Date by which this work item must be finished</li> </li></ul> <li>• <b>itemBody</b> - This is the work item body containing the data changes. If no object is passed, the item body will not be changed. The body is the data as a name/value pair. The parameters are: <ul style="list-style-type: none"> <li>– <b>Parameter</b>: Details of the data fields associated with this work item.</li> <li>– <b>complexValue</b>: Value(s) of the complex object. Only one complex value or value will be specified. The value is of xs:anyType as it contains the entire complex object as XML.</li> <li>– <b>value</b>: Parameter value(s) - for example, 10.</li> </ul> </li> <p>This function can be used when the work item is in any state other than Completed.</p> <p><b>Required system action:</b> rescheduleWorkItem</p>	<a href="#">RescheduleWorkItemResponseType</a>
<a href="#">saveOpenWorkItem</a>	<p>Saves a work item (and updates the associated input and output data).</p> <p><b>Required system action:</b> None</p>	<a href="#">SaveOpenWorkItemResponseType</a>


Function	Description	Returns
<a href="#">setWorkItemPriority</a>	<p>Sets a work item priority, which can be used to indicate the work item's relative importance.</p> <p>You can change the priority of an individual work item or multiple work items. Depending on your requirements, you have a choice of one of the following:</p> <ul style="list-style-type: none"> <li>• <b>absPriority</b> - Set a specific numeric priority.</li> <li>• <b>offsetPriority</b> - Offset a work item priority by specific value. For example, an employee is away on holiday and you want to offset the priority on all their work items by 20.</li> </ul> <p><b>Required system action:</b></p> <ul style="list-style-type: none"> <li>• <b>changeAllocatedWorkItemPriority</b> - This system action allows you to reset the priorities of work items with a status of Allocated and that is allocated to the calling resource.</li> <li>• <b>changeAnyWorkItemPriority</b> - This system action allows you to change the work item priority of all work items.</li> </ul>	<a href="#">SetWorkItemPriorityResponseType</a>
<a href="#">skipWorkItem</a>	<p>Skips a work item. This means that no action is carried out on the work item.</p> <p>This function can only be used on work items that have default values for all required output and in/out data fields, or have values set already for such fields.</p> <p><b>Required system action:</b> skipWorkItem</p>	<a href="#">SkipWorkItemResponseType</a>
<a href="#">unallocateWorkItem</a>	<p>Reoffers a currently allocated work item to the original offer set.</p> <p>This function can only be used if the work item is in a state from which it can be unallocated. See <a href="#">Work Item State Transitions</a>.</p> <p><b>Required system action:</b> workItemAllocation</p>	<a href="#">UnallocateWorkItemResponseType</a>




## WorkListService

The WorkListService contains functions to get work item lists for organization model entities, and to get or to set sort/filter criteria for work item lists.

The table below lists the functions available from the [WorkListService](#).

For more information about work lists, see [Work Lists and Work List Views](#).

Function	Description	Returns
<a href="#">addCurrentResourceToView</a>	<p>Adds the current resource to a work view.</p> <p>The work view must be specified as public, either when it is created (using the <code>createWorkListView</code> function -- see below) or by editing it (using the <code>editWorkListView</code> function -- see below).</p> <p><b>Required system action:</b> <code>viewWorkList</code></p>	<a href="#">AddCurrentResourceToViewResponseType</a>
<a href="#">createWorkListView</a>	<p>Creates a new work view.</p> <p>The request must specify a name for the work view. It can optionally specify the details of the new work list view. For example, a description, any sort/filter criteria to be applied to this work view, the organizational entity whose work items you want to view or the resources that can access the work view.</p> <p>The response contains the unique ID (<b><code>workListViewID</code></b>) of the new work view.</p> <div>  <p>The owner of a view needs the <code>viewWorkList</code> system action for the target. Also, any GROUP, POSITION or ORG-UNIT that is added to the "Users" set also needs the <code>viewWorkList</code> system action.</p> </div> <p><b>Required system action:</b> <code>viewWorkList</code></p>	<a href="#">CreateWorkListViewResponseType</a>
<a href="#">deleteCurrentResourceFromView</a>	<p>Removes a public work view from the work view list for the calling resource.</p> <p><b>Required system action:</b> <code>viewWorkList</code></p>	<a href="#">DeleteCurrentResourceFromViewResponseType</a>
<a href="#">deleteWorkListView</a>	<p>Deletes an existing work list view. The calling resource must be the Owner or an Author of the work view to be able to delete it.</p> <p>The work view must be locked using the <code>getWorkListViewDetails</code> function (see below) before it can be deleted.</p> <p><b>Required system action:</b> <code>viewWorkList</code></p>	<a href="#">DeleteWorkListViewResponseType</a>

Function	Description	Returns
<a href="#">editWorkListView</a>	<p>Edits an existing work view.</p> <p>A work view must be locked using the <code>getWorkListViewDetails</code> function (using the <code>lockView</code> parameter) before you can edit the view with this function.</p> <p>The request must specify the name and the unique ID of the work view to be edited. It can also optionally specify the details of the work view that you want to edit. For example, its description, any sort/filter criteria or the organizational model entities that can access the work view. If any of the optional elements are specified, they overwrite the existing attributes specified for this view.</p> <div>  You only need to pass the data you want to change. Parameters that are not passed are unchanged.         </div> <div>  The owner of a view needs the <code>viewWorkList</code> system action for the target. Also, any <code>GROUP</code>, <code>POSITION</code> or <code>ORG-UNIT</code> that is added to the "Users" set also needs the <code>viewWorkList</code> system action.         </div> <p><b>Required system action:</b> <code>viewWorkList</code></p>	<a href="#">EditWorkListViewResponseType</a>
<a href="#">getAllocatedWorkListItems</a>	<p>Gets an allocated work list for an organization model entity.</p> <p><b>Required system action:</b> <code>viewWorkList</code></p>	<a href="#">GetWorkListItemsResponseType</a>
<a href="#">getEditableWorkListViews</a>	<p>Retrieves a list of work views that the calling resource can edit.</p> <p><b>Required system action:</b> <code>viewWorkList</code></p>	<a href="#">GetEditableWorkListViewsResponseType</a>
<a href="#">getPublicWorkListViews</a>	<p>Retrieves a list of all public work views.</p> <p>Work views are specified as public, either when they are created (using the <code>createWorkListView</code> function -- see above) or edited (using the <code>editWorkListView</code> function -- see above).</p> <div>  Sets of authors and users are not returned.         </div> <p><b>Required system action:</b> <code>viewWorkList</code></p>	<a href="#">GetPublicWorkListViewsResponseType</a>
<a href="#">getResourceOrderFilterCriteria</a>	<p>Gets previously defined sort/filter criteria for work item lists for a resource.</p> <p><b>Required system action:</b> <code>setResourceOrderFilterCriteria</code></p>	<a href="#">GetResourceOrderFilterCriteriaResponseType</a>

Function	Description	Returns
<a href="#">getViewsForResource</a>	<p>Retrieves a list of work views to which the calling resource has access. In other words, a list of work views to which the calling resource is either an Owner or an Author.</p> <div>  <p>Sets of authors and users are not returned.</p> </div> <p><b>Required system action:</b> viewWorkList</p>	<a href="#">GetViewsForResourceResponseType</a>
<a href="#">getWorkItemOrderFilter</a>	<p>Gets the specified fields defined by BRM that can be used to define sort/filter criteria for a work item list.</p> <p><b>Required system action:</b> None</p>	<a href="#">GetWorkItemOrderFilterResponseType</a>
<a href="#">getWorkItemOrderFilterNoLimit</a>	<p>Gets all fields that can be used to define sort/filter criteria for a work item list.</p> <p><b>Required system action:</b> None</p>	<a href="#">GetWorkItemOrderFilterResponseType</a>
<a href="#">getWorkListItems</a>	<p>Gets a work list for an organization model entity. Also see the <a href="#">getWorkListItemsAllResources</a> function (below).</p> <p><b>Required system action:</b> viewWorkList</p>	<a href="#">GetWorkListItemsResponseType</a>
<a href="#">getWorkListItemsAllResources</a>	<p>Gets a list of work items for all resources. This is similar to the <a href="#">getWorkListItems</a> function above, except it returns all work items, not just those for a specific entity.</p> <p><b>Required system action:</b> viewGlobalWorkList</p>	<a href="#">GetWorkListItemsResponseType</a>
<a href="#">getWorkListItemsForGlobalData</a>	<p>Gets a list of work items associated with a global data reference.</p> <p><b>Required system action:</b> viewGlobalWorkList</p>	<a href="#">GetWorkListItemsForGlobalDataResponseType</a>
<a href="#">getWorkListItemsForView</a>	<p>Retrieves the work item list for a specific work view.</p> <p>This must be executed as a resource who has access to the work view. See <a href="#">Work List View Access</a>.</p> <p><b>Required system action:</b> viewWorkList</p> <p>This requires the viewGlobalWorkList system action if you are retrieving work items for all resources.</p>	<a href="#">GetWorkListItemsForViewResponseType</a>

Function	Description	Returns
<a href="#">getWorkListViewDetails</a>	Retrieves the details of a specific work view. For example, its description, creation date, whether or not it is public and its owner. It can also be used to optionally lock a work view for editing.  If you want to edit a work view using the <a href="#">editWorkListView</a> function (see above), you must lock it first using this function.  <b>Required system action:</b> viewWorkList	<a href="#">WorkListView</a>
<a href="#">previewWorkItemFromList</a>	Previews one or more work items (to get the associated input and output data without opening the work items).  <b>Required system action:</b> None	<a href="#">PreviewWorkItemFromListResponseType</a>
<a href="#">setResourceOrderFilterCriteria</a>	Sets sort/filter criteria for work item lists for a resource.  <b>Required system action:</b> setResourceOrderFilterCriteria	<a href="#">SetResourceOrderFilterCriteriaResponseType</a>
<a href="#">unlockWorkListView</a>	Unlocks a locked work list view.  Use this to unlock a view that has been locked using the <a href="#">getWorkListViewDetails</a> function (see above).  <b>Required system action:</b> viewWorkList	<a href="#">UnlockWorkListViewResponseType</a>

## WorkPresentationService


The WorkPresentationService contains functions to perform various operations on work items.

The table below lists the functions available from the [WorkPresentationService](#).

The WorkPresentationService functions do not require a system action.

Function	Description	Returns
<a href="#">cancelWorkItem</a>	Cancels the specified work item. Any changes made to the data will be lost when you cancel a work item.	<a href="#">WorkResponseType</a>

Function	Description	Returns
<a href="#">closeWorkItem</a>	<p>Closes a work item form and updates the associated data with any changes the user has made while it was open.</p> <p>Note that there is a closeWorkItem function in two services:</p> <ul style="list-style-type: none"> <li>• <b>WorkPresentationService</b> (the one described here) - Use the function in this service to close a form that was opened when the work item was opened with the openWorkItem function in the WorkPresentationService. This is typically called in response to a user clicking the <b>Close</b> button on a work item form.</li> <li>• <b>WorkItemManagementService</b> - Use this function if the user task was designed to not open a form nor start a pageflow. This would typically be used only in special use-case client applications -- see <a href="#">closeWorkItem</a>.</li> </ul> <p>This function puts the work item into a Pended state.</p>	<a href="#">WorkResponseType</a>

Function	Description	Returns
<a href="#">completeWorkItem</a>	<p>Use this function when work on a work item or pageflow has completed. Input and output data associated with the work item is saved to the database. The work item must have a current state of Opened. Its state is changed to Completed.</p> <p>Note that there is a <code>completeWorkItem</code> function in two services:</p> <ul style="list-style-type: none"> <li>• <b>WorkPresentationService</b> (the one described here) - Use this one if a form was opened when the work item was opened with the <code>openWorkItem</code> function in the <code>WorkPresentationService</code>. This is typically called in response to a user clicking the <b>Submit</b> button on a work item form.</li> <li>• <b>WorkItemManagementService</b> - Use this function if the user task was designed to not open a form nor start a pageflow. This would typically be used only in special use-case client applications -- see <a href="#">completeWorkItem</a>.</li> </ul> <p>The request must specify the work item to be completed and also provide the payload.</p> <p>The response returns the execution status, the payload received in the request message, the work type details, presentation details and pageflow details (if any). If the work item is part of a chained group, the response returns the next work item in the group. If the work item is piled, the response returns the next piled work item.</p> <div>  <p>If the work item is part of a chained group and is also piled, the chained group takes precedence. Once all the chained work items are executed, the next piled work item is executed.</p> </div> <p>This function puts the work item into a Completed state. Once completed, no further operations can be performed on the work item.</p>	<a href="#">WorkResponseType</a>
<a href="#">openNextWorkItem</a>	Opens the next work item in the work queue.	<a href="#">WorkResponseType</a>
<a href="#">openWorkItem</a>	<p>Retrieves the associated input and output data and opens the work item form.</p> <p>There is an <code>openWorkItem</code> function in two services:</p> <ul style="list-style-type: none"> <li>• <b>WorkPresentationService</b> (the one described here) - Use this function if the user task was designed to either open a form or start a pageflow, which is typical for most client applications.</li> <li>• <b>WorkItemManagementService</b> - Use this one if the user task was designed to not open a form nor start a pageflow. This would typically be used only in special use-case client applications -- see <a href="#">openWorkItem</a>.</li> </ul>	<a href="#">WorkResponseType</a>



## Using the objectAPI in BPM Applications

BPM applications consist of some common elements, such as, organization models, processes, work items, and so on. The following subordinate topics describe using the objectAPI to interact with these elements.

[Organization Models and LDAP](#)

[Processes](#)

[Business Services](#)

[Pageflows](#)

[Work Lists and Work List Views](#)

[Work Items](#)

[Case Data Models](#)

[Forms](#)

[Events](#)

[Calendars](#)

### Organization Models and LDAP

The objectAPI provides functions to perform various tasks involving organization models and LDAP containers.

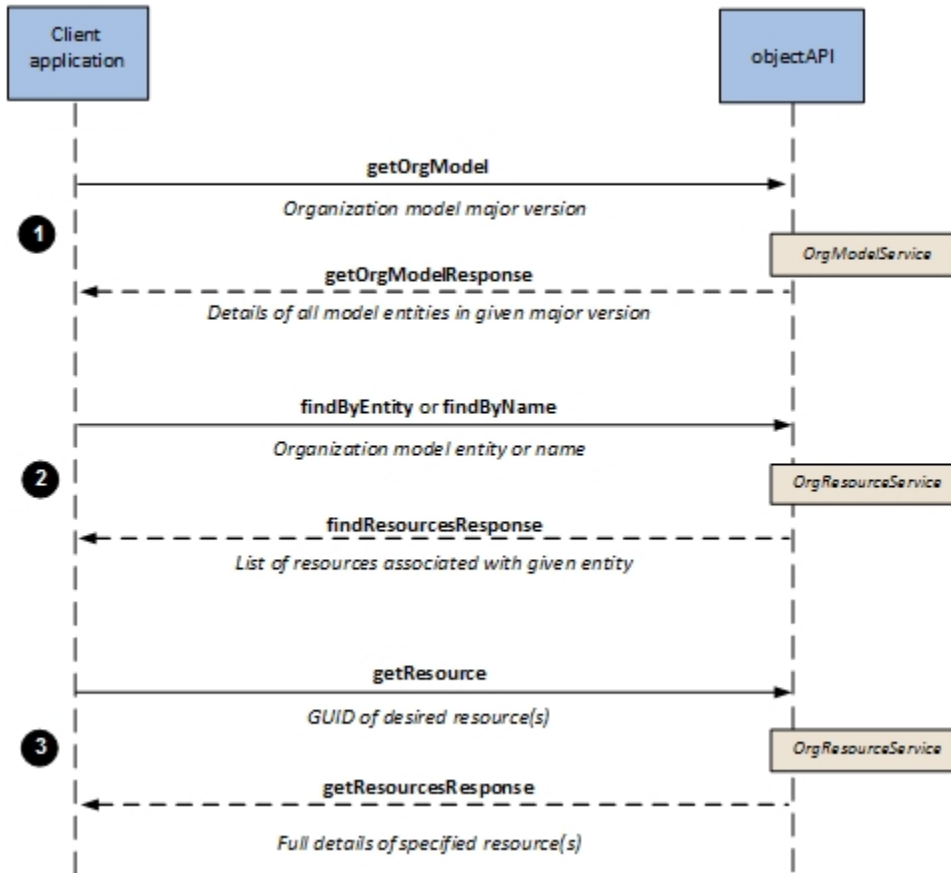
An organization model defines the organizational structure of an enterprise. The [OrgModelService](#), [OrgResourceService](#), and [DirectoryService](#) contains functions that allow you to perform operations such as browse the organization model, create LDAP containers, map users to groups or positions in the organization model, and so on.

### Navigating the Organization Model

The object API provides functions to navigate the organization model to locate a particular organizational entity for work allocation.

The following diagram shows an example of how calls to the [DirectoryService](#) functions can be used to navigate the organization model.

## Navigating the Organization Model



### Procedure

1. Call [getOrgModel](#) to get details about the entities in an organization model identified by its major version number (you can obtain the major version numbers using the [listOrgModelVersions](#) function).

The response from the [getOrgModel](#) function provides GUIDs for all entities in the organization model.

2. Call [findByEntity](#) or [findByName](#) to search for resources.

The response identifies the resources found in the search.

3. Call [getResource](#) to get resource details, including all current attribute values.

## Navigating Large Organization Models

Functions are provided that allow you to "drill down" into an organization model structure, rather than retrieving the entire organization model.

The following functions, which are provided in the [OrgModelService](#), allow for improved performance over using the [getOrgModel](#) function when retrieving a very large organization model:

- [getOrgModelRoots](#) - Requests the details of the entities that form the root elements of the organization model identified by the given major version. These include organizations, groups, locations, capabilities, privileges, resource attributes, and model templates. Using these root entities, it is possible for the client to navigate the entire organization model in a "drill-down" fashion.
- [getOrgModelEntity](#) - Requests the details of the entities identified by the given major version and their GUIDs. The GUIDs can be obtained using the [getOrgModelRoots](#) function, then the details can be retrieved with this function.

If the identified entity has associations with other entities (for example, positions associated with an organization unit), the GUIDs of those entities are included in the response. This allows details of those entities to be retrieved.

For a list of the properties that are returned for each organizational entity type, as well as definitions for these properties, see [Organization Model Entity Properties](#).

## Organization Model Entity Properties

This topic provides lists of the properties that are returned by operations that retrieve organization model entities, as well as definitions of the properties.

See also: [getOrgModel](#), [getOrgModelRoots](#), and [getOrgModelEntity](#).

The following lists the properties that are returned for each type of organizational entity:

- Organization
  - GUID
  - name
  - label
  - calendar alias
  - schema type identity
  - allocation method
  - location identity
  - org-unit identities
  - attributes
  - push destinations
- Organization Unit
  - GUID
  - name
  - label
  - calendar alias
  - dynamic indicator
  - schema type identity
  - allocation method
  - location identity
  - privilege holdings
  - attributes
  - push destinations
  - extension point configuration
  - model template instance identity values
  - position identities
  - sub-unit identities
- Position

- GUID
- name
- label
- ideal number
- resource count
- calendar alias
- dynamic indicator
- schema type identity
- allocation method
- location identity
- required capabilities
- privilege holdings
- attributes
- push destinations
- candidate query configuration
- Group
  - GUID
  - name
  - label
  - resource count
  - calendar alias
  - allocation method
  - required capabilities
  - privilege holdings
  - candidate query configuration
  - sub-group identities
- Location
  - GUID
  - name
  - label
  - schema type identity
  - attributes
- Capability
  - GUID
  - name
  - label
  - resource count
  - qualifier data type

- qualifier enumeration set
- Privilege
  - GUID
  - name
  - label
  - qualifier data type
  - qualifier enumeration set
- Resource Attribute
  - GUID
  - name
  - label
  - data type
  - enumeration value(s)
- Organization Unit - Dynamic Organization Model Templates
  - GUID
  - name
  - label
  - calendar alias
  - dynamic indicator
  - schema type identity
  - allocation method
  - location identity
  - privilege holdings
  - attributes
  - push destinations
  - model position identities
  - model sub-unit identities
- Position - Dynamic Organization Model Templates
  - GUID
  - name
  - label
  - ideal number
  - dynamic indicator
  - schema type identity
  - allocation method
  - location identity
  - required capabilities

- privilege holdings
- attributes
- push destinations
- candidate query configuration

The following are definitions of the organization model entity properties that are returned when navigating the organization model structure:

#### **allocation method**

Consists of two properties; the method of allocation and an, optional, "plugin" name. The method is one of: ANY, NEXT, THIS or PLUGIN. The plugin name property is for future use only, and would only appear if the method is PLUGIN.

#### **attributes**

Organization model entities are assigned attributes via the model schema from which they take their "type". This model schema association is optional; but, when included, the associated schema element of an organization model entity may define attributes that the organization model entity may carry. The attribute elements in a request describe the attribute definition (its GUID, name, label, and data type), and the value(s) assigned to that attribute.

#### **calendar alias**

Calendars can be assigned to groups, organizations, organization-units, positions, model organization-units, and model positions. These assignments are made using an alias; a design-time proxy to which a calendar definition can be associated at run-time.

The fact that an organization model entity holds a calendar alias does not mean that the calendar alias is associated with a calendar definition.

#### **candidate query configuration**

Candidate queries can be assigned to groups, positions, and model positions. When assigned to these entities, this property will describe its configuration.

#### **data type / qualifier data type**

For organization model attributes (including capability and privilege qualifiers) and resource attributes, this property names the type of data that the attribute may hold. Possible values and their interpretations, are:

- String - any UTF-8 encoded string value.
- Decimal - a floating point, decimal value.
- Integer - a numeric value.
- Boolean - a true/false, yes/no. 0/1 value.
- DateTime - a date/time value in ISO 8601 format.
- Date - a date value in ISO 8601 format.
- Time - a time value in ISO 8601 format.
- Enum - one of a pre-defined, finite list of values.
- EnumSet - a sub-set of a pre-defined, finite list of values.

#### **dynamic indicator**

A boolean indication of whether an organization-unit or position is derived from a dynamic organization model Template. If "false", or not specified, the entity is not derived from a dynamic organization model template.

#### **enumeration set / qualifier enumeration set**

For an attribute definition of data type "EnumSet" (including capability and privilege qualifiers), this lists the finite value from which an attribute of that definition can take its value. For an attribute value, this lists the sub-set of values taken from the attribute's definition: the attribute definition must be of data type "EnumSet".

#### **extension point configuration**

For those organization-unit entities identified as extension points, this shows the configuration of that extension point, including any dynamic instance ID attribute mappings. The extension point configuration may not be complete; in which case the configuration will carry the property complete="false".

#### **guid**

The unique identifier for any organization model entity. Although unique to a given entity, that entity may appear in multiple major versions; therefore, the major version may often be required in order to avoid any ambiguity.

#### **ideal number**

Positions and model positions can carry an integer value to indicate the number of resources that should, ideally, populate that position. This is purely for information and is not enforced, that is, fewer or more resources can be assigned.

#### **label**

Each organization model entity carries an optional label value. This is generally a non-normalized form of the name property entered in TIBCO Business Studio, but it can be used as a description of the organization model entity.

#### **location identity**

Organizations, organization-Units, positions, model organization-Units and model positions may be associated with a location. If so, this property gives the GUID, name, and label of that location entity. The location entity must be located within the same major version of the organization model as the entity referring to it.

#### **model position identities**

Model organization-units may have any number of model positions within them. This property array gives the GUID, name, and label of each model position entity. Each model position entity must be located within the same major version of the organization model as the model organization-unit.

#### **model sub-unit identities**

Model organization-units may have any number of model organization-units within them (sub-units). This property array gives the GUID, name, and label of each model organization-unit entity. Each model organization-unit entity must be located within the same major version of the organization model as the model organization-unit.

#### **model template instance identity values**

The root organization-unit of a dynamic organization model instance (an organization model fragment generated from a dynamic organization model template) will carry the dynamic instance ID attribute values that distinguish that instance from all other instances of the same dynamic organization model template. These attributes consist of the attribute name (as configured in the

dynamic organization model template) and the instance's value (taken from the LDAP attribute associated with the attribute in the extension point's configuration).

#### **name**

Each organization model entity carries a name. The name need not be unique.

#### **org-unit identities**

Organizations may have any number of organization-units located directly within them. This property array gives the GUID, name, and label of each organization-unit entity. Each organization-unit entity must be located within the same major version of the organization model as the organizations.

#### **position identities**

Organization-units may have any number of positions within them. This property array will give the GUID, name, and label of each position entity. Each position entity must be located within the same major version of the organization model as the organization-unit.

#### **privilege holdings**

Organization-units, positions, and groups may hold any number of privileges, which are inherited by those resources directly within them (for groups, those resources within sub-groups inherit the privileges of parent groups). This property array gives the GUID, name, and label of each privilege held. Each privilege entity must be located within the same major version of the organization model as the entity referring to them.

For those privilege holdings that refer to a qualified privilege, the holding will also list any qualifier value that applies to the holding.

#### **push destinations**

Push destinations allow resources to be notified of offered and allocated work items using mechanisms such as e-mail or SMS. Push destinations can be configured against organizations, organization-units, positions, model organization-units, and model positions.

#### **required capabilities**

Groups, positions and model positions may hold any number of required capabilities. Although not enforced; these are the capabilities a resource should hold in order to be a member of that group or position. This property array gives the GUID, name, and label of each required capability. Each capability entity must be located within the same major version of the organization model as the entity referring to them.

For those required capabilities that refer to a qualified capability, the holding will also list any qualifier value that applies to the holding.

#### **resource count**

For those organization model entities to which resources can be directly assigned (groups, positions and capabilities), the number of resources assigned to those entities can be, optionally, included in the response.

#### **schema type identity**

An organization model can include a model schema; a meta-model that gives type information to entities within the organization model itself. In particular, the model schema entities provide attributes that can be assigned to organization model entities that reference those entities.

Organizations, organization-units, positions and locations can carry a reference to a model schema entity. If so, this property gives the GUID, name, and label of the model schema entity. The model



schema entity must be located within the same major version of the organization model as the entity referring to them.

### sub-group identities

Groups may have any number of groups within them (sub-groups). This property array gives the GUID, name, and label of each group entity. Each group entity must be located within the same major version of the organization model as the group referring to it.

### sub-unit identities

Organization-units may have any number of organization-units within them (sub-units). This property array gives the GUID, name, and label of each organization-unit entity. Each organization-unit entity must be located within the same major version of the organization model as the organization-unit.

## Creating an LDAP Container

Before resources can be mapped to groups or positions in an organization model, you must create an LDAP container from which the resources can be selected.

An LDAP container is a collection of one or more LDAP<sup>2</sup> sources.

An LDAP source represents a connection to an LDAP server, which holds information about potential (or "candidate") resources (users) who may need to use or participate in TIBCO applications.

An LDAP container consists of a *primary* LDAP source, and zero or more *secondary* LDAP sources. When creating an LDAP container and specifying its LDAP sources, you use one of the following source types to locate resources:

- **LDAP Query Source** - An LDAP query is used to identify the directory entries that will be candidate resources.  
For an example of creating an LDAP container that uses an LDAP Query Source to locate candidate resources, see [Creating an LDAP Container Using an LDAP Query Source](#).
- **LDAP Group Source** - A Group DN is used to identify the directory entry that is the group. When a Group DN is specified, a *member attribute* is also specified, which holds the collection of member identifiers, that is, their DNs. This provides the list of candidate resources.

For an example of creating an LDAP container that uses an LDAP Group Source to locate candidate resources, see [Creating an LDAP Container Using an LDAP Group Source](#).

For more information about primary and secondary LDAP sources, see the *TIBCO ActiveMatrix® BPM Organization Browser User's Guide*.

## Creating an LDAP Container Using an LDAP Query Source

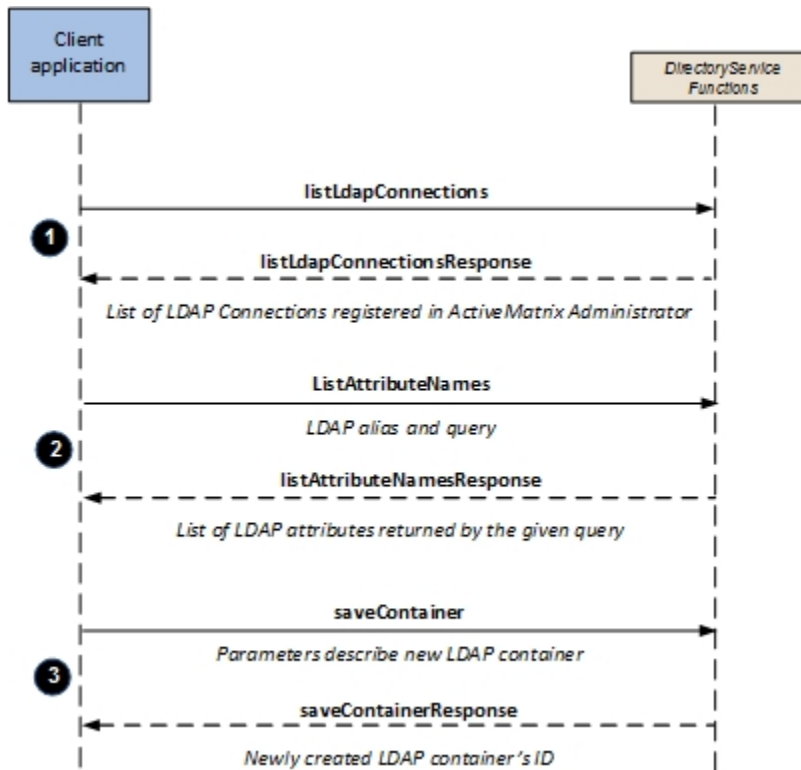
An LDAP query is used to identify the directory entries that will be candidate resources.

The following diagram shows an example of how calls to the [DirectoryService](#) functions can be used to create an LDAP container using an LDAP Query Source.

---

<sup>2</sup> Lightweight Directory Access Protocol, which is an application protocol for querying and modifying directory services.

### Creating an LDAP Container Using an LDAP Query Source



#### Procedure

1. Find out the LDAP Connection Shared Resources that are configured in ActiveMatrix Administrator by calling `listLdapConnections`.  
Only those whose instance name is prefixed with "ldap/de/" are listed.  
One of the returned sources must be specified as the primary LDAP source when calling `saveContainer` to create the LDAP container.
2. Call `listAttributeNames` to get LDAP attributes that can be used in the required LDAP query when calling `saveContainer` in the next step.
3. Call `saveContainer` to create and save a new LDAP container.

The required parameters are:

- **name** - This is the name you want assigned to the new LDAP container. It must be unique on the Directory Engine.
- **primary-ldap.ldap-alias** - You must specify a primary LDAP source, and one that identifies a valid LDAP connection. A list of the available LDAP sources were returned from `listLdapConnections` in Step 1.
- **primary-ldap.ldap-query** - Each LDAP source, whether primary or secondary, must specify a valid LDAP query. LDAP queries can be validated against a named LDAP connection using the `executeLdapQuery` function. For information about LDAP queries, see [LDAP Queries](#).

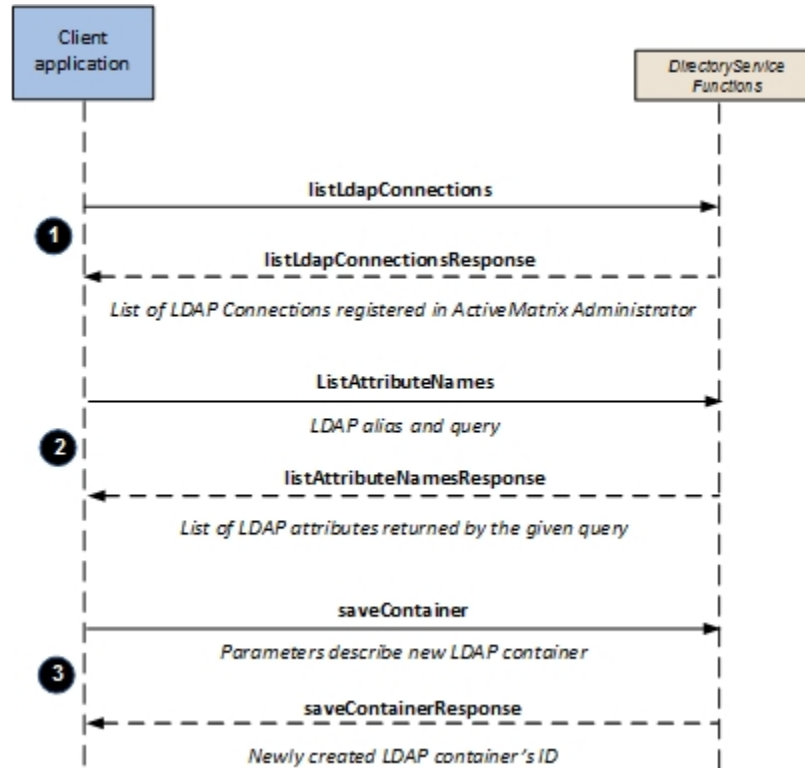
Additional optional parameters are also available for adding secondary LDAP sources, setting up organizational relationships, and so on. For information about those parameters, see `saveContainer`.

## Creating an LDAP Container Using an LDAP Group Source

A Group DN is used to identify the directory entry that is the group. When a Group DN is specified, a *member attribute* is also specified, which holds the collection of member identifiers, that is, their DN's. This provides the list of candidate resources.

The following diagram shows an example of how calls to the [DirectoryService](#) functions can be used to create an LDAP container using an LDAP Group Source.

### Creating an LDAP Container Using an LDAP Group Source



### Procedure

1. Find out the LDAP Connection Shared Resources that are configured in ActiveMatrix Administrator by calling [listLdapConnections](#).

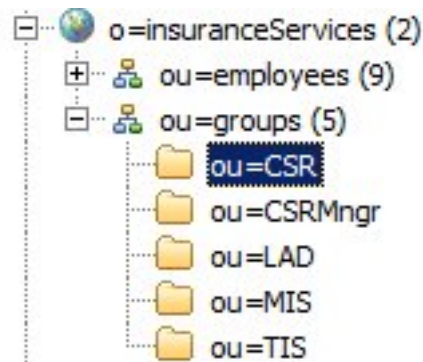
Only those whose instance name is prefixed with "ldap/de/" are listed.

One of the returned sources must be specified as the primary LDAP source when calling [saveContainer](#) to create the LDAP container.

2. Call [listAttributeNames](#) to get the groups and member attribute that can be used when calling [saveContainer](#) in the next step.
3. Call [saveContainer](#) to create and save a new LDAP container.

The required parameters are:

- **name** - This is the name you want assigned to the new LDAP container. It must be unique on the Directory Engine.
- **primary-ldap.ldap-alias** - You must specify a primary LDAP source, and one that identifies a valid LDAP connection. A list of the available LDAP sources were returned from [listLdapConnections](#) in step 1.
- **group-dn** - The LDAP directory entry that is the group. If the following LDAP directory identifies the group, the group-DN is "ou=CSR,ou=groups,o=insuranceServices":



- **member-attribute** - Identifies the attribute within the group entry that holds the collection of DNs that identifies the candidate resources. In the following example, the `roleoccupant` attribute identifies candidate resources:

DN: ou=CSR,ou=groups,o=insuranceServices	
Attribute Description	Value
<b>objectClass</b>	<b>organizationalRole (structural)</b>
<b>objectClass</b>	<b>top (abstract)</b>
<b>cn</b>	<b>Customer Services Representative</b>
<b>ou</b>	CSR
<b>roleoccupant</b>	uid=jparkin,ou=Employees,o=insuranceServices
<b>roleoccupant</b>	uid=rcrewell,ou=Employees,o=insuranceServices

- **resource-name-attributes** - The attribute(s) whose value(s) the resource will use to log into the system.

Additional optional parameters are also available for adding secondary LDAP group sources, setting up organizational relationships, and so on. For information about those parameters, see [saveContainer](#).

## Defining Secondary LDAP Sources in an LDAP Container

Each LDAP container that you define must include one *primary* LDAP source. It can also include one or more *secondary* LDAP sources.

If there are secondary LDAP sources defined, they will be used to find additional information about each potential resource from the primary LDAP source. Lookups are performed into each secondary LDAP source. If an exact match of a potential resource can be found in every secondary LDAP source, the data from all sources is merged together. In other cases, the potential resource may be omitted or labeled invalid. It determines that based on attribute relationships you specify when adding a secondary LDAP source to your container.

The following are reasons you might want to define a secondary LDAP source:

- The business process needs to access attribute data that is in both the primary and secondary LDAP sources.
- the business process needs to access attribute data from an LDAP source that is not used for login authentication (the primary LDAP source is always used for authentication).

As an example, suppose you have two LDAP sources: Acme-Employees and Acme-Developers. The Acme-Employees LDAP source includes sales and support resources, as well as developers. The Acme-Developers LDAP source includes Acme employees who are developers, as well as developer contractors who are not Acme employees. If you want the list of potential resources to include all Acme employees that are developers, and the business process needs attribute data from both LDAP sources,

you would add both sources to one container, using filter criteria to filter out all resources other than Acme developers.

Conversely, if the attribute data needed by the business process is available in one of the LDAP sources, it is much more efficient to include only the one LDAP source in the container.

*Potential resources* are the resources in an LDAP container that can be created and mapped to groups and positions. A list of potential resources is created by the system as follows:

- It starts with a list of potential resources from the primary LDAP source (that satisfy the specified query string in the **PrimaryLdap** parameter of the [saveContainer](#) function).
- An attempt is made to match those potential resources with entries in each secondary LDAP source. If an exact match is found in every secondary LDAP source, the data from the secondary sources is merged in with the data from the primary source.
  - If a potential resource is not found inside one or more of the secondary LDAP sources, the potential resource is eliminated from the list.
  - If matches are found in every secondary LDAP source, they must uniquely identify only one LDAP entry in each source. If one or more match multiple items, the item remains in the potential resources list but is marked as invalid.

This primary / secondary match is accomplished by specifying the **PrimaryLink** within the **SecondaryLdapQuerySource**, when calling the [saveContainer](#) function. A **PrimaryLink** specifies an attribute in the primary LDAP, and the attribute which it must match in each secondary LDAP.

The response simply returns the ID of the newly-created (or updated) LDAP container.

The goal of comparing primary with secondary attributes is to ensure that data from the secondary LDAP source is only merged together with the appropriate potential resource from the primary LDAP source.

Where a match cannot be found, or where it is not one-to-one, the potential resource will not have a complete, accurate set of information, and it will be either omitted (where no match is found) or marked as invalid (where the match isn't one-to-one).

For example, if there are several resources with the same last name, you need to map more than just the last name; maybe mapping first name too will be enough, or maybe you need to map more attributes (because there may be multiple resources in the secondary LDAP source with the same first and last name — it would not know which to include). Maybe there are other types of data, such as an employee ID that would work better and would avoid inconsistencies in data entry (typos, nicknames, abbreviations, and so on).

You can get the LDAP attributes and their values to use by calling the [listAttributeNames](#) function.



If you choose attributes that contain names, always be aware that there may be differences in the way those names were entered in the different LDAP sources, for example, Bob vs. Robert, Mike vs. Michael, or simple misspellings. Things like employee numbers tend to make good attributes to map.

## LDAP Queries

An LDAP query can be specified as a parameter when listing attribute names and saving LDAP containers.

These functions are accomplished using the following functions: [listAttributeNames](#) and [saveContainer](#).

Query strings must be enclosed in parentheses. This allows you to specify multiple strings, each one enclosed in its own parentheses (see the examples below).

You can use the following special characters with query strings:

Special Character	Meaning
*	Wild card character. Matches zero or more of any character.
&	Logical AND. Returns resources that satisfy the first string AND the second string. Place this special character to the left of the first query string, then enclose the entire expression in parentheses, as follows: <code>(&amp;(string1)(string2))</code>
	Logical OR. Returns resources that satisfy the first string OR the second string. Place this special character to the left of the first query string, then enclose the entire expression in parentheses, as follows: <code>( (string1)(string2))</code>
!	NOT. This means that you want all resources that do NOT match the specified value. Place this special character to the left of the query string to which it applies, inside of the parentheses: <code>!(string))</code>

The following are some examples.

- The following query returns all resources that have **sn** attribute values beginning with “s”:  
`(sn=s*)`
- The following query returns all resources that have **sn** attribute values beginning with “s” or “p”:  
`(|(sn=s*)(sn=p*))`
- The following query returns all resources with **carlicense** attribute values equal to “Full” and **employeetype** attribute values equal to “Permanent”:  
`(&(carlicense=Full)(employeetype=Permanent))`
- The following query returns all resources where **sn** attribute values *don’t* start with “s” and *don’t* start with “p”:  
`(&(!(sn=s*))(!(sn=p*)))`



Depending on the specific LDAP Server being used, the query syntax can vary. If the syntax described above does not return the expected results, consult the documentation for your LDAP Server.

## Mapping Users

Mapping resources is an administrative task that will typically be performed before users start using a client application.

Mapping resources can then be performed on an ongoing basis as resources are added or removed from the system, or if the organization model is revised (for example, new positions or groups).

Mapping resources involves assigning resources to specific groups and/or positions in an organization model, which results in the resources receiving work items that are sent to the groups/positions to which the resources have been mapped. Resources can be mapped to:

- A **group**, which represents a job type within the organization. It allows resources to be grouped by their job characteristics.

Groups can be hierarchical, i.e., you can have parent groups with sub groups. For example, you can have a `CustomerServiceRepresentative` group with subgroups of `MotorInsuranceSpecialist`, `TravelInsuranceSpecialist`, and `HomeInsuranceSpecialist`. Typically, sub-groups are specializations of the parent group. Groups can be nested several layers deep.

If a group has sub-groups, you can assign resources to either the parent group or the sub-group. Resources that belong to sub-groups receive work items that are offered to their parent groups, as well as to the sub-group to which they belong. Using the example above, resources in the `MotorInsuranceSpecialist` group will also receive work items offered to the `CustomerServiceRepresentative` group.

- A **position**, which represents a set of responsibilities for a job within an organization unit. It allows resources to be grouped by job responsibility.

Positions are subordinate to an organization unit in the organization model. An organization unit can have many positions. For example, you can have a `TravelInsurance` organization unit with positions of `OfficeManagerTravel` and `TravelInsuranceCSR`. Positions cannot be nested, although organization units can.

Resources can be mapped to positions, but not organization units.

Resources that belong to a position receive work items offered to the position, as well as work items offered to the organization unit that is the immediate parent of the position.

Before a resource can be mapped to group or position, the resource must be *created*. This is accomplished with the [createResource](#) function. Creating the resource assigns a GUID to that resource.

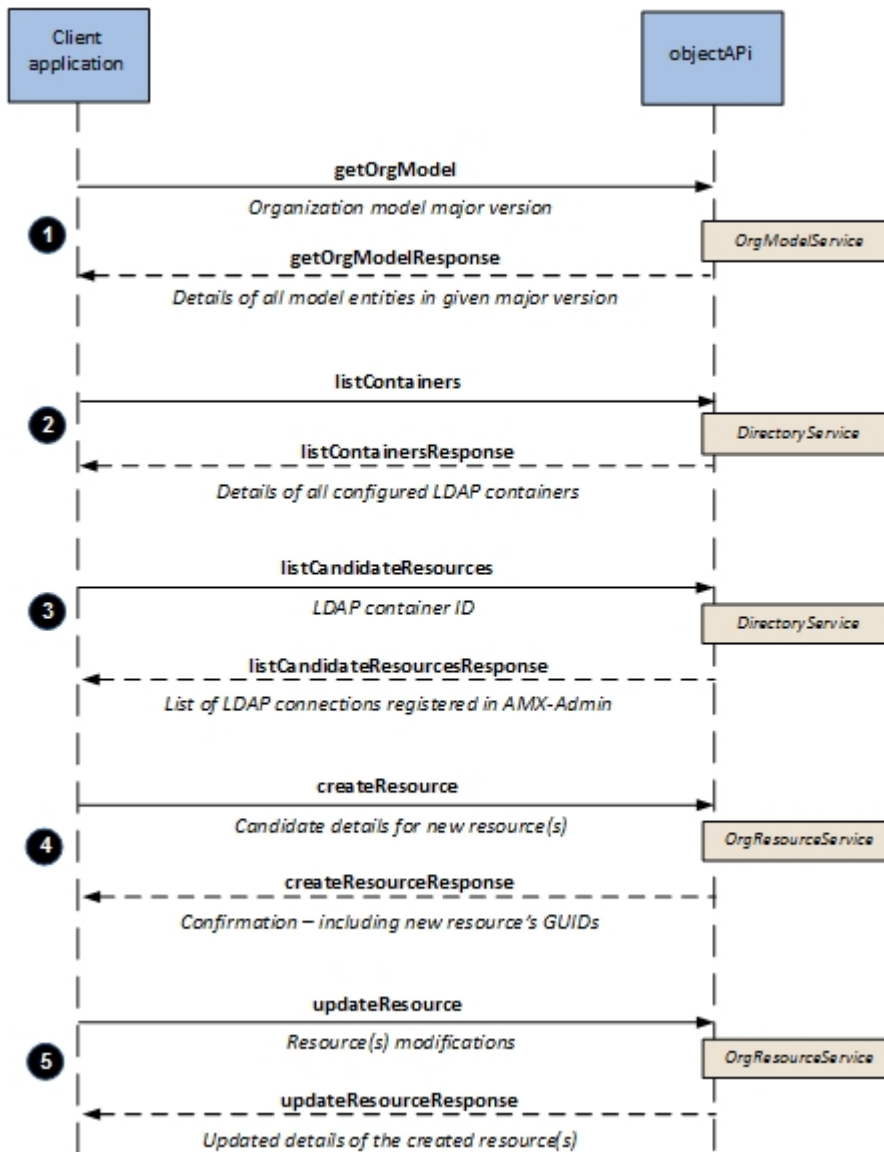
After being created, a resource can log into an ActiveMatrix BPM application, even if the resource has not been mapped to a group or position yet; the resource would not have access to any work items, however, as work items are only sent to resources mapped to groups or positions.

Also note that you will see references to whether or not a resource *exists*; a resource exists if it has been created with the [createResource](#) function.

The following diagram shows an example of how calls to functions in [OrgModelService](#), [DirectoryService](#), and [OrgResourceService](#) can be used to map users.



## Mapping a User



### Procedure

1. Call [getOrgModel](#) to get details about the entities in an organization model identified by its major version number (you can obtain the major version numbers using [listOrgModelVersions](#)).  
The response from the `getOrgModel` function provides the GUIDs for all entities in the organization model, including the positions and groups to which users can be mapped.
2. Call [listContainers](#) to list the LDAP containers to which you (the calling user) have access. LDAP containers contain resources that can be mapped to groups and positions in the organization model.  
The response from the `listContainers` function provides names, descriptions, and other information about each container, as well as the container ID (`id` attribute).
3. List the available resources (candidates) in the specified LDAP container.  
The [listCandidateResources](#) function can be used to list:
  - *all* resources in the container by passing "ALL" in the `includes` parameter,
  - the *existing* resources (that is, those that have been created already with the [createResource](#) function) by passing "EXISTING" in the `includes` parameter, or



- the resources that do not *exist* yet (that is, the resources in the container that have not been created yet with the [createResource](#) function) by passing "NON-EXISTING" in the **includes** parameter.

The response from the listCandidateResources function provides an **ldap-dn** for all available resources; for resources in the container that already exist, the resource's GUID is also returned.

4. Create the resource(s) that you want to map; you can skip this step if you are mapping only resources that already exist (that is, they have already been created and have a GUID assigned to them).

Call [createResource](#), identifying the LDAP container in which the resource resides, as well as the **ldap-dn**, which identifies the resource. If a secondary LDAP source is defined for the container, you must also identify that source in the request.

The response from the createResource function returns the GUID that is assigned to the resource when it is created. The response also contains an **already-present** attribute that tells you whether or not the resource was already created.

5. Map the resource(s) to the desired groups or positions using the [updateResource](#) function.

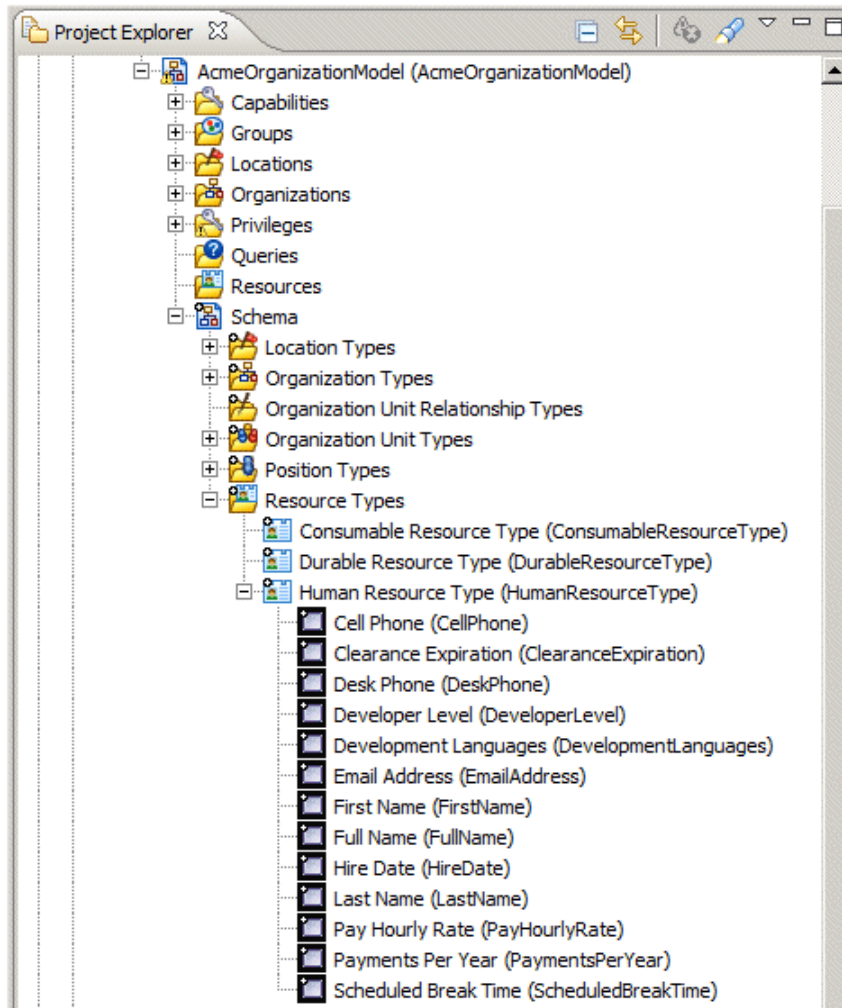
The updateResource function request identifies the resource, as well as the groups and positions to which you are mapping the resource; they are all identified with GUIDs. (Note that the updateResource function can be used to also assign privileges, capabilities, locations, resource attribute values, and so on.)

The response from the updateResource function acknowledges the update by returning properties for the updated resource.

## Resource Attributes

When an organization model is created using the TIBCO Business Studio Organization Modeler, *resource attributes* can also be created. Resource attributes are used to store information about resources. For example, there may be an "EmailAddress" attribute defined, in which each resource's email address is stored. These attributes can contain data that the business process may access during runtime.

The following shows an example of some resource attributes that were created in TIBCO Business Studio:



These attributes are available for each BPM resource. For example, each resource's cell phone number can be stored in the CellPhone attribute.

For information about creating attributes for the "Human Resource Type", see the *TIBCO Business Studio Organization Modeler User's Guide*.

The objectAPI provide functions to set the values of resource attributes for a resource -- see [Setting Resource Attribute Values](#).

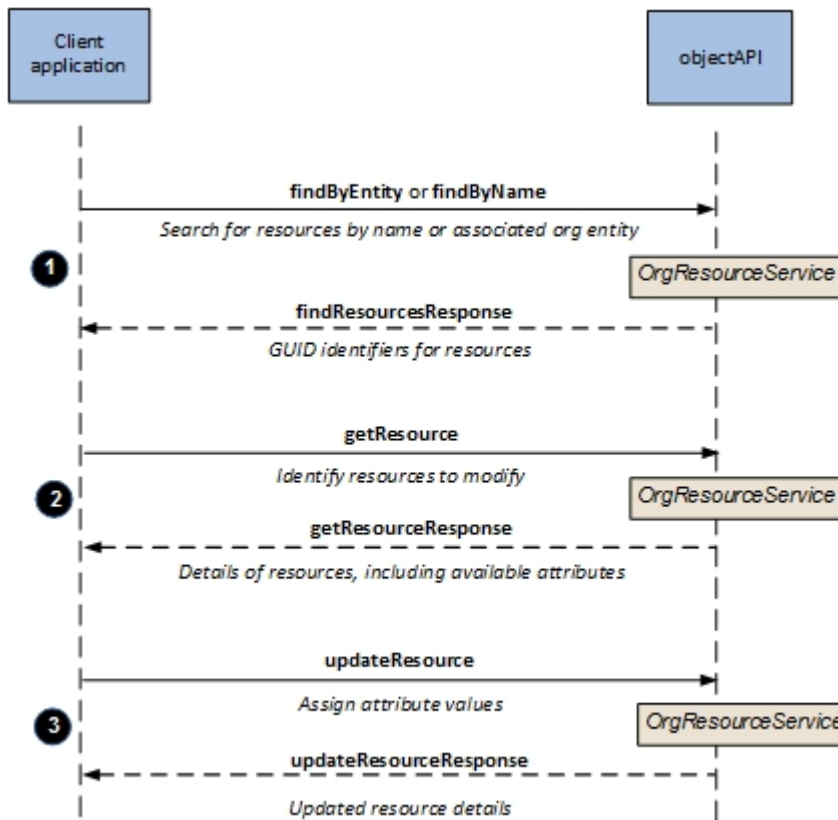
You can also map resource attributes to LDAP attributes so that the process has access to the data in the LDAP attributes at runtime -- see [Mapping Resource Attributes to LDAP Attributes](#).

## Setting Resource Attribute Values

You can set the values of resource attributes using objectAPI functions.

The following diagram shows an example of how calls to the [OrgResourceService](#) functions can be used to add values to attributes.

## Setting Resource Attribute Values



### Procedure

1. Search for resources using either [findByEntity](#) or [findByName](#).  
The search request can include either one or more organization model entities, or resource names, for which associated resources are returned.
2. Get details, including all current attribute values, for the desired resources using the [getResource](#) function.
3. Use the [updateResource](#) function to set any attribute values for the desired resources. (You can also add or remove privileges and capabilities, or assign or remove the resources from the specified groups and positions using the `updateResources` function.)

The response returns all details for the updated resources.

### Mapping Resource Attributes to LDAP Attributes

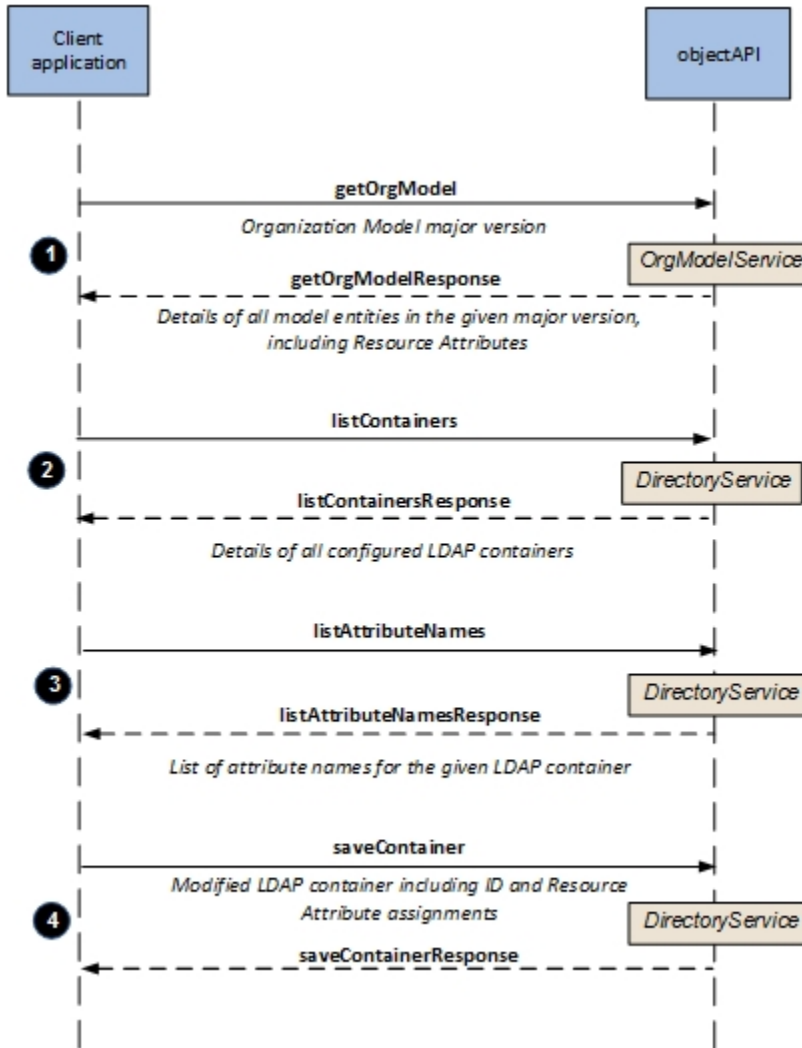
You may need to map one or more resource attributes to attributes in the LDAP sources you have defined in your LDAP container. You may need to do this because the business process does not have direct access to the attributes in the LDAP sources, but it does have access to the resource attributes in the organization model.

When you map a resource attribute to an LDAP attribute, it gives the business process access to the data in the LDAP attribute at runtime.<sup>3</sup>

The following diagram shows an example of how you can map resource attributes to LDAP attributes.

<sup>3</sup> Note that resource attributes of type "Enum" and "EnumSet" cannot be mapped to LDAP attributes.

## Mapping Resource Attributes to LDAP Attributes



### Procedure

1. Call [getOrgModel](#) to get details about the entities in an organization model.  
The request identifies the major version number of the organization model, which can be obtained using the [listOrgModelVersions](#) function.  
The response from the `getOrgModel` function provides GUIDs for all entities in the organization model, including resource attributes defined in the model.
2. Call [listContainers](#) to retrieve the LDAP containers to which you (the calling user) have access.  
The response from `listContainers` provides **ldap-alias** and **ldap-query**, which you will need to request the list of attributes from the LDAP source.
3. Call [listAttributeNames](#) to list the LDAP attributes in the identified LDAP source.
4. Call [saveContainer](#) to specify which resource attributes to map to named LDAP attributes.  
The `saveContainer` function response indicates success by returning the container identifier.

## Organization Relationships

When you are creating or editing an LDAP container, you can specify that the LDAP container have a relationship with one or more organizations.

These *organization relationships* allow you to prevent users from seeing LDAP containers and organizations they are not intended to see, as well as prevent resources from being mapped to positions in organizations they should not be in.

Organization relationships are created by specifying the **RestrictedOrg** when calling the [saveContainer](#) function.

When an organization relationship is set up, it can affect the response from the following functions (unless the caller has a system action that overrides organization relationships — see [Overriding Organization Relationships](#)):

- [getOrgModel](#) - This function returns only those organizations to which the calling user has access because of organization relationships.
- [listCandidateResources](#) - This function returns an empty response if the calling user does not have access to an organization to which an LDAP container is associated because of organization relationships.
- [listContainers](#) - This function returns only the LDAP container in which the caller was created, as well as LDAP containers that have no organization relationships set up.
- [updateResource](#) - This function is used to map users to positions in an organization. It will only allow you to map resources to organizations allowed according to any organization relationships that have been set up. (This cannot be overridden by a system action; if a resource is barred from being mapped to an organization because of an organization relationship, you cannot map that resource to the organization regardless which system actions you possess.)

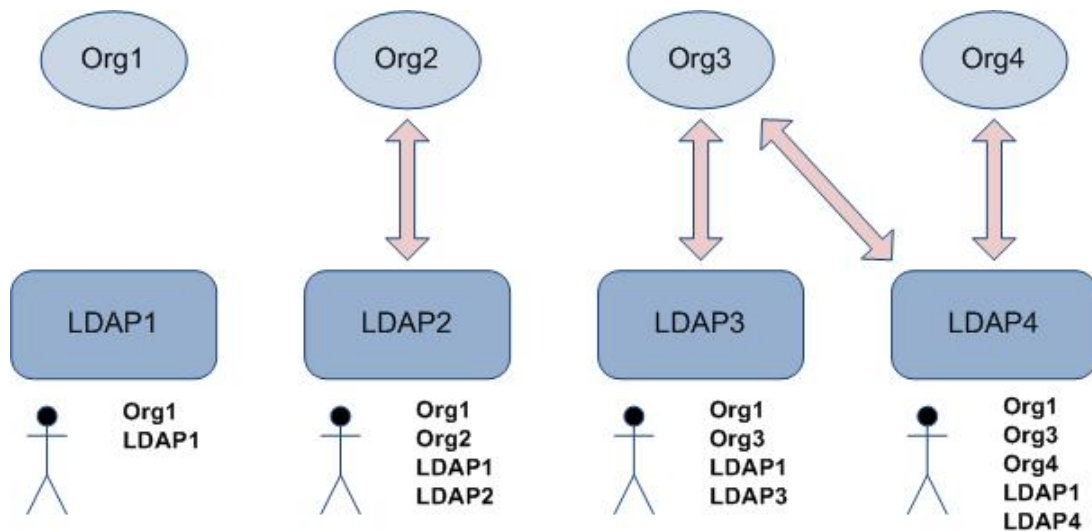


Organization relationships do not apply to groups, as they are separate from, and outside, the organization structure of the organization model.

The following summarizes the result of organization relationships (these descriptions assume the calling resource does not have a system action that overrides organization relationships):

- Resources that are in a container that does not have a relationship with any organization can:
  - see containers that do not have a relationship with any organization.
  - see organizations that do not have a relationship with any container.
  - be mapped only to organizations that do not have a relationship with any container.
- Resources that are in a container that has one or more organization relationships can:
  - see containers that do not have a relationship with any organization, as well as the container they are in.
  - see organizations that do not have a relationship with any container, as well as organizations to which their container has a relationship.
  - be mapped to organizations that do not have a relationship with any container, as well as to the organizations that have a relationship with the container the resource is in.

The following graphic illustrates these points by showing four organizations and four LDAP containers. The arrows represent a relationship between the container and the organization. Under each container is a resource that is in that container, and to the right of each resource is shown the organizations and LDAP containers the resource can see.



- All resources can see Org1 and LDAP1 because neither has an explicit relationship set up.
- The resources in containers that have an organization relationship can also see the LDAP container they are in, as well as the organizations for which their container has a relationship.
- Any of the resources can be mapped to Org1, as well as to the organization(s) to which their container has a relationship.

An important point to understand here is that if multiple containers have a relationship with a single organization, the resources in one container will *not* be able to see the resources from the other container when viewing the organization to which they both have a relationship.

For example, using the illustration above, if a resource from both LDAP3 and LDAP4 are mapped to the same position in Org3, when a resource from LDAP3 looks at that position (using the Organization Browser, when creating supervised work views, or when allocating work items to world), that resource will *not* see the LDAP4 resource that is mapped to that position. Likewise, a resource from LDAP4 will not see the LDAP3 resource when looking at that position.



If a resource has been mapped to a position, then an organization relationship is set up that would bar that resource from membership to that position, it results in an *invalid membership*. Resources that have been mapped to invalid memberships should be removed (the [updateResource](#) function can be used to remove memberships).

## Overriding Organization Relationships

System actions are provided that override organization relationships, giving the caller access to all organizations, regardless the organization relationships that have been set up. These system actions are typically given to administrative users.

The system actions that override organization relationships are:

- **organizationAdmin** - This system action is only applicable to [OrgModelService](#) functions. Users with this system action will see all organizations when calling functions that return organization models, regardless the organization relationships set up.

Note that to call any function in the [OrgModelService](#), the user must also possess the `browseModel` system action — holders of the `organizationAdmin` system action get additional access (if there are organization relationships defined).

- **LDAPAdmin** - This system action, which is required for many functions, may also give the caller access to all organizations, regardless the organization relationships set up, depending on the operation.

When calling the [DirectoryService](#) functions listed below, the caller must possess *either* the `resourceAdmin` or `LDAPAdmin` system action. If the caller has only the `resourceAdmin` system

action, the organizations that the user can see are restricted by organization relationships. But if the user also has (or has only) the LDAPAdmin system action, all organizations will be visible, regardless the organization relationships, when using the following functions:

- [listCandidateResources](#)
- [listPagedCandidateResources](#)
- [listContainers](#)

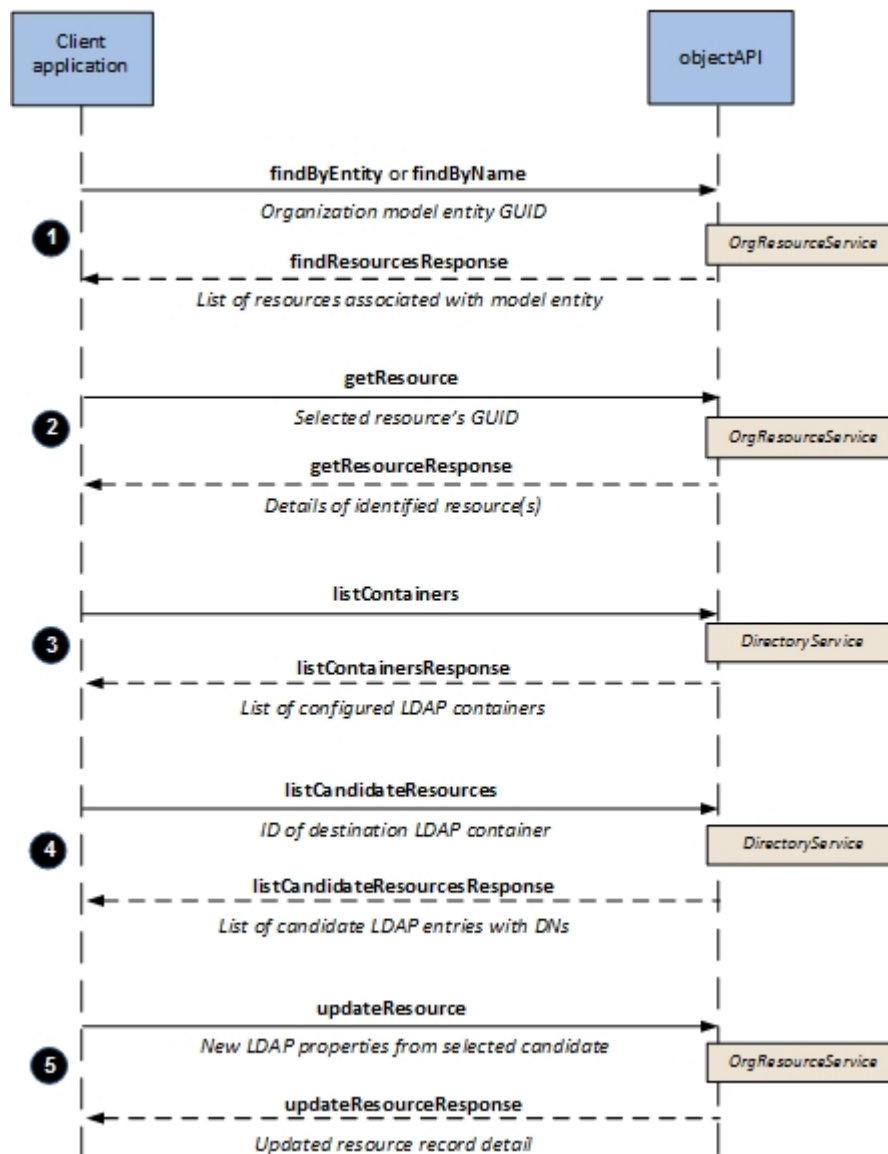
You can determine if a user has a specific system action by using the [listAuthorisedOrgs](#) function.

## Moving a Resource to Different LDAP Container

You can change the LDAP references of a resource, for instance to move it to a different LDAP container. This may be desired if a resource has moved to a different part of the company.

The following diagram shows an example of how to move a resource.

### *Moving a Resource to a Different LDAP Container*





## Procedure

1. Call either [findByEntity](#) or [findByName](#) to find a resource for user selection.
2. Call [getResource](#) to get resource details for confirmation.
3. Call [listContainers](#) to list LDAP containers for selection of the destination LDAP container.
4. Call [listCandidateResources](#) to get candidates from the destination LDAP container.
5. Call [updateResource](#) to update the resource with the selected candidate details.

The `updateResource` request must include an LDAP reference for each LDAP source referenced by the LDAP Container.

## Updating Push Destinations

Work items can be "pulled" when a user logs into a BPM application and accesses their work list; this "pulls", from the server, the work items that were sent to that user.

Work items can also be "pushed" to users as an email. The email contains the URL of the work item, which the user can click to open and process the work item.

A *push destination* is the specification of where the email is pushed. Push destinations can be specified for:

- **Organizational entities** - These specify the destination(s) to which work items sent to the organizational entity are to be pushed. You can specify one or more push destinations for each organizational entity.
- **Resources** - These specify the destination(s) to which work items that are sent *directly* to the resource are to be pushed. You can specify one or more push destinations for each resource.

For more information, see:

[Updating Push Destinations for Organizational Entities](#)

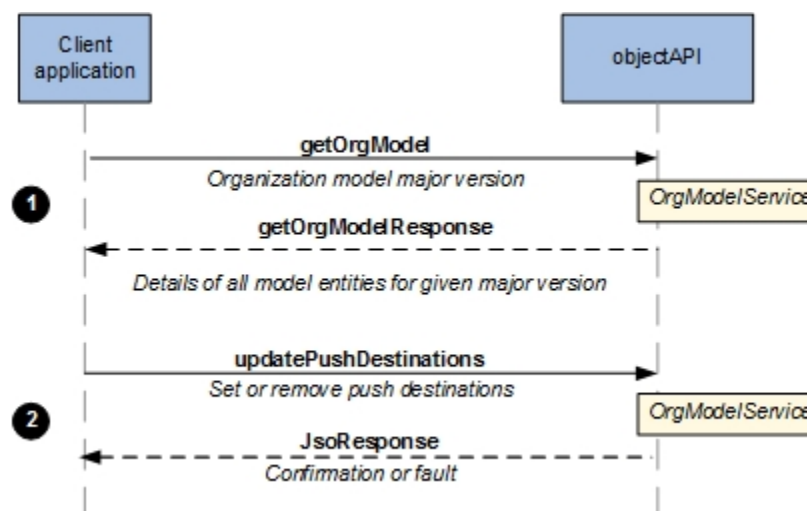
[Updating Push Destinations for Resources](#)

## Updating Push Destinations for Organizational Entities

This topic describes how to push work items, via email, to organizational entities.

Functions in [OrgModelService](#) are used to specify push destinations for organizational entities.

### Updating Push Destinations for Organizational Entities





## Procedure

1. Call [getOrgModel](#) to get details about the entities in an organization model.

The request identifies the major version number of the organization model, which can be obtained using the [listOrgModelVersions](#) function.

The response from the [getOrgModel](#) function provides GUIDs for all entities in the specified version of the organization model.

2. Call [updatePushDestinations](#) to set push destinations for the desired organizational entities.

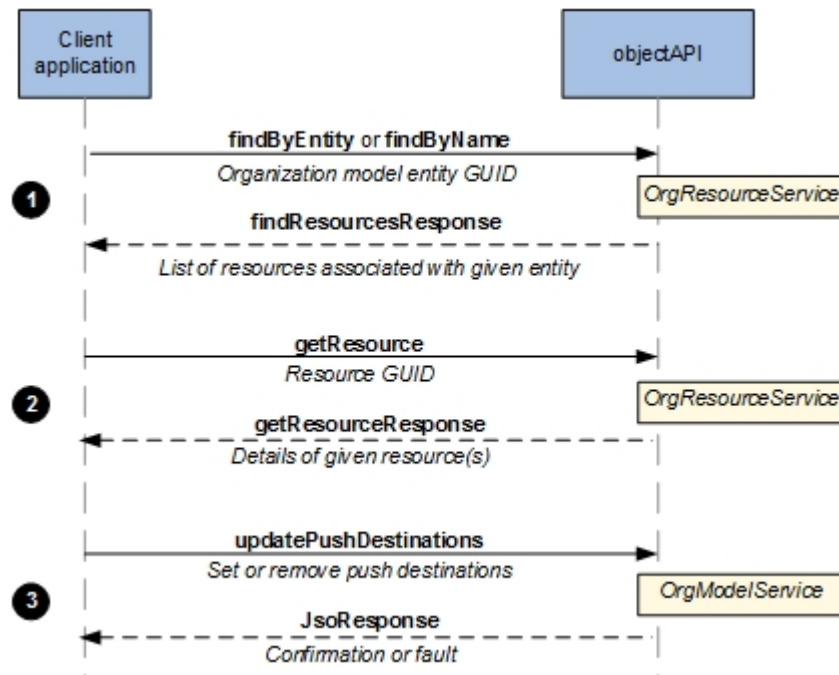
An empty response indicates a successful execution.

## Updating Push Destinations for Resources

This topic describes how to push work items, via email, to resources.

Functions in [OrgModelService](#) and [OrgResourceService](#) are used to specify push destinations for resources.

### Updating Push Destinations for Resources



## Procedure

1. Call [findByEntity](#) or [findByName](#) to get details about the resources in an organization model.

The request identifies the organizational entity, which can be obtained using the [getOrgModel](#) function.

The response from the [findResources](#) function provides a list of resources that are associated with the given organizational entity.

2. Call [getResource](#) to get full details of the identified resource(s), including the push destinations already defined for the resource(s).
3. Call [updatePushDestinations](#) to set push destinations for the specified resource(s).

An empty response indicates a successful execution.

## Configuring a Dynamic Organization Model Extension Point

An organization model extension point is an organization unit that references an organization model template. The extension point configuration is used to dynamically generate instances of the organization model template directly below it.

The designation of the extension point is a design-time function; it consists only of the assignment of the organization model template. The remaining extension point configuration — that is, the LDAP connection information — is not known nor can be interrogated at design-time, and is therefore performed after deployment.

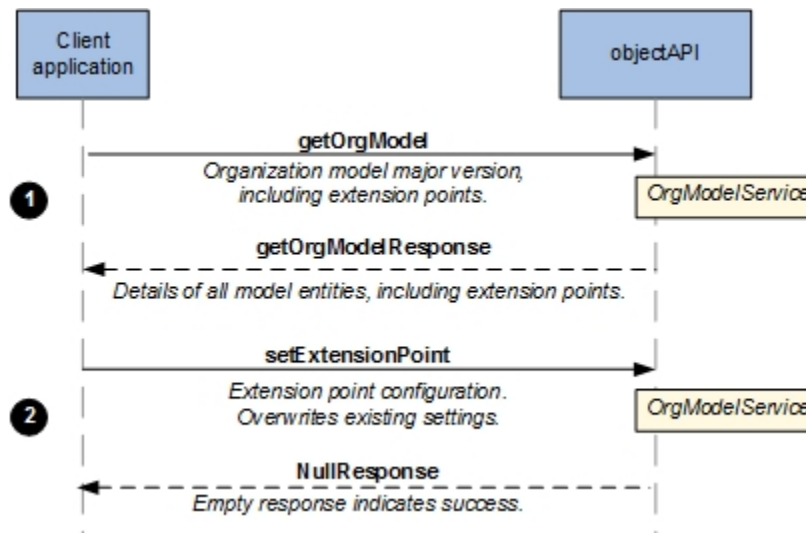
Note that any extension point that is not fully configured is ignored, and not does result in the creation of organization model template instances.

After an extension point is configured, instances of the organization model template are generated using the **ExtensionPointProcessStart** property in `DE.Properties`. For information about all extension point-related properties, see the *TIBCO ActiveMatrix BPM Administration* guide.

For more information about dynamic organizations, see the *TIBCO Business Studio Modeling* guide.

The following diagram shows an example of organization model extension point configuration.

### Configuring a Dynamic Organization Model Extension Point



### Procedure

1. Call [getOrgModel](#) to get an organization model with extension points.  
The response also includes dynamic organization model templates.
2. Call [setExtensionPoint](#) to set the extension point configuration.

### Candidate Queries

A candidate query is an LDAP Query assignment to a position or group.

Candidate queries are configured using the [setGroupCandidateQuery](#) and [setPositionCandidateQuery](#) functions. After a candidate query is configured, the query is scheduled to run, using the **AutoResourceGenStart** property in the `DE.properties` file, and candidates are assigned to the group or position. For information about all candidate query-related properties, see the *TIBCO ActiveMatrix BPM Administration* guide.

An LDAP container must be specified in the [setGroupCandidateQuery](#) and [setPositionCandidateQuery](#) functions. The primary LDAP source of the LDAP container identifies the LDAP connection on which

the query is performed. This also determines the LDAP container to which any newly created resources are assigned.

Depending on the class of the primary LDAP source of the LDAP container, further configuration may be allowed. There are two classes of LDAP sources:

- Those that identify candidates as the members of an LDAP group. For this class, no further configuration is available; the candidate query will take all the entries identified by the LDAP container as its candidate list.
- Those that identify candidates using an LDAP query. For this class, the following additional configuration is available:
  - **Base-DN** - Names the LDAP branch to which the LDAP query (see below) will be restricted. This is optional and is relative to any Base-DN already configured on the primary LDAP source of the identified LDAP container.
  - **LDAP Query** - This expression locates entries that identify candidate resources. The expression is combined with that of the primary LDAP source of the identified LDAP container.
  - **Search Scope** - This determines the depth to which the search is performed, as follows:

ONELEVEL	Only the elements directly within the Base-DN level are searched.
SUBTREE	Elements directly within, and below, the Base-DN level are searched.

The candidate query cannot be more inclusive in its Search Scope than the LDAP container's primary LDAP source. Therefore, if the primary LDAP source has a search scope of ONELEVEL, then the candidate query must also use ONELEVEL. However, if the primary source is SUBTREE, then the candidate query may be either.

The Base-DN and LDAP Query of the LDAP container's primary LDAP source are combined with those of the candidate query. The following table shows how the properties are combined.

	Base-DN	LDAP Query
Primary LDAP Source	o=acme	(objectclass=person)
Candidate Query	ou=london	(department=1234)
Combined Result	ou=london,o=acme	(&(objectclass=person)(department=1234))

Any resource identified by the candidate query of a group or position must also be visible via the associated LDAP container. That is, no resource can be created dynamically that could not also be created manually using an LDAP container. This ensures that any resource attributes are able to retrieve their values from the mapped LDAP attributes of an LDAP container.

Each candidate query will only identify potential entries from the primary LDAP source of the associated LDAP container. If that LDAP container has any secondary LDAP sources, the rules that bind entries within the secondary LDAP sources to those of the primary LDAP source must be followed. It is only when those rules have been completed that the true set of candidate resources can be resolved.

The deletion of the LDAP container will, depending on the request parameters, cause the deletion of all resources belonging to that LDAP container; whether they were created manually or dynamically. If the request to delete the LDAP container does not specify that associated resources should also be deleted,

and associated resources exist, the request will fail. The deletion of the LDAP container will always result in the deletion of candidate queries that reference that LDAP container.

### Populating Instantiated Model Templates Using Substitution Variables

Positions and groups can be populated using candidate queries assigned to those positions and groups. By extending this slightly, and assigning candidate queries to positions within an organization model template (see [Configuring a Dynamic Organization Model Extension Point](#)), it is possible to both instantiate an organization model template, and populate it.

By using simple variable substitution, it is possible to construct the Base-DN and/or the LDAP query of the candidate query in such a way that it identifies the resources for a particular organization model template instance. The substitution variables must be enclosed in braces: { }.

The following substitution variables are available for reference within a candidate query (the variable names are case-insensitive):

- **Root-DN** - The DN of the LDAP entry that initiated the organization model template instance.
- **Root-Name** - The name assigned to the root organization unit of the model template instance; that is, the value of the LDAP attribute named in the extension point.

An example of a Base-DN containing substitution variables:

```
"ou= {root-name} , {root-dn}"
```

An example of an LDAP Query:

```
"(&(ou= {root-name})(dept=123)(role=manager))"
```

Candidate query substitution variables can only be used for organization model templates, and are evaluated only when the organization model template is instantiated.

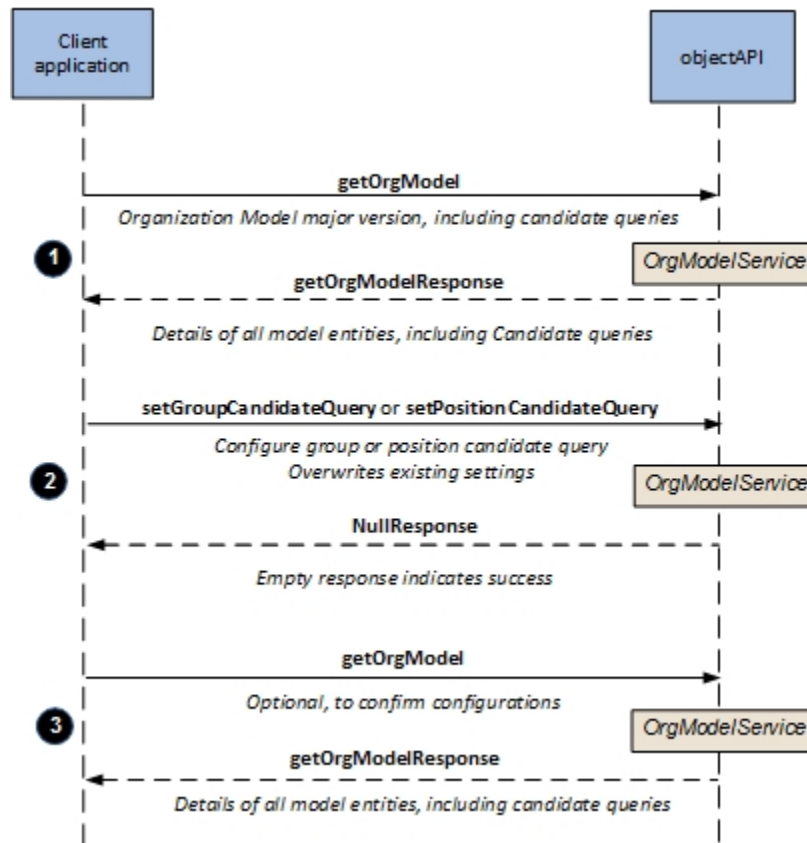
### Configuring a Candidate Query

Candidate queries are configured so that candidates are assigned to a group or position.

Candidate queries are configured using the [setGroupCandidateQuery](#) and [setPositionCandidateQuery](#) functions.

The following diagram shows an example of configuring a candidate query.

### Configuring a Candidate Query



#### Procedure

1. Call `getOrgModel` to get an organization model with candidate queries.
2. Call `setGroupCandidateQuery` or `setPositionCandidateQuery` to set the candidate query, depending on whether it is for a group or a position.
3. Call `getOrgModel` again (requesting candidate queries) to get the organization model again to confirm configuration.

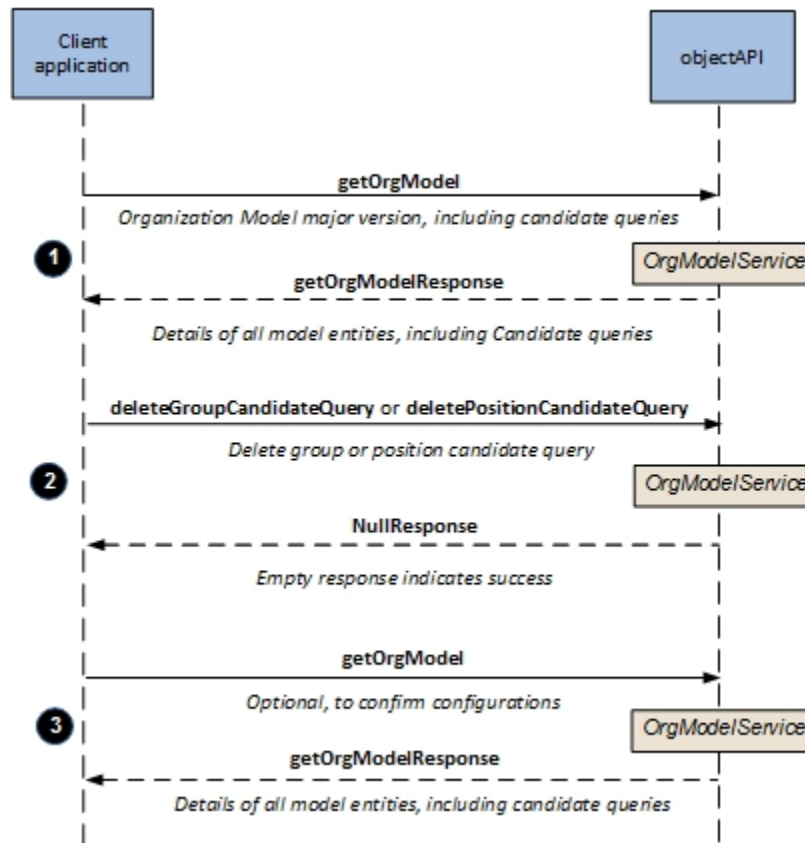
### Removing a Candidate Query Configuration

Candidate query configurations can also be deleted.

This is done using the `deleteGroupCandidateQuery` and `deletePositionCandidateQuery` functions.

The following diagram shows an example of configuring a candidate query.

### Removing a Candidate Query



#### Procedure

1. Call [getOrgModel](#) to get an organization model with candidate queries.
2. Call [deleteGroupCandidateQuery](#) or [deletePositionCandidateQuery](#) to remove the candidate query, depending on whether it is for a group or a position.
3. Call [getOrgModel](#) again (requesting candidate queries) to get the organization model again to confirm removal.

#### Processes

A *process* defines the flow of information in your application. It ensures that information flows in a consistent and timely manner through the system.

Processes are created using TIBCO Business Studio™. They are saved as *process templates*.

A *process instance* is created when a process template is “started,” and remains in existence until that instance of the process is purged from the system.

Process instances are typically started when you start a *business service*. A business service is a set of actions that accomplishes some sort of business function. For example, a business service could be designed to handle an incoming insurance claim.

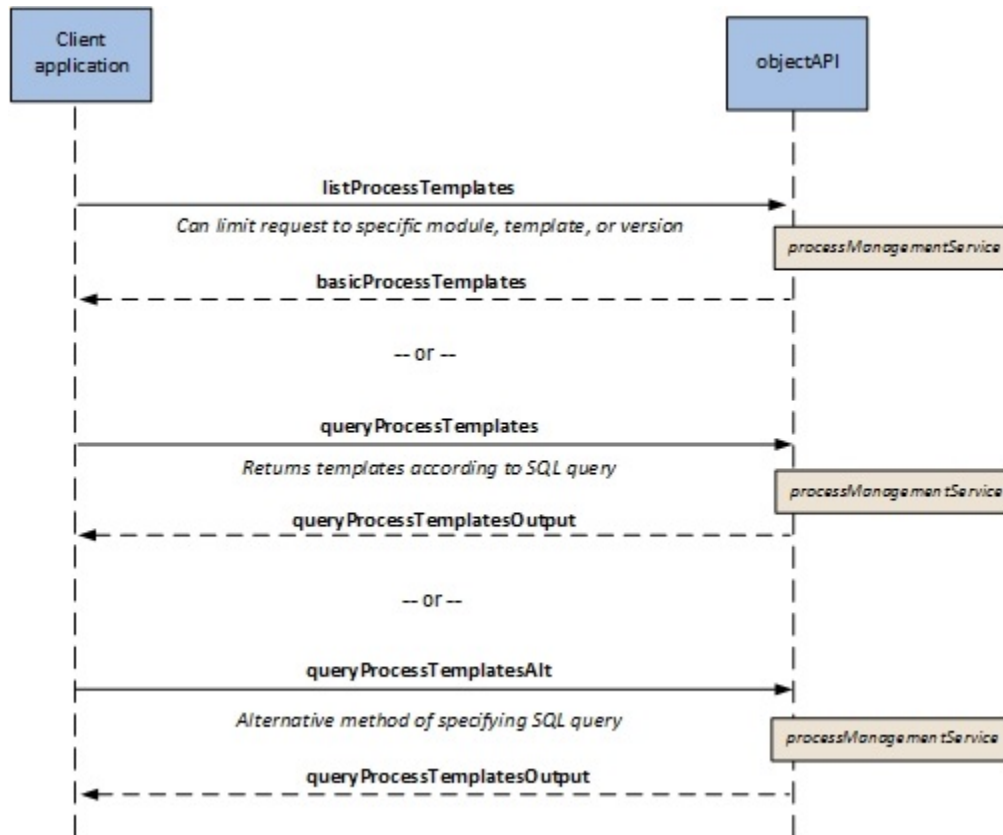
#### Listing Available Process Templates

There are three functions available to retrieve process templates, each requiring a different set of criteria.

These [ProcessManagementService](#) functions are:

- [listProcessTemplates](#)
- [queryProcessTemplates](#)
- [queryProcessTemplatesAlt](#)

#### Listing Available Process Templates



#### Starting a Process Instance

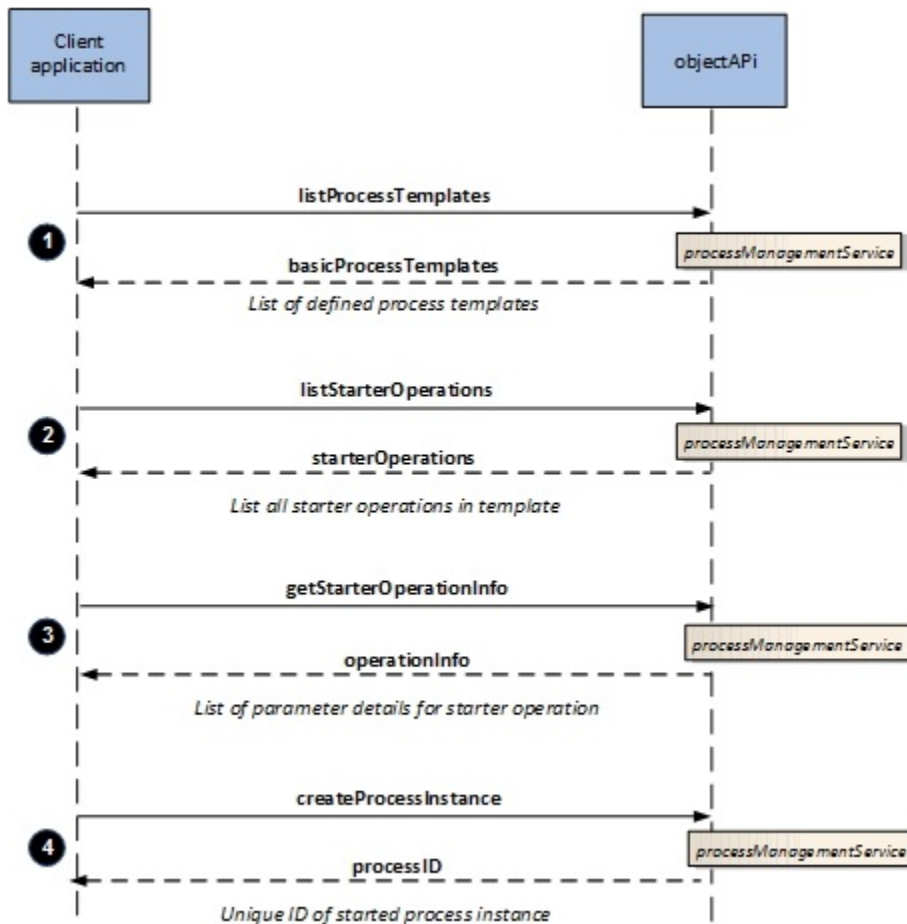
Starting a process instance involves "creating" an instance of a process template.

The way in which you create, or start, an instance of a process template depends on the *trigger type* of the start event in the process template definition, as follows:

- "None" - A process with a start event with this trigger type can be directly started using the [createProcessInstance](#) function. This topic describes starting a process instance using the [createProcessInstance](#) function
- "Message" - A process with a start event with this trigger type can be started in the following ways:
  - If the process was "published as a business service", an instance of the process is started by starting the business service. For information, see [Starting a Business Service](#). This is a common way to start a process instance.
  - If the process was "published as a REST service", an instance of the process is started by using the [startProcessIncomingReceiveTask](#) function.

The following diagram shows an example of how calls to the [ProcessManagementService](#) API can be used to start a process instance.

### Starting a Process Instance



### Procedure

1. Find the available process templates.

The [listProcessTemplates](#) function can be used to list available process templates on the node. (You could also use the [queryProcessTemplates](#) or [queryProcessTemplatesAlt](#) functions to get a list of process templates.)

The response from the `listProcessTemplates` function provides module names, process name, and versions of the available process templates. These are needed when calling the [createProcessInstance](#) function to start a process instance.

2. Get starter operations using the [listStarterOperations](#) function.

Starter operations refer to start events of processes. To be able to directly start an instance of a process template, the start event of the process must have a trigger type of "None" (as opposed to a trigger type of "Message", which requires that the process be started by a business service).

3. If the process template for which you are starting an instance has "start parameters" (also known as "formal parameters"), which are used to pass data into the process instance being started, you should also call the [getStarterOperationInfo](#) function to get the names of the available start parameters.

4. Start the process instance.

The [createProcessInstance](#) function is used to start the process instance and pass in start parameters, if required.

The function returns the process ID of the instance that is started.

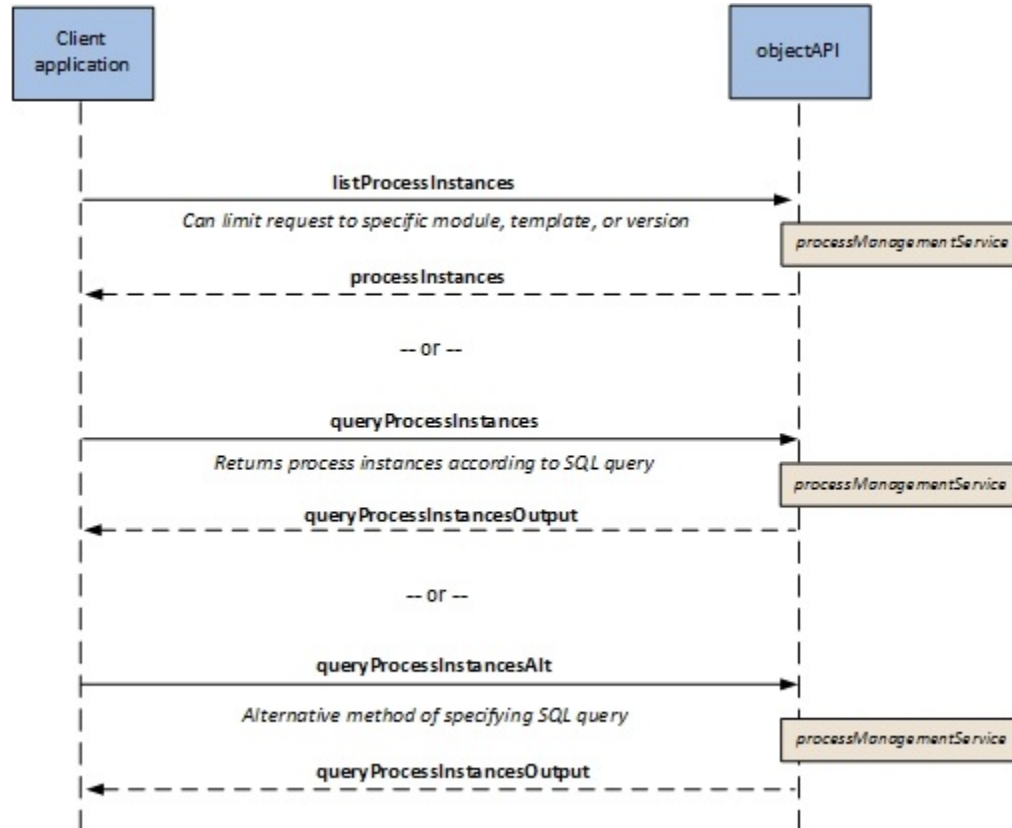


## Listing Available Process Instances

There are three functions available to retrieve process instances, each requiring a different set of criteria. These [ProcessManagementService](#) functions are:

- [listProcessInstances](#)
- [queryProcessInstances](#)
- [queryProcessInstancesAlt](#)

### Listing Available Process Instances



## Sorting and Filtering Lists of Process Templates and Instances

The objectAPI contains functions that include a query parameter, which contains an expression defining the sort and/or filter criteria to be applied to lists of process templates and instances.

These functions, which are available in the [ProcessManagementService](#), are:

- [queryProcessTemplates](#) - Lists process templates that match certain criteria.
- [queryProcessTemplatesAlt](#) - Lists process templates that match certain criteria. (This is a variation of `queryProcessTemplates` in which the query string is divided into its constituent parts.)
- [queryProcessTemplateCount](#) - Counts the number of process templates that match certain criteria.
- [queryProcessInstances](#) - Lists process instances that match certain criteria.
- [queryProcessInstancesAlt](#) - Lists process instances that match certain criteria. (This is a variation of `queryProcessInstances` in which the query string is divided into its constituent parts.)
- [queryProcessInstanceCount](#) - Counts the number of process instances that match certain criteria.
- [queryProcessInstanceCountAlt](#) - Counts the number of process instances that match certain criteria.

For information about the query syntax, see [Query Parameter Syntax](#).

## Query Parameter Syntax

When querying process templates and instances, the query parameter must use specific SQL syntax.

For example:

```
SELECT attribute(s) FROM process WHERE condition
[ORDER BY attribute(s) ASC|DESC];
```

where:

- *attribute* is a valid process data attribute. For list of valid attributes, see [Query Parameter Attributes](#).
- *condition* is an expression that defines the filter to be used. The *condition* expression must use the following syntax:

```
attribute operator value
```

where:

- *attribute* is a valid process data attribute for the WHERE clause. For list of valid attributes, see [Query Parameter Attributes](#).
- *operator* is a valid operator for the WHERE clause. For a list of valid operators, see [Query Parameter Operators](#).
- *value* is a constant literal. The constant literal can be a numeric type, date/time type or text type. A text literal must be enclosed in single quotes e.g. 'mytext'.

SQL keywords such as SELECT, FROM and WHERE are case insensitive.

The ORDER BY clause can be used to sort the data.

Filtering can be done based on multiple conditions by the use of the **AND** and **OR** logical operators. Parenthesis can be used to form complex expressions. For example:

- SELECT attribute(s) FROM process WHERE condition1 AND condition2;
- SELECT attribute(s) FROM process WHERE condition1 OR condition2;
- SELECT attribute(s) FROM process WHERE (condition1 OR condition2) AND condition3;

Only a subset of the SQL SELECT statement is supported. The following features are not supported:

- SELECT in the WHERE clause
- JOIN
- GROUP BY
- SQL FUNCTIONS
- SELECT INTO



## Query Parameter Attributes

All attributes can be used for both filtering and sorting.

Note that:

- If the attribute specified is "\*", all attributes for the process template or process instance will be returned.
- The predefined attribute names, with the exception of globalDataRef, are prefixed by DEFINITION, MODULE or INSTANCE.
- The predefined attributes are case insensitive, with the exception of globalDataRef. User-defined attribute names are case sensitive.

The following attributes can be used to query process templates.

Attribute Name	Description	Type
DEFINITION.NAME	Process name	Text
DEFINITION.DESCRPTION	Process description	Text
DEFINITION.VERSION	Process version	Text
DEFINITION.PRIORITY	Process priority	Numeric (short)
DEFINITION.CREATION_DATE	Process creation date	Date/Time
DEFINITION.SERVICE_PROCESS	Whether the process is a service process (true) or a business process (false)	Boolean
MODULE.NAME	Module name	Text

The following attributes can be used to query process instances.

Attribute Name	Description	Type
MODULE.NAME	Module name	Text
INSTANCE.NAME	Process instance name	Text
INSTANCE.ID	Process instance identifier	Text
INSTANCE.ORIGINATOR_NAME	Originator of the process instance	Text
INSTANCE.PARENT_PROCESS_ID	ID of the parent process	Text
INSTANCE.VERSION	Process instance version	Text
INSTANCE.PRIORITY	Process instance priority	Numeric (short)
INSTANCE.STATUS	Process instance state/status	Text
INSTANCE.START_DATE	Process instance start date	Date/Time
INSTANCE.COMPLETION_DATE	Process instance completion date	Date/Time
INSTANCE.WAITING_WORK_COUNT	Process instance waiting work count	Numeric (integer)
INSTANCE.FAILED_ACTIVITY_NAME	Name of the failed activity that resulted in a halted process instance	Text
INSTANCE.ACTIVITY_FAULT_DATA	Activity default data	Text
INSTANCE.ACTIVITY_FAULT_NAME	Name of activity fault	Text

Attribute Name	Description	Type
globalDataRef <sup>4</sup>	Global data reference	Text
<i>User-defined attribute name</i>	Process instance custom attribute	Text, Numeric, Date-Time, Boolean

### Using User-Defined Attributes

If a user-defined attribute is used in the SELECT statement of a paginated query, the attribute and its type should be included in the request, otherwise a database query must be made to get the type of the user-defined attribute, making the request less efficient.

To include the attribute and type in the request, use the **names** and **types** parameters with the [queryProcessInstances](#) and [queryProcessInstancesAlt](#) functions.

For the [queryProcessInstanceCount](#) function (which doesn't provide the **names** and **types** parameters), you can specify a type casting as an in-place alternative. For example:

```
SELECT MODULE.NAME, INSTANCE.ID, LONG(BALANCE), DOUBLE(PAYMENT)
```

### Query Parameter Operators

Specific operators can be used in a query WHERE clause.

Note that the **attribute** and **value** operands must both be of the same data type.

Unless otherwise noted, each operator can be applied to all data types (Text, Numeric, Boolean and DateTime).

Operator	Description	Examples
=	Equals - compares two values for equality.  Wildcards can be used when comparing attributes of type string. See <a href="#">Using Wildcards</a> .	INSTANCE.NAME = 'LoanApproval'  INSTANCE.PRIORITY = 3  IsLoanApproved = TRUE  INSTANCE.COMPLETION_DATE = TS '2008-06-20T10:30:20'
<>	Not equal to	INSTANCE.PRIORITY <> 3
>	Greater than	INSTANCE.PRIORITY > 5000  Balance > 300.53
<	Less than	INSTANCE.PRIORITY < 5000
>=	Greater than or equal to	INSTANCE.PRIORITY >= 5000
<=	Less than or equal to	INSTANCE.PRIORITY <= 5000

<sup>4</sup> Only applicable for processes that contain global data.

Operator	Description	Examples
LIKE	Pattern match comparison - matches a string value against a pattern string containing wildcard characters. See <a href="#">Using Wildcards</a> .  Applies to Text data types only.	To match any processes with a module name that begins with “module”:  <code>MODULE.NAME LIKE 'module%'</code>  To match any processes with a module name that begins with “module number” and ends with a single character):  <code>MODULE.NAME LIKE 'module number _'</code>
BETWEEN	Range comparison - tests whether a value is <i>between</i> two other values	<code>INSTANCE.PRIORITY BETWEEN 4000 AND 5000</code>  <code>INSTANCE.START_DATE between TS '2008-06-20T10:30:20' AND TS '2008-06-20T10:52:20';</code>
IN	Implements comparison to a list of values, that is, it tests whether a value matches any value in a list of values.  Applies to Text and Numeric data types only.	<code>MODULE.NAME IN ('moduleBalance', 'modulePayment');</code>
IS [NOT] NULL	Tests whether the attribute has a value	<code>MODULE.NAME IS NOT NULL</code>  <code>Payment IS NULL</code>
NOT	Inverts the result of a condition clause	<code>MODULE.NAME NOT IN ('moduleBalance', 'modulePayment');</code>  <code>INSTANCE.PRIORITY NOT BETWEEN 4000 AND 5000</code>

### Using Wildcards

Wildcards can be used with the LIKE operator (and with the = operator when comparing attributes of type string).

For example:

- ‘?’ or ‘\_’ matches any *single* character.
- ‘\*’ or ‘%’ matches zero or more characters.

Wildcards can be escaped using the \ character. For example, “\%”.

Escape characters can be escaped using “\\” which evaluates to matching “\”.

If escaping a non-wildcard character, the escape character will be ignored. For example, “\abc” evaluates to matching “abc”.



The \* wildcard can be used in selection clauses only for non-paginated queries (that is, when the **PageSize** parameter is specified as 0 (zero) when calling the following functions: [queryProcessInstances](#), [queryProcessInstancesAlt](#), [queryProcessTemplates](#), and [queryProcessTemplatesAlt](#)).

## Date Time Data Type Specification

Specific date/time formats must be used in filter expressions.

They are:

General Format	Usage	Example
YYYY	DATE 'YYYY'	DATE '1997'
YYYY-MM	DATE 'YYYY-MM'	DATE '1997-07'
YYYY-MM-DD	DATE 'YYYY-MM-DD'	DATE '1997-07-16'
YYYY-MM-DDThh:mm+hh:mm	TS 'YYYY-MM-DDThh:mm+hh:mm'	TS '1997-07-16T19:20+01:00'
YYYY-MM-DDThh:mm-hh:mm	TS 'YYYY-MM-DDThh:mm-hh:mm'	TS '1997-07-16T19:20-01:00'
YYYY-MM-DDThh:mmZ	TS 'YYYY-MM-DDThh:mmZ'	TS '1997-07-16T19:20Z'
YYYY-MM-DDThh:mm:ss	TS 'YYYY-MM-DDThh:mm:ss'	TS '1997-07-16T19:20:30'
YYYY-MM-DDThh:mm:ss+hh:mm	TS 'YYYY-MM-DDThh:mm:ss+hh:mm'	TS '1997-07-16T19:20:30+01:00'
YYYY-MM-DDThh:mm:ss-hh:mm	TS 'YYYY-MM-DDThh:mm:ss-hh:mm'	TS '1997-07-16T19:20:30-01:00'
YYYY-MM-DDThh:mm:ssZ	TS 'YYYY-MM-DDThh:mm:ssZ'	TS '1997-07-16T19:20:30Z'
YYYY-MM-DDThh:mm:ss.s	TS 'YYYY-MM-DDThh:mm:ss.s'	TS '1997-07-16T19:20:30.45'
YYYY-MM-DDThh:mm:ss.s+hh:mm	TS 'YYYY-MM-DDThh:mm:ss.s+hh:mm'	TS '1997-07-16T19:20:30.45+01:00'
YYYY-MM-DDThh:mm:ss.s-hh:mm	TS 'YYYY-MM-DDThh:mm:ss.s-hh:mm'	TS '1997-07-16T19:20:30.45-01:00'
YYYY-MM-DDThh:mm:ss.sZ	TS 'YYYY-MM-DDThh:mm:ss.sZ'	TS '1997-07-16T19:20:30.45Z'

Where:

- YYYY = four-digit year
- MM = two-digit month (01=January, etc.)
- DD = two-digit day of month (01 through 31)
- hh = two digits of hour (00 through 23) (am/pm NOT allowed)

- mm = two digits of minute (00 through 59)
- ss = two digits of second (00 through 59)
- s = one or more digits representing a decimal fraction of a second

If no time zone information is given with a time representation, the time is assumed to be in local time.

## Query Parameter Examples

This topic provides query examples.

Using boolean types:

```
SELECT * FROM process WHERE (isBalanceFull = TRUE);
SELECT * FROM process WHERE (isBalanceFull = FALSE);
```

Using numeric types:

```
SELECT * FROM process WHERE (Payment > 3.569);
SELECT * FROM process WHERE (Payment < 50);
```

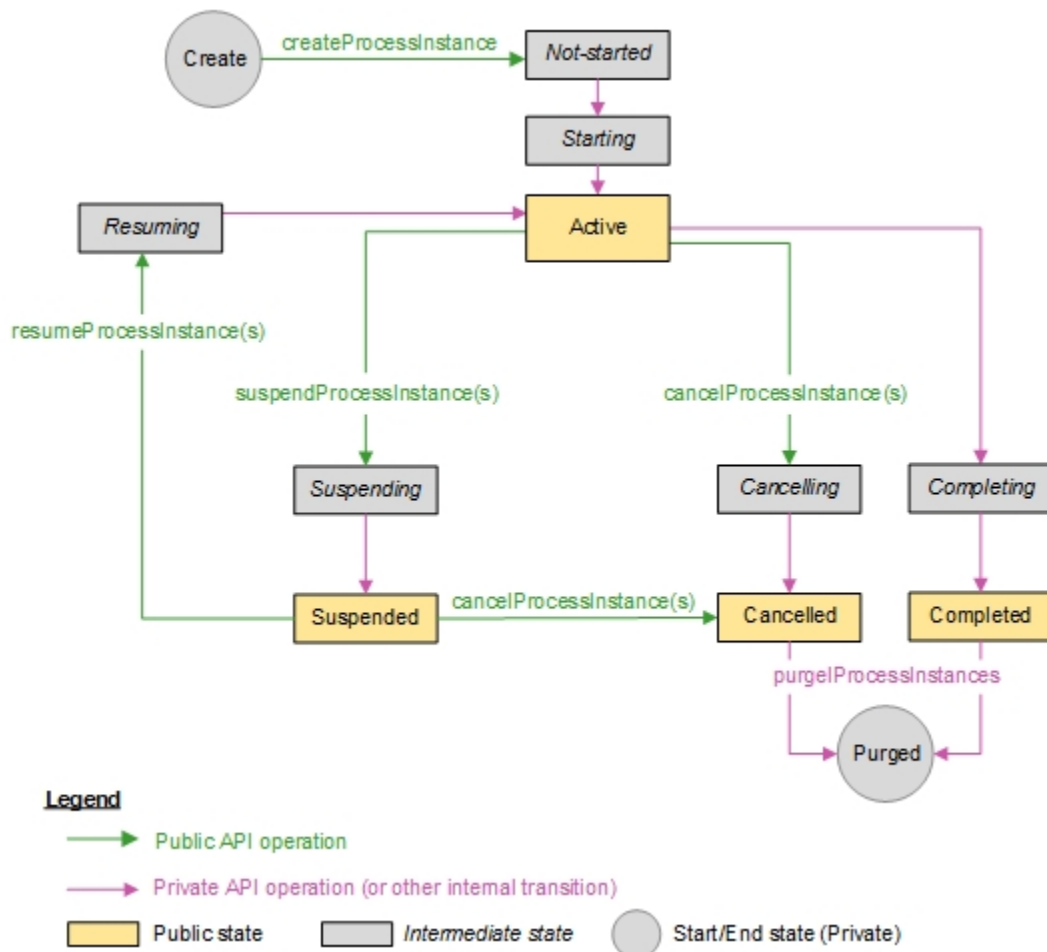
Using complex conditions:

```
SELECT * FROM process WHERE (DEFINITION.NAME = 'Process1' OR
DEFINITION.NAME = 'Process2') ORDER BY name DESC;
SELECT * FROM process WHERE (DEFINITION.NAME = 'Process1' OR
DEFINITION. NAME = 'Process2') ORDER BY DEFINITION. NAME ASC;
SELECT * FROM process WHERE (INSTANCE.ID > 20 AND MODULE.NAME
= 'ModuleA' AND order_amount > 10000) ORDER BY INSTANCE.ID DESC;
SELECT * FROM process WHERE (INSTANCE.ID > 20 AND INSTANCE.ID
< 100 ORDER BY INSTANCE.ID DESC;
SELECT * FROM process WHERE (MODULE.NAME like 'module%');
SELECT * FROM process WHERE INSTANCE.START_DATE BETWEEN TS
'2008-06-20T10:30:20Z' AND TS '2008-06-20T10:52:20Z';
SELECT MODULE.NAME, INSTANCE.VERSION, INSTANCE.NAME,
INSTANCE.DESCRPTION FROM process WHERE (MODULE.NAME
= 'module_1' OR MODULE.NAME = 'module_2') ORDER BY MODULE.NAME
DESC;
SELECT * FROM process WHERE (isBalanceZero = TRUE) ORDER BY
INSTANCE.ID DESC;
```

## Process Instance State Transitions

Processes go through a series of state transactions.

The following diagram shows the different states that a (non-failing) process instance may be in, the possible transitions between states and the [ProcessManagementService](#) functions that can be used to trigger these transitions.



(For information about state transitions involving failing processes, see [Handling Process Instance Failures](#).)

The following table lists the [ProcessManagementService](#) functions that can be used to change the state of (non-failing) process instances:

- **Initial State** is the state that a process instance must be in to allow the indicated function to be used.
- **End State** is the state to which the process instance will be transitioned on execution of the function.

These are the only state transitions that can be performed using the public API. Any state transitions not shown in this table can only occur as a result of internal operations.

Initial State	End State		
	Active	Suspended	Cancelled
<b>Active</b>	<i>None</i>	<a href="#">suspendProcessInstance</a> <a href="#">suspendProcessInstances</a>	<a href="#">cancelProcessInstance</a> <a href="#">cancelProcessInstances</a>
<b>Suspended</b>	<a href="#">resumeProcessInstance</a> <a href="#">resumeProcessInstances</a>	<i>None</i>	<a href="#">cancelProcessInstance</a> <a href="#">cancelProcessInstances</a>





There are similarly-named singular and plural versions of these functions:

- Use the singular-named version of the function (for example, [suspendProcessInstance](#)) to change the state of a single, specified process instance.
- Use the plural-named version of the function (for example, [suspendProcessInstances](#)) to change the state of all process instances for a specified process template.

## Handling Process Instance Failures

A process instance may fail if an executing activity encounters an unexpected error condition (that is, one that is not caught and handled by the process itself).

For example:

- a database connection failure occurs while a database service task is performing an update.
- a system out-of-memory error occurs while a script task is running.

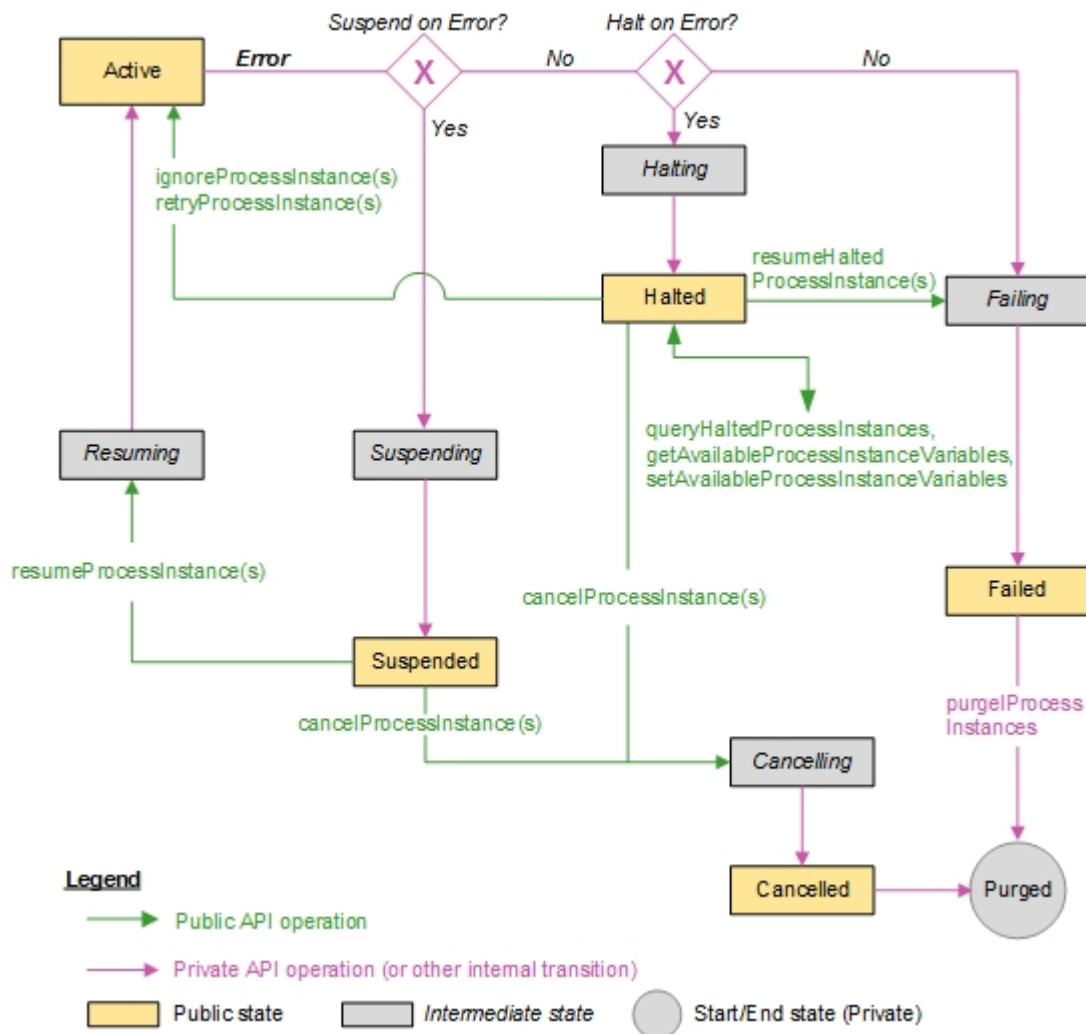
When a process instance activity is executing, if a system error causes the activity to throw a Java exception, Process Manager places the process instance in a SUSPENDED, HALTED or FAILED state, depending on:

1. the error handling configuration that has been applied at the activity, process and/or system-wide levels. The BPM runtime supports two types of error handling -- `suspendOnError` and `haltOnError`.
2. the version of TIBCO Business Studio that was used to deploy the process application:
  - `suspendOnError` is only supported by a process application that was deployed from a pre-3.5.10 version of TIBCO Business Studio.
  - `haltOnError` is only supported by a process application that was deployed from TIBCO Business Studio version 3.5.10.



For more information about `haltOnError` and `suspendOnError` and how they are configured, see "Configuring Error Handling Behavior for Process Instances" in *TIBCO ActiveMatrix BPM Administration*.

The following diagram shows the different states that a failing process instance may enter, the possible subsequent transitions between states and the [ProcessManagementService](#) functions that can be used to trigger these transitions.



The following table lists the [ProcessManagementService](#) functions that can be used to change the state of halted process instances. **End State** is the state to which the process instance will be transitioned on execution of the function.

These are the only state transitions that can be performed using the BPM public API. Any state transitions not shown in this table can only occur as a result of internal operations.

Operation	Description	End State
<a href="#">queryHaltedProcessInstances</a>	List halted process instances that match certain criteria.	Halted
<a href="#">getAvailableProcessInstanceVariables</a> <a href="#">setAvailableProcessInstanceVariables</a>	Get/set details of variables available to the failed activity for a particular halted process instance.	Halted
<a href="#">ignoreProcessInstance</a> <a href="#">ignoreProcessInstances</a>	Resume execution of a halted process instance, ignoring the failed activity.	Active
<a href="#">retryProcessInstance</a> <a href="#">retryProcessInstances</a>	Resume execution of a halted process instance, retrying the failed activity.	Active

Operation	Description	End State
<a href="#">cancelProcessInstance</a> <a href="#">cancelProcessInstances</a>	Cancel execution of a halted process instance.	Cancelled
<a href="#">resumeHaltedProcessInstance</a> <a href="#">resumeHaltedProcessInstances</a>	Resume execution of a halted process instance as if had not halted. The instance therefore immediately enters a FAILED state.	Failed



Where there are similarly-named singular and plural versions of these functions:

- Use the singular-named version of the function (for example, [retryProcessInstance](#)) to change the state of a single, specified process instance.
- Use the plural-named version of the function (for example, [retryProcessInstances](#)) to change the state of all process instances for a specified process template.

## Migrating a Process Instance to a Different Version

Process migration is the ability to migrate a long running process instance to a different version of the process template from which it was generated.

For a general overview of process migration, see "Process Migration" in *TIBCO ActiveMatrix BPM Concepts*.

## Migration Points

A *migration point* is a task in the process template at which a process instance can be migrated to a different version.

Not all tasks are valid migration points. For example:

- tasks that are on a parallel path, or tasks that may have parallel executions due to multiple tokens flowing on a single path
- tasks in an embedded sub-process
- start events or boundary events
- un-named tasks. (The task name is used to identify a migration point. A well-formed process should therefore not use duplicate task names.)

Valid migration points are automatically identified (and marked internally in the BPEL file) by TIBCO Business Studio at design time.



There are restrictions on the types of change that can be made between the two versions (source and destination) of a migrated process template, as follows:

- In a normal flow (not a cycle), a task that precedes the migration point in the source version should not be rearranged so that it follows the migration point in the destination version.
- A task that is concurrent with the migration point in the source version should not be rearranged so that it follows the migration point in the destination version.
- If compensation is triggered in the destination version for a task that completed in the source version, the compensation defined in the *source* version will be performed.

## Migration Rules

A *migration rule* defines when, how and to what version a process instance will migrate.

A migration rule identifies:

- a *qualified task name* (module name, module version, process template name and task name) from which a process instance will migrate.
- the module/process template version to which the process instance will migrate.

### How Process Instances Are Migrated

At runtime, when a process instance task that is defined as a migration point executes, Process Manager checks if a migration rule is set (using the qualified task name).

If a migration rule is set, Process Manager:

1. migrates the process instance to the new version of the process template defined in the migration rule.
2. issues a `BX_INSTANCE_PROCESS_MIGRATED` audit event to indicate that the process instance has migrated.



The `managedObjectVersion` for this event is the process template version *after* migration.

The `managedObjectId` for this event is the ID of the process instance. This remains the same after migration as it was before, and so it can be used to connect audit events before and after migration.

3. continues execution of the task using the migrated to version of the process instance.

### Process Migration Functions

Functions are provided to control process migration.

These [ProcessManagementService](#) functions are:

- [listMigrationRules](#) - List the process instance migration rules that are set for one or more process templates.
- [isSetMigrationRule](#) - Test whether any migration rules are set for a particular qualified task name.
- [getMigrationPoints](#) - List the permissible process instance migration points for one or more process templates.
- [setMigrationRules](#) - Set one or more process instance migration rules.
- [unsetMigrationRules](#) - Unset one or more previously set process instance migration rules.
- [clearMigrationRules](#) - Clear all process instance migration rules for one or more process templates.

### Authorization of Process Migration Functions

A resource can only perform process migration if they have the necessary privileges to execute the **handleProcessMigration** system action. (The system-wide default value for this system action is Denied.)

Process Manager validates the calling resource's privileges whenever a process migration service function is called. If the calling resource does not have the necessary authorization to execute the **handleProcessMigration** system action, the call will fail with an appropriate error.



You can use the [SecurityService](#) functions (for example, [IsActionAuthorized](#)) to test whether a resource has the necessary authorization before calling a process migration service function.

### Business Services

A business service is a set of actions that accomplishes some sort of business function. For example, a business service could be designed to handle an incoming insurance claim.

A business service may consist of the following actions:

- When the business service is started, a form is displayed that allows you to enter the claimant's policy number.
- When the form with the policy number is submitted, a database is accessed to retrieve the claimant's policy information.
- Another form is displayed that contains the policy information.
- After reviewing the policy information, and deciding it is a valid claim, submitting the form may start an instance of a process that is used to process the claim.
- The process instance that is started causes a work item to be sent to the appropriate user, who must open it and work on it.

Note that business services are *stateless*, meaning that if it consists of a number of forms, and you enter data into some forms, then cancel the business service (by clicking the **Cancel** button on a form) before completing the business service, none of the data you entered on the previous forms is saved.

If a business performs a *stateful* action, that is, something that cannot be reversed (e.g., writing to a database, starting a process instance, etc.), typically it will be the last action performed by the business service.

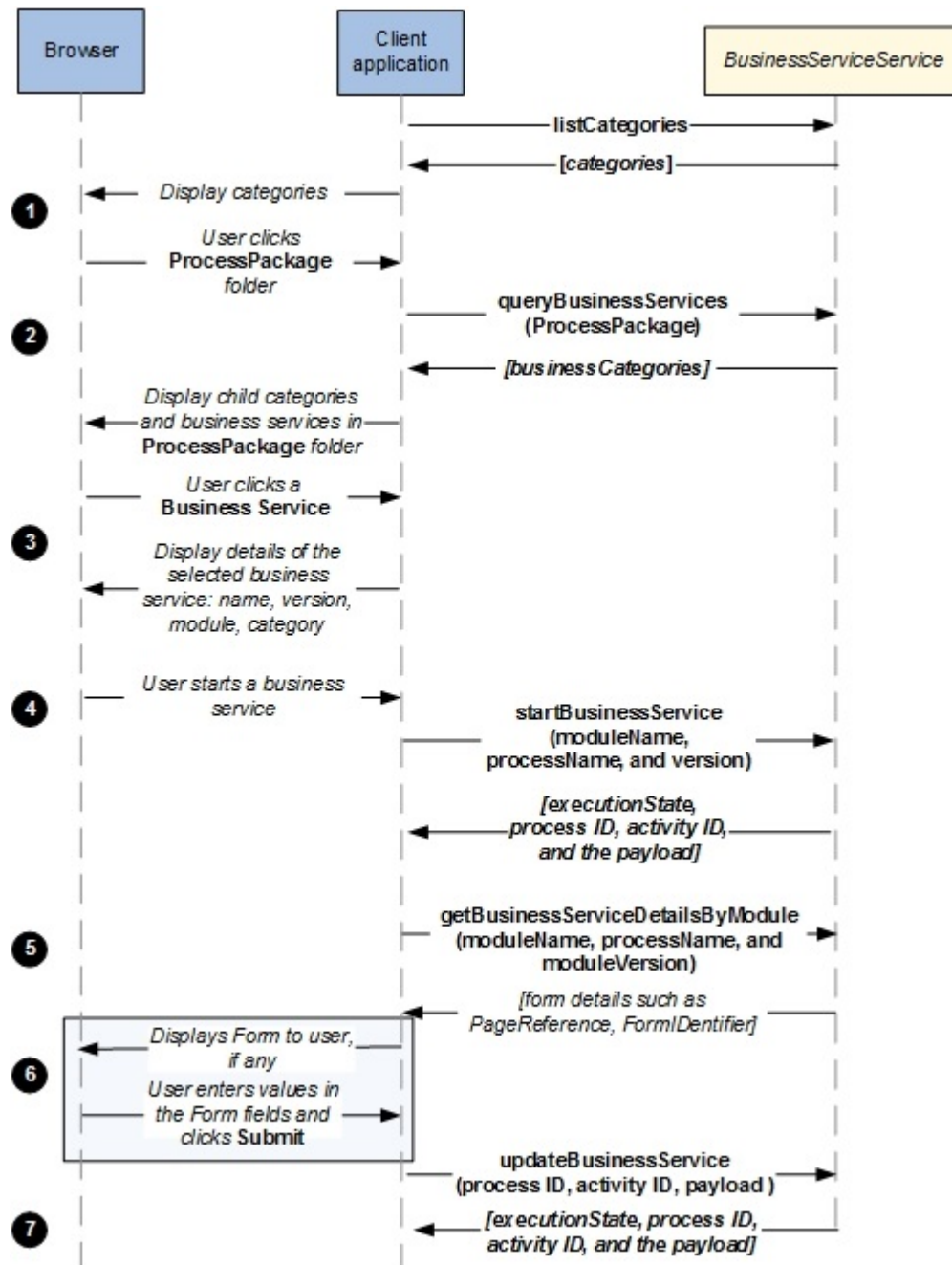
The objectAPI contains a [BusinessServiceService](#) to perform business service-related functions.

## Starting a Business Service

Starting a business service involves listing all the available business services and then selecting a business service to start.

The following example shows how calls to the objectAPI can be used to start and update a business service.

## Starting a Business Service



### Procedure

1. When you login to an application, [listCategories](#) is called and all the available categories are listed in the **Business Services** tab.
2. When the user clicks on a category, [queryBusinessServices](#) is called and all the business services under the selected category are displayed. The response contains the details of all the business services under the selected category.
3. Selecting a business service displays the details of that business service on the client, which are obtained from the response message of the [queryBusinessServices](#) call.
4. When the user clicks the button to start a new instance of the business service, it calls [startBusinessService](#) and passes the module name, process name, and version number. The values for these parameters can be obtained by calling [queryBusinessServices](#) or [listBusinessServices](#).

The response returns the execution state, process ID, activity ID, and the payload.

5. If the business service has a form associated with it, [getBusinessServiceDetailsByModule](#) is called. The response contains the form details, which are then passed on to the Forms Adapter.
6. The Forms Adapter parses the data and displays an appropriate Form to the end user. Similarly, any input provided in the form is processed by the Form Adapter and handed back to the servlet.

For details about how the data is passed to, and received from, the Forms Adapter, see [Forms](#).



For details about the TIBCO Forms APIs, see the *TIBCO Business Studio Forms User's Guide*.

7. Once the user enters values in the form fields and clicks **Submit**, [updateBusinessService](#) is called with the payload containing the values entered by the user. Repeat steps [Step 5](#) and [Step 6](#) until the business service completes or is cancelled.

The response returns the execution state, process ID, activity ID, and the payload. If the business service is complete, the execution state is COMPLETED.

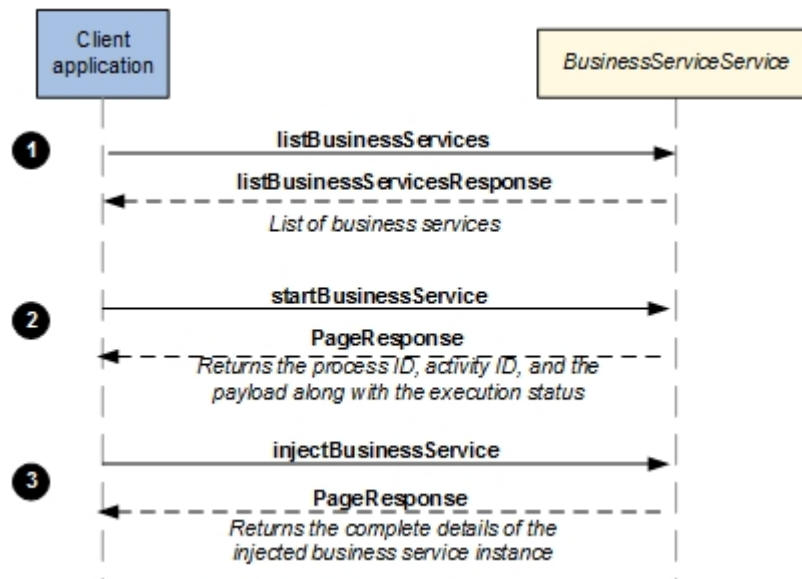
## Injecting Data into a Business Service

You can inject data into a business service multiple times while it is in progress, regardless of whether the flow from any previous occurrence is still in progress.

You can use an event handler to do this. An event handler is a catch intermediate event with no specific trigger and incoming flow that can be triggered by an external client application using the objectAPI. Note that the event handler does not have to happen for the business service to complete.

The following example illustrates injecting data into a business service. It involves listing all the deployed business services, starting a business service, and while it is running, injecting data.

### Injecting Data into a Business Service



### Procedure

1. Find out the list of deployed business services by calling [listBusinessServices](#).

This function returns the module version number, which is required as an input when calling [startBusinessService](#).

You can choose to start one of the business services returned.

2. Call [startBusinessService](#) and specify the **module name** and **process name** to start the selected business service.



This returns the instance ID that is required for the next step in which data is injected into the business service.

3. Call [injectBusinessService](#).

This injects the provided DataPayload into the formal parameters associated with the business service.

## Pageflows

A pageflow process is a short-lived process designed to display user interface pages to the user in sequence. It can also include other activities -- such as web service or database calls -- which can be used to drive the interaction with the user.

Unlike a normal business process, a pageflow process is *stateless* and *non-transactional*. If the process is not completed in full, any data that is set earlier in the process is lost. If a pageflow process performs a *stateful* action -- something that cannot be reversed, such as writing to a database or starting a process instance -- it should be the final action performed by the pageflow process.

## Starting and Processing a Pageflow

Step-by-step descriptions of the calls that a client application needs to make to start and process a pageflow, including displaying a form, are provided.

For this information, see [Displaying a Form in a Pageflow](#).

Also note that you can configure the Pageflow Engine for optimum pageflow performance; for more information, see the `PFEConfig.properties` file in the "BPM Properties Files" section of the *TIBCO ActiveMatrix BPM - BPM Administration* guide.

## Injecting Data into a Pageflow

You can inject data into a pageflow process multiple times while it is in progress, regardless of whether the flow from any previous occurrence is still in progress.

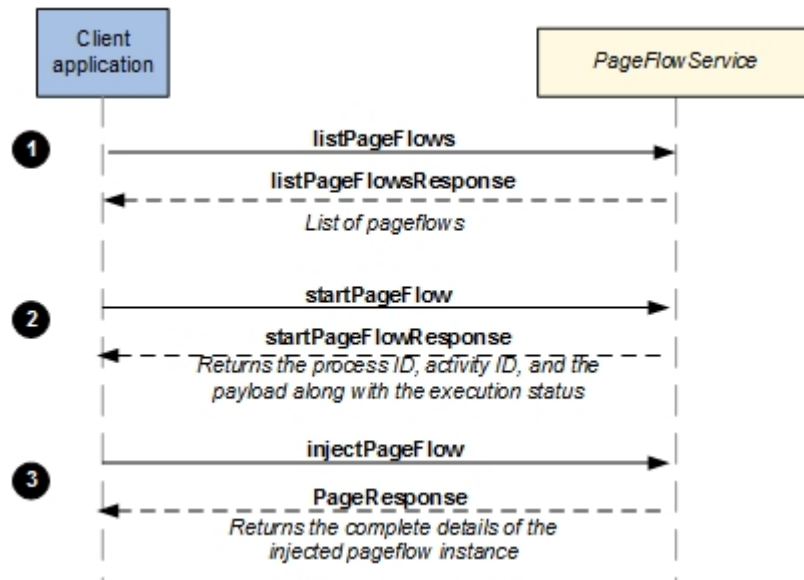
You can use an event handler to do this. An event handler is a catch intermediate event with no specific trigger and incoming flow that can be triggered by an external client application using the objectAPI. Note that the event handler does not have to happen for the pageflow process to complete.

You can update a pageflow local data cache using an event handler. For example, an event handler updates the exchange rate information regularly. A pageflow process used to manage an order request can use the updated exchange rate information when issuing an invoice.

The following example illustrates injecting data into a pageflow. It involves listing all the deployed pageflows, starting a pageflow, and while it is running, injecting data using an intermediate event.



### Injecting a Pageflow Event



#### Procedure

1. Find out the list of deployed pageflows by calling [listPageFlows](#).  
This function returns the module version number, which is required as an input when calling the [startPageFlow](#) function.  
You can choose to start one of the pageflows returned.
2. Call [startPageFlow](#) and specify the parameters **module name** and **process name** to start the selected pageflow.  
This returns the instance ID, which is required for the next step in which data is injected into the pageflow.
3. Call the [injectPageFlowEvent](#) function.  
This injects the provided DataPayload into the formal parameters associated with the pageflow.

### Work Lists and Work List Views

The terms *work lists* and *work list views* are often used interchangeably. Both are a list of work items.

A *work list* is typically referring to the list of work items that have been offered or allocated to a particular organizational entity. Whereas, a *work list view* is referring to a list of work items that has been filtered and sorted in some way.

#### Work Lists

Work lists represent lists of work items assigned to an organizational entity. Users see a list of the work items that they can view and work on.

The [WorkListService](#) provides functions to manage work lists in your application. These functions can be used to:

- Retrieve work items from a work list .
- Sort the work items in a work list into a desired order, or to filter the work items so only the desired work items are displayed in the list.

## Retrieving a Work List

Retrieving a work list for an organizational entity, such as a position, a group, or a resource, causes the work items that have been offered or allocated to the organizational entity to be displayed.

The [WorkListService](#) provides the following functions to retrieve work lists:

- [getWorkListItems](#) - Returns the work list for the specified organizational entity.
- [getWorkListItemsAllResources](#) - Returns the work list for all resources.
- [getAllocatedWorkListItems](#) - Returns a list of work items that are allocated to the specified organizational entity.

## Sorting and Filtering Work Lists

Work lists can be sorted and filtered so that they list only the desired work items, in the desired order.

The [WorkListService](#) contains the following functions to get and set sort and filter criteria:

- [getWorkItemOrderFilter](#) - Get the fields that can be used to define sort and filter criteria for a work list.
- [setResourceOrderFilterCriteria](#) - Sets sort and filter criteria for work lists for a specified resource.
- [getResourceOrderFilterCriteria](#) - Gets sort and filter criteria, which was previously set with the [setResourceOrderFilterCriteria](#) function, for work lists for a specified resource.

Note, however, sort and filter criteria can also be specified when retrieving work lists using the [getWorkListItems](#), [getWorkListItemsAllResources](#), and [getAllocatedWorkListItems](#) functions.

The syntax and criteria for sort and filter expressions is the same regardless which of the methods described above is used to set the criteria..

The default sort criteria, used to display a work list if no other sort is specified, are defined by the `defaultSort` property in the `brm.properties` file. For information about this property, see "Configuring TIBCO ActiveMatrix BPM Resource Management" in *TIBCO ActiveMatrix BPM - BPM Administration*.

## Filtering Performance Considerations

Some installations that make heavy use of filtering in TIBCO ActiveMatrix BPM may benefit from either tuning of existing database indexes or the creation of new indexes. Work items can be filtered and sorted on any number of columns that are present on the `brm_work_items` table. On a standard installation, only the first four attributes are indexed. If work list sorting and filtering is being applied to columns that aren't indexed, for example, listing all work items that have **attribute19** greater than a specified date, then both filter and overall system performance may benefit if the database administrator adds an index to the **attribute19** column.

## Sort and Filter Expression Syntax

The syntax of sort/filter expressions is similar to that used in Java expressions.

Expressions are made up of a set of terms that can be parenthetically separated using the `(` and `)` characters, and can be joined using the **AND** and **OR** operators. Delimiters are used as follows:

- Text string fields are delimited using the `'` character.
- Expressions denoting date/time ranges can use different delimiters with different meanings, as described in [Using Date Time Ranges](#).
- Other fields require no delimiters.
- You can include `Attributex = NULL//Empty` or `Attributex != NULL//Empty` in query strings to show attributes which have not been set.

Some examples of filter expressions are:

```
(priority = 1) OR ((priority < 12) AND (priority > 4)) OR (description='fred')
(startDate>=2010-05-23T14:25:10) OR description='a_b*a' AND (priority<>13) OR
(description='1?3*')
(priority=35) OR description='a_b*a' AND (priority<>13) OR (description = '1?3*')
visible=TRUE OR ((startDate>2010-05-23T14:25:10) AND (distributionStrategy=OFFERED))
[2010-05-23,2006-05-23T14:25:10.123Z]
(startDate <= 2006-05-23T14:25:10.123Z) OR endDate<2010-05-23
priority=35 AND [2010-05-23T14:25:10,2006-05-23T14:25:10]
attribute1=5346 OR attribute4='Urgent'
attribute4='NULL'
```

### Wildcard Characters

Wildcard characters can be used in filter expressions.

The following wildcard characters can be used:

- ? Matches any single character.
- \* Matches any string of zero or more characters.

If you want to search for a string that actually contains these characters, you must escape them using \. For example, the string "fr?d" (with the ? character escaped) results in a query using "fr?d".

### Unsupported Characters

You cannot filter or sort on an expression containing a single quote character, and you cannot escape the quote character using \. For example, filtering on the following expression returns an error:

```
(priority = 1) OR ((priority < 12) AND (priority > 4)) OR (description='fred
o'shea')
```

To filter on a string like this, use a wildcard character, such as 'fred o\*shea' or 'fred o?shea'.

### Sort Expressions

Sort order can be defined using the **ASC** (ascending sort) or **DESC** (descending sort) keywords.

Multiple sort criteria can be used in a single expression. Individual criteria must be comma-separated. For example:

```
id ASC, priority ASC, startDate ASC, endDate DESC
```

### Sort Filter Criteria

The table below describes the criteria that can be used to filter and/or sort work items in a user's work list.



Organizations that make heavy use of filtering may benefit from either tuning of existing database indexes, or the creation of new indexes. Work items can be filtered and sorted on any number of columns that are present on the `brm_work_items` table. On a standard installation, only the first four attributes are indexed. If work list sorting and filtering is being applied to columns that aren't indexed, for example, listing all work items that have **attribute19** greater than a specified date, then both filter and overall system performance may benefit if the database administrator adds an index to the **attribute19** column.

Field ID in API	Name	Type	Operators	Usage	Description
1	id	Numeric	= ◇ > < >= <=	Sort and filter expressions.	ID of the work item.  Can be any number specified with or without a decimal point.
2	name	String	= ◇ > < >= <=	Filter expressions only.  Cannot be used in sort expressions.	The name of the work item.  Can be any text. Wildcards are specified using ? and *, single and multiple character matching.
3	description	String	= ◇ > < >= <=	Filter expressions only.  Cannot be used in sort expressions.	Description of the work item.  Can be any text. Wildcards are specified using ? and *, single and multiple character matching.
4	priority	Numeric	= ◇ > < >= <=	Sort and filter expressions.	Priority of the work item.  Can be any number, with or without a decimal point.
5	startDate	Date/Time	= ◇ > < >= <=	Sort and filter expressions.	Earliest date at which the work item can be started.  See <a href="#">Using DateTime Fields</a> for details of the format.
6	endDate	Date/Time	= ◇ > < >= <=	Sort and filter expressions.	Date by which the work item should be completed.  See <a href="#">Using DateTime Fields</a> for details of the format.
7	distributionStrategy	Enum (Dist. strategy)	= ◇	Sort and filter expressions.	Whether the item was offered or allocated when it was scheduled. Must be either ALLOCATED or OFFERED.

Field ID in API	Name	Type	Operators	Usage	Description
8	visible	Boolean	= ◇	Filter expressions only.  Cannot be used in sort expressions.	Whether or not the item is currently visible.  Must be either TRUE or FALSE.
9	state	Enum (State)	= ◇	Filter expressions only.  Cannot be used in sort expressions.	Current state of the work item.  Must be one of: <ul style="list-style-type: none"> <li>• ALLOCATED</li> <li>• OFFERED</li> <li>• CREATED</li> <li>• OPENED</li> <li>• PENDED</li> <li>• PENDHIDDEN</li> <li>• CANCELLED</li> <li>• SUSPENDED</li> </ul>
10	appInstance	String	= ◇ > < >= <=	Sort and filter expressions.	The id of the application instance that created the work item.  Can be any text. Wildcards are specified using ? and *, single and multiple character matching.
11	appName	String	= ◇ > < >= <=	Sort and filter expressions.	The name of the application instance that created the work item.  Can be any text. Wildcards are specified using ? and *, single and multiple character matching.

Field ID in API	Name	Type	Operators	Usage	Description
12	appInstance Description	String	= <> > < >= <=	Sort and filter expressions.	The description of the application instance that created the work item.  Can be any text. Wildcards are specified using ? and *, single and multiple character matching.
13	attribute1	Numeric (Integer)	= <> > < >= <=	Sort and filter expressions	The integer value of the first custom work item attribute defined. See <a href="#">Using Work Item Attribute Fields</a> .  Must be an integer.
14-16	attribute2 - attribute4	String	= <> > < >= <=	Sort and filter expressions.	The value of the second to fourth custom work item attributes defined. See <a href="#">Using Work Item Attribute Fields</a> .  Can be any text up to a maximum of 64 characters. Longer strings will be truncated to 64. Wildcards are specified using ? and *, single and multiple character matching.
17	attribute5	Numeric	= <> > < >= <=	Sort and filter expressions.	The decimal value of the fifth custom work item attribute defined. See <a href="#">Using Work Item Attribute Fields</a> .  Must be a Decimal.  In filter expressions, you can specify values in exponential notation. For example:  =1.234E+11 =1.234e-1 =-.234e-2

Field ID in API	Name	Type	Operators	Usage	Description
18-19	attribute6 - attribute7	Date/Time	= <> > < >= <=	Sort and filter expressions.	The value of the sixth to seventh custom work item attributes defined. See <a href="#">Using Work Item Attribute Fields</a> .  See <a href="#">Using DateTime Fields</a> for details of the format.
20-27	attribute8 - attribute14	String	= <> > < >= <=	Sort and filter expressions.	The value of the eighth to fourteenth custom work item attributes defined. See <a href="#">Using Work Item Attribute Fields</a> .  Can be any text up to a maximum of 20 characters. Longer strings will be truncated to 20. Wildcards are specified using ? and *, single and multiple character matching.
28	attribute15	Numeric (Integer)	= <> > < >= <=	Sort and filter expressions	The integer value of the fifteenth custom work item attribute defined. See <a href="#">Using Work Item Attribute Fields</a> .  Must be an integer.
29-31	attribute16 - attribute18	Numeric	= <> > < >= <=	Sort and filter expressions.	The decimal value of the sixteenth to eighteenth custom work item attribute defined. See <a href="#">Using Work Item Attribute Fields</a> .  Must be a Decimal.  In filter expressions, you can specify values in exponential notation. For example:  =1.234E+11 =1.234e-1 =-.234e-2

Field ID in API	Name	Type	Operators	Usage	Description
32-33	attribute19-attribute20	Date/Time	= < > < >= <=	Sort and filter expressions.	<p>The value of the nineteenth to twentieth custom work item attributes defined. See <a href="#">Using Work Item Attribute Fields</a>.</p> <p>See <a href="#">Using DateTime Fields</a> for details of the format.</p>
34-39	attribute21-attribute26	String	= < > < >= <=	Sort and filter expressions.	<p>The value of the twenty-first to twenty-sixth custom work item attributes defined. See <a href="#">Using Work Item Attribute Fields</a>.</p> <p>Can be any text up to a maximum of 20 characters. Longer strings will be truncated to 20. Wildcards are specified using ? and *, single and multiple character matching.</p>
40-51	attribute27-attribute38	String	= < > < >= <=	Sort and filter expressions.	<p>The value of the twenty-first to twenty-sixth custom work item attributes defined. See <a href="#">Using Work Item Attribute Fields</a>.</p> <p>Can be any text up to a maximum of 64 characters. Longer strings will be truncated to 64. Wildcards are specified using ? and *, single and multiple character matching.</p>



Field ID in API	Name	Type	Operators	Usage	Description
51-52	attribute39-attribute40	String	= <> > < >= <=	Sort and filter expressions.	The value of the thirty-ninth to fortieth custom work item attributes defined. See <a href="#">Using Work Item Attribute Fields</a> .  Can be any text up to a maximum of 255 characters. Longer strings will be truncated to 255. Wildcards are specified using ? and *, single and multiple character matching.



You can also use the syntax `Attributex = NULL/Empty` or `Attributex != Null/Empty` in query strings.

#### Using DateTime Fields

Date/Time fields (**startDate**, **endDate**, and **attribute6 - attribute7**) can appear any number of times within a filter expression.

No wildcards are supported.

If no hours, minutes, seconds or milliseconds are specified then the field adds the time and milliseconds as required by the operations being performed.

The following formats for date/times are allowed:

- yyyy
- yyyy-MM
- yyyy-MM-dd
- yyyy-MM-ddTHH
- yyyy-MM-ddTHH:mm
- yyyy-MM-ddTHH:mm:ss
- yyyy-MM-ddTHH:mm:ss.SSS
- yyyyTD
- yyyy-MMTD
- yyyy-MM-ddTD
- yyyy-MM-ddTHHD
- yyyy-MM-ddTHH:mmD
- yyyy-MM-ddTHH:mm:ssD
- yyyy-MM-ddTHH:mm:ss.SSSD

where *D* can be either of:

- Z
- <+|->HHmm | HH:mm | HH>

For example:

```
2010-03-03T23:45+01
2010-03-03T23:45+01:30
2010-03TZ
2010T-0100
```

The operators given in the table are used as follows:

`<startDate | endDate> <operator> <date/time>`

For example:

- `startDate >= 2010-05-23T14:25:10`
- `endDate <= 2010-05-23T14:25:10Z`
- `startDate > 2010-05-23T14:25:10.123`
- `endDate < 2010-05-23T14:25:10.123Z`

### Using DateTime Ranges

You can specify a query for a date/time range, and use it in filter expressions (but not in sort expressions) to query a range of **endDate**s, a range of **startDate**s, or a **startDate** to **endDate** period.

Different separator characters have the following effects:

Separator	Specifies that...
[	The start of the range is exclusive.
]	The end of the range is exclusive.
{	The start of the range is inclusive.
}	The end of the range is inclusive.
,	<b>startDate</b> is specified first, <b>endDate</b> next.
^	The whole range is <b>endDate</b> values.
!	The whole range is <b>startDate</b> values.

All the date formats supported by **startDate** and **endDate** (see [Sort Filter Criteria](#)) are supported.

The following examples all use the date format **yyyy-MM-dd**:

Filter Expression	Description
<code>[2010-05-23,2010-07-23]</code>	<code>(start_date&gt;'2010-05-23T23:59:59.997' AND end_date&lt;'2010-07-23T00:00:00.000')</code>
<code>{2010-05-23,2010-07-23}</code>	<code>(start_date&gt;='2010-05-23T00:00:00.000' AND end_date&lt;='2010-07-23T23:59:59.997')</code>
<code>[2010-05-23,2010-07-23}</code>	<code>(start_date&gt;'2010-05-23T23:59:59.997' AND end_date&lt;='2010-07-23T23:59:59.997')</code>
<code>{2010-05-23,2010-07-23]</code>	<code>(start_date&gt;='2010-05-23T00:00:00.000' AND end_date&lt;'2010-07-23T00:00:00.000')</code>

Filter Expression	Description
[2010-05-23^2010-07-23]	(end_date>'2010-05-23T23:59:59.997' AND end_date<'2010-07-23T00:00:00.000')
{2010-05-23!2010-07-23}	(start_date>='2010-05-23T00:00:00.000' AND start_date<='2010-07-23T23:59:59.997')

### Using Work Item Attribute Fields

You can use the values of the 40 custom work item attribute fields as criteria in both sort and filter expressions.

These fields correspond to the customizable attributes **workItemAttributes.attribute1** to **workItemAttributes.attribute40** that you can set using the work item scripting methods described in the *TIBCO ActiveMatrix BPM Business Data Services Guide* ("Process Manager and Work Manager Scripting" > "WorkItem" topic). Also, for information about defining scripts in your processes at design time, see the "Using Scripts" topic in the *TIBCO Business Studio BPM Implementation* guide.

By default, these custom fields are locally logged, but are neither audited nor published by Event Collector. However, you can configure the system to audit and/or to publish any of these attributes for any event. See "Configuring TIBCO ActiveMatrix BPM Auditing" in *TIBCO ActiveMatrix BPM - BPM Administration* for information on changing the default auditing and logging behavior.

You can use these attributes to hold data associated with the work item. For example, you could use them to track data about the customer with whom the work item is associated. Assume that a script in your business process sets the following values, mapping fields from the business process to work item attributes:

```
WorkManagerFactory.getWorkItem().workItemAttributes.attribute1=customer_number
WorkManagerFactory.getWorkItem().workItemAttributes.attribute2=customer_name
WorkManagerFactory.getWorkItem().workItemAttributes.attribute3=customer_contact
```

You could then use a [setResourceOrderFilterCriteria](#) function to sort or filter by any of these values.

### Work List Views

Work list views are lists of work items sorted and filtered as desired.

You can create work list views of your own work list or for a specific organizational entity.

You can also grant specific resources and organizational model entities access to your work list views. This means work list views can be shared by many resources in the organization. Any changes in your organizational model are automatically applied to your work list views. For example, if an organizational entity is removed, any work list views associated with that entity are removed. Secondly, if an entity is defined as the user of a work list view and the entity is deleted, the work list view remains but the entity is removed from its list of users.

The following [WorkListService](#) functions are available to work with work list views:

- [createWorkListView](#) - Creates a new work view.
- [getWorkListViewDetails](#) - Retrieves the details of a specific work view.
- [editWorkListView](#) - Edits an existing work view.
- [deleteWorkListView](#) - Deletes an existing work list view.
- [unlockWorkListView](#) - Unlocks a locked work list view (which was locked with [getWorkListViewDetails](#)).
- [getEditableWorkListViews](#) - Retrieves a list of work views that the calling resource can edit.
- [addCurrentResourceToView](#) - Adds the current resource to a work view.
- [deleteCurrentResourceFromView](#) - Removes a public work view from the work view list for the calling resource.

- [getPublicWorkListViews](#) - Retrieves a list of all public work views.
- [getViewsForResource](#) - Retrieves a list of work views to which the calling resource has access.
- [getWorkListItemsForView](#) - Retrieves the work item list for a specific work view.

## Creating a Work List View

You can create a work list view for a variety of entities.

For example:

- a specific organizational entity, of type organization unit, position, group, or resource.
- all users (that is, every existing organizational entity of type resource).
- the user executing the [CreateWorkListView](#) call.

You must have the **viewWorkList** system action for the target entities of the work list. Similarly, if you want to grant another user access to a work list view for another entity in an organization model then that user must have the **viewWorkList** system action for the work list you require them to have access to.

Note that:

- When creating a work view of another organization unit, group or position, only offered work items are displayed (unless the **getAllocatedItems** parameter is set to **true**, either on the work view itself or overridden when accessing the work view). These are work items that were offered to the selected organizational entity, and that still have a state of Offered. For resources, both Offered and Allocated are displayed.
- When viewing a work view for an organizational entity, you can only see work items that are offered directly to that entity. You cannot see the work items for other entities in that organizational entity. For example, if you have created a work view for Group1 and Group1A is also a member of Group1, you must create a separate work view for Group1A. (You must also have the **viewWorkList** system action for Group1A).
- If a work view is for another organizational entity's work list, and not your own work list, you cannot open or complete work items in that view. This is because the system always requires a user to process work items in their own work list.
- It could also be the case that you can view a work list but are not a member of the organization entity whose work list you are viewing. In this case, you cannot open work items in that view. For example, if you have the **viewWorkList** system action for a group called Group1, and you create a work view for Group1, you will be able to see the work items in Group1's work list, but you will not be able to open them.

If you want to create a work view for an organizational entity, you must specify the organizational model entity type when calling [CreateWorkListView](#).

## Custom Data

When creating a work list view, the **customData** parameter can be used to include data that you want to store with your work list view.

The system does not look at, nor use, the data included in the **customData** parameter. It simply stores it with the work view definition. If storing XML data, the value should be wrapped in a CDATA section.

## Sorting and Filtering Work List Views

You can define whether any sort and/or filter criteria should be applied to the work list view using **orderfiltercriteria**.

For information about syntax and criteria for sort/filter expressions, see [Sort and Filter Expression Syntax](#).

The default sort criteria is used to display a work list, as defined by the **defaultSort** property in the `brm.properties` file, if no other sort is specified. For information about this property, see "Configuring TIBCO ActiveMatrix BPM Resource Management" in *TIBCO ActiveMatrix BPM - BPM Administration*.

## Work List View Access

You can grant access to your work list views.

You can make your work views available to, either:

- individual users
- specific organizational entities. The hierarchy between different organizational entities is described in "The Organization Model" in *TIBCO ActiveMatrix BPM - BPM Concepts Guide*.

To access work views of other organizational entities, users must have authorization for the **viewWorkList** system action.

To specify access to a work list view, use [createWorkListView](#) and [editWorkListView](#). The following table describes the types of users and the permissions they have.

Description	Permissions
Owner	The user who created the work list view. The Owner can view, edit, and delete a work list view. The Owner can specify authors and users of the view.
Authors	Authors can view, edit, and delete the work list view. Authors can also specify users of the work list view. If you have been specified as an Author of a work list view, the work list view is not displayed in your work list view list by default. You must load the view using <a href="#">getEditableWorkListViews</a> .
Users	Users can only view the work list view. A user cannot specify any access to the work list view. If you have been specified as a User of a work list view, the view is automatically displayed in your work list view list. If you are a user of a work list view, you cannot remove it from your work list view list. Only an Owner or Author can amend access to, or delete, the view.

## Editing Work List Views

You can edit an existing work list view.

The [editWorkListView](#) function must specify the name and the unique entity ID of the work list view to be edited. It can also optionally specify the details of the work list view that you want to edit, for example, its description, any sort/filter criteria or the organizational model entities that can access the work view. If any of the optional elements are specified, they overwrite the existing attributes specified for this view.



You only need to pass the data you want to change. Parameters that are not passed are unchanged.

The following steps describe the functions involved when editing a work list view.

### Procedure

1. Use the [getEditableWorkListViews](#) function to retrieve a list of work views that the calling organizational entity has access to. For this function you must specify:
  - `startPosition` - the position in the list from which to start the page of results. The list is zero-based. To start at the first item, specify 0.

- `numberOfItems` - the number of items from the list to be displayed.
2. Use the [getWorkListViewDetails](#) function to lock the work list view for editing. You must lock the work list view if you want to edit it. If `lockView` is `true`, a lock record is placed against this view for the calling resource ID. The lock is only removed when the view is either updated or an explicit [deleteWorkListView](#) function is called.  
  
The [deleteWorkListView](#) function must specify the `workListViewID` for the work list view. The `workListViewID` is returned when the view is created with [createWorkListView](#).
  3. Use the [editWorkListView](#) function to edit a work list view. You must specify the `workListViewID`. Optionally, you can specify the information you want to edit for the work list view. You only need to pass the data you want to change. Parameters that are not passed are unchanged. This information is the same as when you are creating a work list view. For more information, see [Creating a Work List View](#).
  4. When you have finished editing the work list view, unlock it using [unlockWorkListView](#). To use [unlockWorkListView](#), you must specify the `workListViewID`.

## Using Work List Views

The `WorkListService` provides two functions to retrieve work views.

They are:

- [getWorkListItemsForView](#)—This function retrieves a list of work items for the calling resource for a specified work list view.
- [getViewsForResource](#)—This function retrieves a list of views you can view.

### Retrieving Work Items from a Work List View

You can retrieve the work items in a specific work list view.

Use the [getWorkListItemsForView](#) function to retrieve the work items. You must specify:

- `viewId` - for the work list view.
- `startPosition` - the position in the work list from which to start the page of results. The list is zero-based. To start at the first item, specify 0.
- `numberOfItems` - the number of items from the work list view to be displayed.
- `totalReq` - whether to include a count of the total number of work items in the work list view list. This defaults to `true`. The count is returned in the `totalItems` value in the response.

You should set `totalReq` to `false` unless you specifically need the total figure, because setting it to `true` is expensive in terms of database access time.

Note that if you pass `false` in this parameter, and there is at least one item in the list, the `totalItems` value returned in the response is `-1`. However, if you pass `false` and there are no work items in the list, the `totalItems` value in the response is `0` (zero).

### Retrieving a List of Work Views for a Resource

You can retrieve a list of work list views to which the calling resource has access, in other words, a list of work list views for which the calling resource is either an Owner or an Author.

The [getViewsForResource](#) function is used to retrieve this type of list. You must specify:

- `startPosition` - the position in the list of views from which to start the page of results. The list is zero-based. To start at the first item, specify 0 (zero).
- `numberOfItems` - the number of views to display.

## Public Work List Views

If a work list view is *public*, a resource can add themselves as a user of the work view. This means they will be able to view the work items in the work list view.

However, users cannot *edit* a work list view unless they are the *Owner* or an *Author* of the work view. You can specify the Owner and Authors of work list views when they are created (using [createWorkListView](#)) or edited (using [editWorkListView](#)).

To retrieve a list of public work list views, use:

- [getPublicWorkListViews](#)

To add yourself as a user of a work list view, use:

- [addCurrentResourceToView](#)

To remove a public work view from the work list view list for the calling resource, use:

- [deleteCurrentResourceFromView](#)

## Creating and Managing Supervised Work List Views

A user—typically a manager or supervisor—may have the ability to display work list views other than their own.

This depends on:

- the organization model entities to which the user belong (groups, positions, organization units, and organizations),
- the privileges that they hold as a result of membership of those entities,
- whether those privileges grant access to the work lists of other organization model entities, via the **viewWorkList** system action,
- whether that system action is *scoped* to apply to the specific group, organization unit, or position whose work list is to be supervised.



For an overview of system actions, see "System Actions" in *TIBCO ActiveMatrix BPM Concepts*.

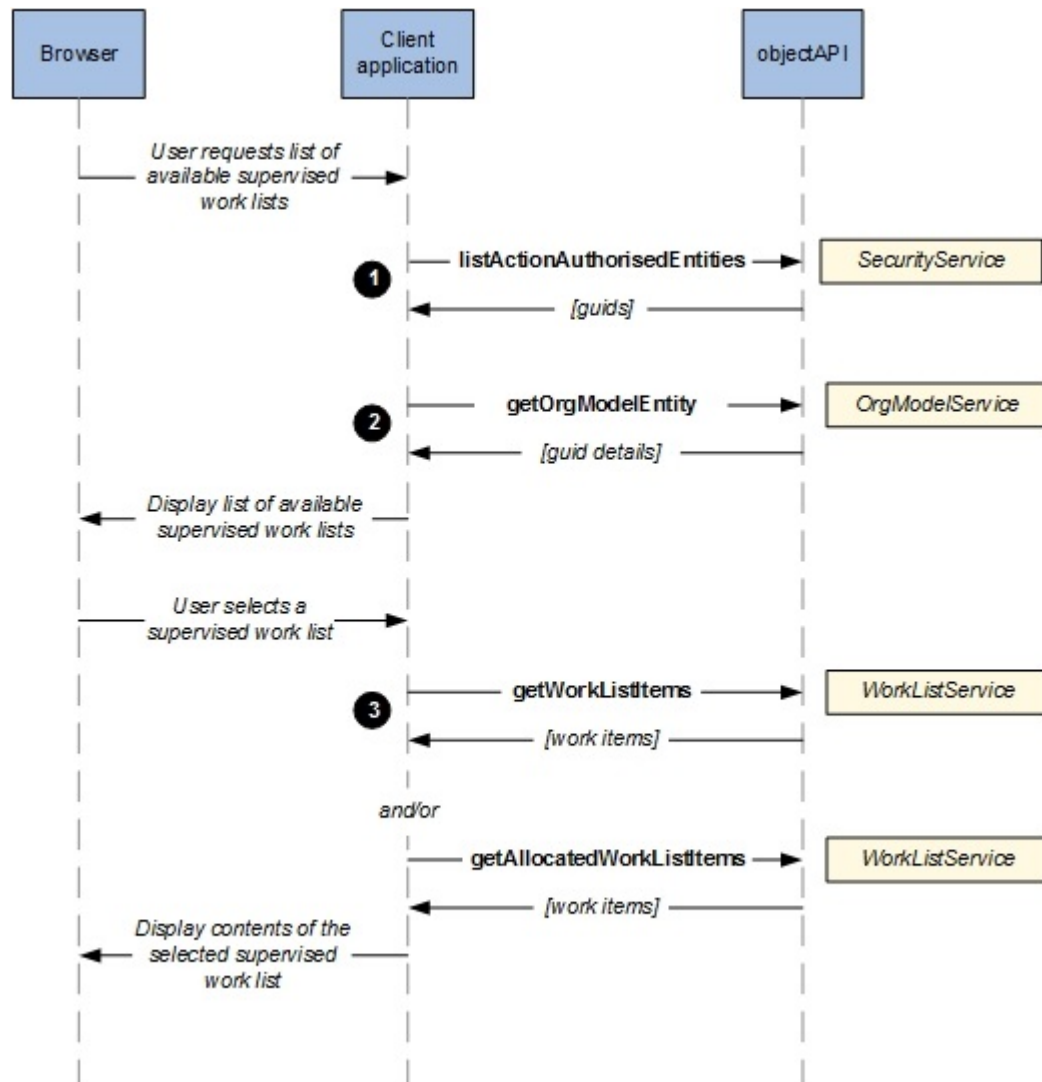
Examples in the following sub-topics illustrate how you can use the API to determine what supervised work lists a user has access to, then retrieve the contents of those work list views.



## Creating and Managing Supervised Work List Views - Example 1

In this example, a user requests a list of available supervised work lists.

### Supervised Work Lists



### Procedure

1. Find out what work lists the user has access to by calling `listActionAuthorisedEntities` for the `viewWorkList` system action. The response contains a list of GUIDs. Each GUID identifies an organization model entity whose work list view the calling user is authorized to view.
2. For each GUID returned in the `listActionAuthorisedEntities` response, call `getOrgModelEntity` to obtain the organization model entity's **entity-type**, along with other details—its name, whether it has any child entities and so on.

You can use this information to present the available supervised work list views to the user in your desired manner. Also see [Navigating the Organization .Model](#)

3. When the user selects a supervised work list view, call either of, or both of, the following:
  - `getWorkListItems`, to retrieve the list of work items currently offered to this user.
  - `getAllocatedWorkListItems`, to retrieve the list of work items currently allocated to this user.



## Creating and Managing Supervised Work List Views - Example 2

You may want to provide views based on different filters of the data.

For example, to allow a supervisor to select a particular process instance and see a particular user's outstanding, allocated work for that process instance:

### Procedure

1. Get the process instance ID.  
You can get this information from any of the [listProcessInstances](#), [queryProcessInstances](#), and [queryProcessInstancesAlt](#) operations.
2. Use [listActionAuthorisedEntities](#) and [getOrgModelEntity](#) in the same way described in [Creating and Managing Supervised Work List Views - Example 1](#).
3. Call [getAllocatedWorkListItems](#), specifying the organizational entity whose list you want to view, and a filter set for the appropriate process instance.

## Deleting Work List Views

To delete a work list view, the calling resource must be the Owner or an Author of the work list view.

The work list view must be locked using [getWorkListViewDetails](#) before it can be deleted.

After locking the work list view, use [deleteWorkListView](#) to delete the view.

## Work Items

Functions are provided that allow you to control work items, such as opening, closing, cancelling, allocating, and so on.

The following services contain work item-related operations:

- [WorkPresentationService](#) - The functions in this service are used when the work item presents a form to the user, which is typically the case for most client applications.

This service contains the following functions for working with work item forms:

- [openWorkItem](#) - Opens a work item form.

Note that this function (from [WorkPresentationService](#)) is also used if there is a pageflow associated with the user task. The response from the [openWorkItem](#) function contains the pageflow details, which you use with the [startPageFlow](#) function in the [PageFlowService](#) to start the pageflow (the pageflow then opens the form).

- [closeWorkItem](#) - Closes the form for the specified work item and updates the associated input and output data. The work item remains in the work list from which it was opened. This is typically called when the user clicks the **Close** button on the work item form.



Also see the [saveOpenWorkItem](#) function (which is in the [WorkItemManagementService](#)) for information about saving changes that have been made on a work item form without closing the form.

- [cancelWorkItem](#) - Closes the form for the specified work item and discards any changes that were made on the form. The work item remains in the work list from which it was opened. This is typically called when the user clicks the **Cancel** button on the work item form.
- [completeWorkItem](#) - Closes the specified work item form and updates the associated input and output data. The work item is removed from the work list from which it was opened, and the process advances. This is typically called when the user clicks the **Submit** button on the work item form.

- [openNextWorkItem](#) - Opens the work item form for the next work item in the specified resource's work list.
- [WorkItemManagementService](#) - This service contains many functions that can be used to manage work items, such as allocating, skipping, getting offer set, and so on.

The thing to keep in mind, however, is that the three functions that this service has in common with [WorkPresentationService](#) ([openWorkItem](#), [closeWorkItem](#), and [completeWorkItem](#)) should be called from [WorkItemManagementService](#) only when the work item does not present a form to the user. In most cases, a form is presented to the user; when a form is displayed, use the functions in [WorkPresentationService](#) instead.

Also see [Processing a Work Item](#) for information about using the functions in the services described above to progress a work item through a process.

## Work Item Versions

Functions that directly specify a particular work item do so by passing the work item ID.

The work item ID includes an optional **version** parameter. The version of a work item starts at 0, and increments by one whenever an operation is performed that changes the work item's state or data.

Various functions also return a version number. If no version number is specified, the latest version is assumed. However, specifying the version can be good practice because it prevents clashes when two or more users attempt to operate on the same work item. For example, two users perform an [allocateAndOpenWorkItem](#) function on a work item. If they do not specify a version number, the second [allocateAndOpenWorkItem](#) function to be executed will fail because the work item is no longer in a suitable state to be allocated or opened. If both users specify the version number of the work item, the second function will still fail, but it will do so with a version number error.


This enables your client application to know that you are attempting to use an out-of-date version of the work item, and to take appropriate action. For example, one possibility would be to use the [getWorkItemHeader](#) function to retrieve the latest information for the work item and bring it back up to date. Another would be to use [previewWorkItemFromList](#) to obtain the current version of the work item.

## Work Item States

A work item is always in one of a small number of defined states.

Some of the operations used to manipulate work items cause an item to change its state, as defined in [Work Item State Transitions](#). The following table shows the possible states.

Work Item State	Meaning
(Created)	This is a private state, accessed only by services that are restricted for internal use.
Offered	This is the initial state of every work item that is made available on the BPM runtime. The work item is offered to those resources who correspond to the organizational entities specified at design time, and is placed in their work list.
Allocated	The work item is assigned to a particular resource to be worked on. A work item may be in the Allocated state more than once in its life cycle, since it can be reallocated to different resources.

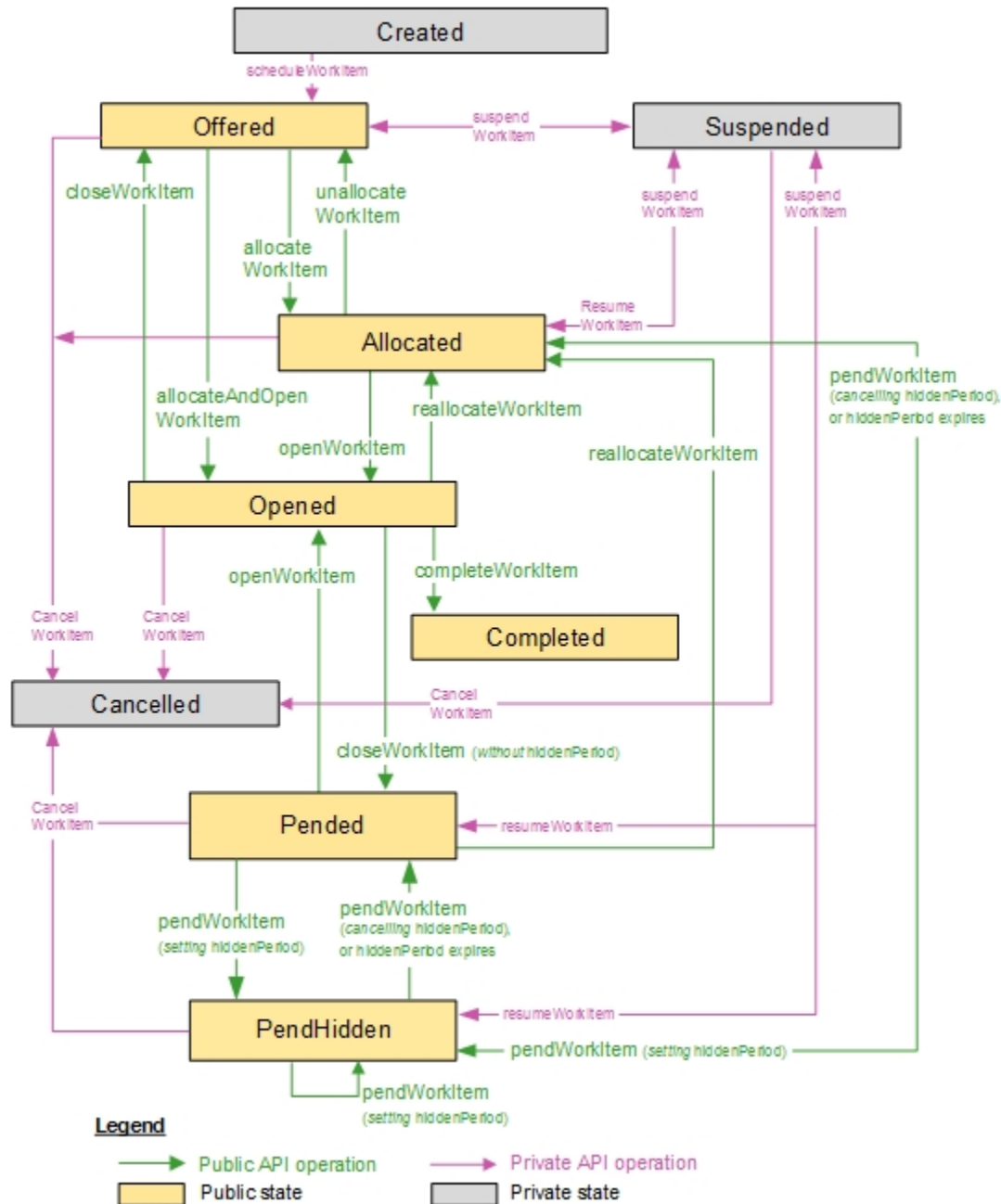
Work Item State	Meaning
Opened	A work item is Opened when a resource (a user) begins work on it, either by selecting it from a work list or by having it automatically allocated and opened.
Suspended	A work item is suspended if its parent process instance is suspended.
Cancelled	<p>An open work item is cancelled when its parent process instance is cancelled. Work items that are not open are deleted when their parent process instance is cancelled.</p> <p>A cancelled work item can be closed by calling <a href="#">closeWorkItem</a> with no associated data. No other operations may affect it.</p>
Pended	<p>The work item is assigned to a particular resource, who has done some work on it, but not yet completed that work.</p> <p>The work item's form has been opened and closed (not cancelled), and data fields may have been modified and saved.)</p> <div>  <p>Pended and Allocated are similar states. The difference is that Pended means the work item <i>has</i> already been worked on. Allocated means that it has <i>not</i> already been worked on. You cannot use <a href="#">skipWorkItem</a> on a Pended work item.</p> </div>
PendHidden	<p>If a work item is Pended with a hiddenPeriod specified, it cannot be accessed for the time defined in that hiddenPeriod.</p> <p>When the hiddenPeriod expires, the work item is returned to the state it was in before it was hidden. (An Allocated work item that was hidden returns to its Allocated state. A Pended work item that was hidden returns to its Pended state.)</p>
Completed	The final state of a work item in the BPM runtime, when work on an Opened work item has been completed. It is no longer on the system and cannot be affected by any API operations.

## Work Item State Transitions

Work items go through transitions from state to state.

The following diagram and table show what transitions a work item goes through between states, and what operations are used to bring about each of the possible changes.

## State Transitions



The [WorkItemManagementService](#) and [WorkPresentationService](#) API includes a number of private services which are intended **only** for internal use by the BPM runtime. These services should not be used by an external application.

State transitions caused by private API service operations have been omitted from the following table.

Start State	End State	BRM API Operation	Description
Offered	Allocated	<a href="#">allocateWorkItem</a>	The offered work item is <i>allocated</i> to the specified organization model entity.

Start State	End State	BRM API Operation	Description
	Opened	<a href="#">allocateAndOpenWorkItem</a> <a href="#">allocateAndOpenNextWorkItem</a>	The offered work item is <i>allocated</i> to the specified organization model entity and immediately <i>opened</i> .
Allocated	Offered	<a href="#">unallocateWorkItem</a>	The allocated object is returned to its original <i>offered</i> state.
	Opened	<a href="#">openWorkItem</a>	The allocated work item is <i>opened</i> .
	PendHidden	<a href="#">pendWorkItem</a> ( <i>setting a hiddenPeriod</i> )	The allocated work item is put into the <i>pendHidden</i> state for the duration of the specified <i>hiddenPeriod</i> .  When the <i>hiddenPeriod</i> timer expires, the work item is returned to its original <i>allocated</i> state.
Opened	Offered	<a href="#">closeWorkItem</a>	The open work item (which must contain no data changes) is closed and returned to its <i>offered</i> state.
	Allocated	<a href="#">reallocateWorkItem</a>	The open work item is reallocated to the specified organization model entity. It will be in the <i>allocated</i> state.
	Complete	<a href="#">completeWorkItem</a>	The opened work item is <i>complete</i> .
	Pended	<a href="#">closeWorkItem</a> ( <i>without a hiddenPeriod</i> )	The open work item is closed and any new data copied. It is then put into the <i>pended</i> state.
	PendHidden	<a href="#">closeWorkItem</a> ( <i>setting a hiddenPeriod</i> )	The open work item is closed and any new data copied. It is then put into the <i>pendHidden</i> state.  When the <i>hiddenPeriod</i> timer expires, the work item is transitioned to the <i>pended</i> state.
Pended	Allocated	<a href="#">reallocateWorkItem</a>	The pended work item is reallocated to another organization model entity and put into the <i>allocated</i> state.
	Opened	<a href="#">openWorkItem</a>	The pended work item is <i>opened</i> .

Start State	End State	BRM API Operation	Description
	PendHidden	<a href="#">pendWorkItem</a> ( <i>setting a hiddenPeriod</i> )	The pending work item is put into the <i>pendHidden</i> state for the duration of the specified <i>hiddenPeriod</i> .  When the <i>hiddenPeriod</i> timer expires, the work item is returned to its original <i>pending</i> state.
PendHidden	Pended or Allocated	<a href="#">pendWorkItem</a> ( <i>cancelling a hiddenPeriod</i> )	A work item that was hidden using <a href="#">pendWorkItem</a> is returned to the state it was in before it was hidden -- <i>pending</i> or <i>allocated</i> .
	PendHidden	<a href="#">pendWorkItem</a> ( <i>setting a hiddenPeriod</i> )	The duration for which the work item will remain in the <i>pendHidden</i> state is reset to the specified <i>hiddenPeriod</i> .



A work item cannot be accessed while it is in the *PendHidden* state. See [Accessing Hidden Work Items](#).

## Retrieving Information on Work Items

Functions are provided to retrieve information about a specified work item.

These functions are in the [WorkItemManagementService](#) and [WorkListService](#) services:

- [previewWorkItemFromList](#) - Use this function to retrieve information on one or more specified work items in a work list. The request specifies the GUID of the requesting organizational entity, the IDs of one or more work items, and optionally a list of the fields about which information is required. You can use this function:
  - To get the type and value of one or more specific fields. You can filter the information returned by specifying one or more field names as **requiredField** parameters. If you do, the function returns information only on those named fields.
  - To get information on all fields. If you do not include the **requiredField** parameter, the response returns the name, type, and value of all fields.
- [getWorkItemHeader](#) - Use this function to retrieve the header information for a specified work item.

The request specifies the `workItemID` of the required work item, which you can get from the response to a previous call to [getWorkListItems](#) or [getAllocatedWorkListItems](#).

The response returns the work item header information. The header information includes the work item's:

- name
- description
- distributionStrategy
- priority
- startDate

- `endDate`

These attributes can be used as criteria for sorting or filtering work items in a work item list; see [Sort Filter Criteria](#).

You can use this function to bring a work item up to date if you have called an obsolete version. For more information, see [Work Item Versions](#).

- [getOfferSet](#) - Use this function to get the initial offered set for a work item, that is, the collection of organizational entities to which the work item was initially offered.

The request specifies the unique ID of the required work item and the response returns the IDs of all entities in the offer set.



You can specify whether this function returns only the GUIDs of the organizational entities, or additional information, including the GUID, the major version of the organization model in use, and the type of the organizational entity.

This information can then be used to allocate the work item to a particular resource from within this set; see [Allocating a Work Item to a Target Resource](#).

## Offering and Allocating Work Items

Functions are provided that are used to offer and allocate work items to resources.

These [WorkItemManagementService](#) functions are:

Operation Name	When to Use
<a href="#">allocateWorkItem</a>	To allocate an offered work item to a single resource, when you need not retrieve the associated input and output data.
<a href="#">allocateAndOpenWorkItem</a>	To allocate an offered work item to a single resource. The work item is immediately opened and the associated input and output data is retrieved.
<a href="#">allocateAndOpenNextWorkItem</a>	To allocate the next offered work item to a single resource and immediately open that work item to get the associated input and output data.  The "next" work item is defined by the sort criteria set for the resource. If no sort criteria are set, the default criteria are used.
<a href="#">reallocateWorkItem</a>	To reallocate an opened work item to a different resource, or to reallocate a pending work item to a resource.  No data is changed.
<a href="#">reallocateWorkItemData</a>	To reallocate an opened or a pending work item to a different resource and write the new work item data.
<a href="#">unallocateWorkItem</a>	To reoffer a work item. A currently allocated work item is unallocated so that it is reoffered to the original offer set.

The following sections describe how to allocate work items to a target resource:

- [Allocating a Work Item to a Target Resource](#)
- [Reallocating a Work Item](#)
- [Reoffering a Work Item](#)

## Allocating a Work Item to a Target Resource

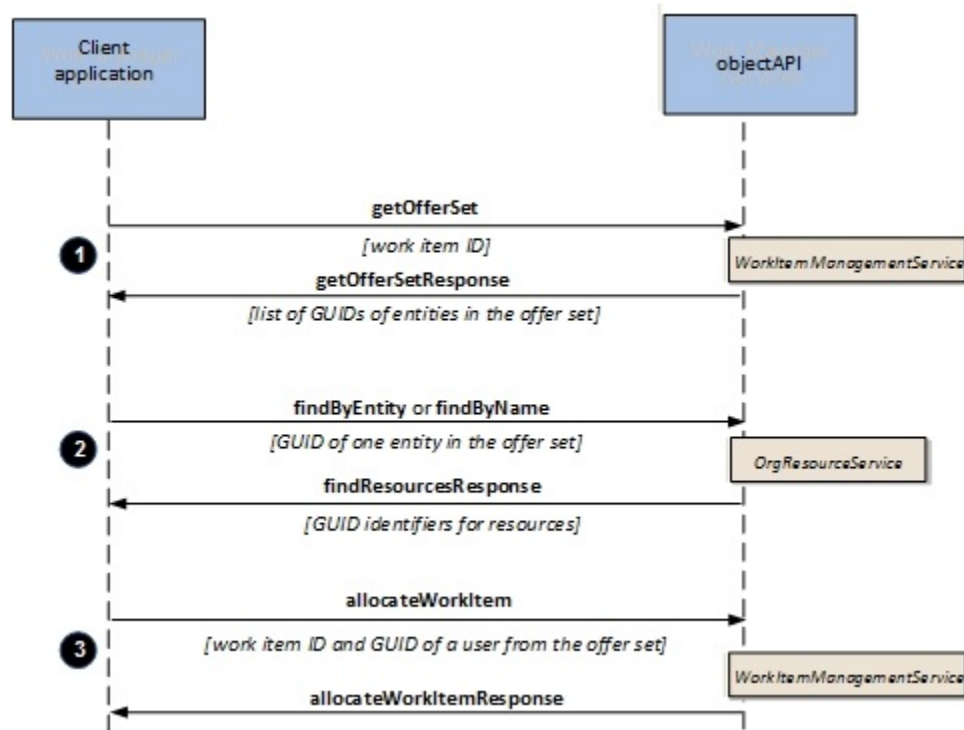
You can choose to allocate a work item to a resource.

The work item can be allocated to one of the following:

- **Initial Offered Set.** You can query the initial offered set using the function [getOfferSet](#), and select a resource from the offer set to allocate the work item to.
- **World.** If your system privileges include the `reallocateWorkItemToWorld` system action scoped at the organization model level, you can allocate the work item to a resource that was not included in the initial offered set. If there is more than one organization in your organization model, this can be any resource in an organization with which the current LDAP container has a relationship. For more information, see [Organization Relationships](#).
- **Self.** You can allocate the work item to yourself.

The following diagram shows an example of how you might take a work item from the list of items offered to an organizational entity, such as a group or position, and allocate it to a specific resource within the original offer set.

### Allocating a Work Item



### Procedure

1. Use [getOfferSet](#), specifying the ID of a work item, to list the entities to which this work item was offered.
2. Use [findByEntity](#) or [findByName](#) to identify the GUID of one resource (user) from the offer set.
3. Use [allocateWorkItem](#) to allocate the required work item to an individual resource.



## Reallocating a Work Item

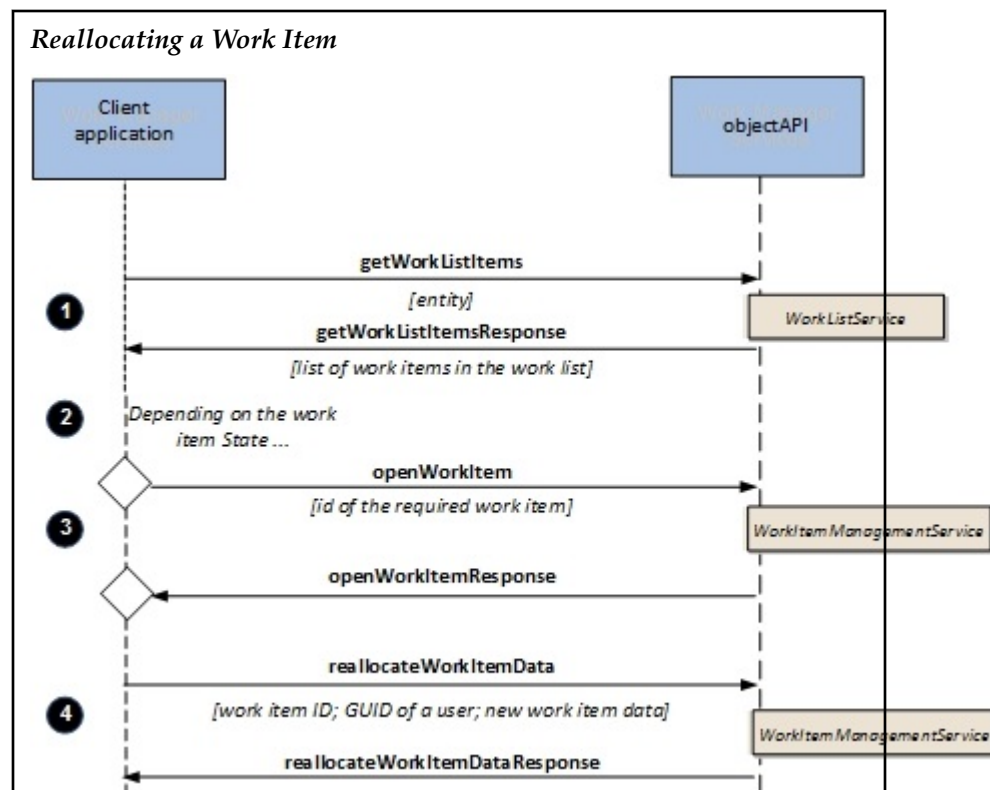
When reallocating a work item to another resource, you need to choose whether to reallocate the work item, or reallocate the work item data.

The principal difference is in the handling of the data associated with the work item:

- Using [reallocateWorkItemData](#) you can update the work item body with new data as part of the operation.
- Using [reallocateWorkItem](#) you do not allocate any new data, but you can choose what to do with the existing data associated with the work item, by setting the boolean element **revertData**:
  - If you set **revertData** to "false", the work item is reallocated just as it is, with existing data unchanged.
  - If you set **revertData** to "true", all existing data is discarded. The work item is returned to its original state by reverting to the snapshot taken when it was scheduled.

You can reallocate an Opened or Pended work item. You can reallocate an Allocated item, but you cannot use [reallocateWorkItemData](#) to update the work item body. If you wish to reallocate and update the data for a work item that is already Allocated, change its status first.

The following diagram shows an example of how you might reallocate a work item with fresh data, bearing in mind the need to determine whether it is in a suitable state.



## Procedure

1. To determine whether the work item is in a suitable state, call [getWorkListItems](#), specifying the resource (user) or other organizational entity on whose work list the work item currently appears. The response to this function returns among other data the work item state.
2. Determine whether the required work item is in a state in which the [reallocateWorkItemData](#) function can be used—that is, if it is in an Opened or Pended state.

3. If it is not, call the [openWorkItem](#) operation to put the work item in a suitable state to be reallocated.
4. Call the [reallocateWorkItemData](#) function to reallocate the work item to a new user and pass new data to them.

### **Reallocating to Offer Set or to World**

Depending on the system actions a user holds, they may be able to reallocate a work item more or less widely.

For example:

- **reallocateToOfferSet** - This system action authorizes you to reallocate a work item to a resource in the original offer set. If, for example, the originally allocated resource is absent or overloaded, this might be sufficient to reallocate a work item to another member of the same team.
- **reallocateWorkItemToWorld** - This system action authorizes you to reallocate a work item to any resource—that is, to anybody in the "world" of the BPM runtime. You might need this level of authorization if, for example, it was decided that a work item needed reallocating to another department or to a resource with significantly different expertise, who was not in the original offer set.

### **Reoffering a Work Item**

If a work item needs to be transferred from a user to whom it is currently allocated, but you want to reoffer it to the initial offered set rather than allocating it directly to a specific resource, you can use the `unallocateWorkItem` function

The [unallocateWorkItem](#) function resets the work item to the Offered state.

Note that you cannot change the resources to which the work item is offered; you can only reallocate to the original offer set. If you need the work item to be assigned to a completely different resource, you must possess the **reallocateWorkItemToWorld** system action, and reallocate the work item using [reallocateWorkItem](#), as described in [Reallocating to Offer Set or to World](#).

### **Required System Actions**

System actions are required to reallocate/reoffer a work item.

They are:

- **DE.browseModel** (allowed by default)
- **DE.resolveResource** (allowed by default)
- **DE.resourceAdmin**
- **BRM.viewWorkList**
- **BRM.workItemAllocation** (You must have this system action at the organization model level, as the one at the scoped level is not used)
- If reallocating to World: **BRM.reallocateWorkItemToWorld**  
(You must have this system action at the organization model level, as the one at the scoped level is not used )
- If reallocating to Offer Set: **BRM.reallocateToOfferSet** (You must have this system action at the organization model level, as the one at the scoped level is not used )

Only work items in the Allocated or Pended states can be reallocated to World (because you cannot try to allocate an item in the offered state to a resource outside the original offer set). However, if the work item is already allocated, then you can reallocate to any resource.

See [System Actions](#).

## Opening a Work Item

Opening a work item results in a form opening, a pageflow starting, or work item data being retrieved.

For example:

- **A form is opened** - If the user task was designed to display a form, use the [openWorkItem](#) request in the [WorkPresentationService](#).

For additional information about opening a work item and displaying a form, see [Displaying a Work Item Form](#).

If the user task was designed to display a custom form (rather than a TIBCO Business Studio form), also see [Custom Forms](#).

- **A pageflow is started** - If the user task was designed to start a pageflow when it is opened, use [openWorkItem](#) request in the [WorkPresentationService](#). The response from the [openWorkItem](#) function contains the pageflow details, which you use with the [startPageFlow](#) request in the [PageFlowService](#) to start the pageflow.

For additional information, and an example of starting a pageflow when a work item is opened, see [Displaying a Form in a Pageflow](#).

- **Work item data is retrieved (but no form is displayed)** - If the user task was designed to neither open a form nor start a pageflow, use the [openWorkItem](#) request in the [WorkItemManagementService](#). This returns all of the associated input and output data for the work item.

Note that this is typically only used in special use-case client applications.

To use either of the [openWorkItem](#) functions, the work item must currently be in a state of Allocated or Pended.

If a work item is currently in a state of Offered, you can use the [allocateAndOpenWorkItem](#) and [allocateAndOpenNextWorkItem](#) requests in the [WorkItemManagementService](#) to allocate and open the work item at the same time.

For more information about work item states, see [Work Item States](#).

For information about additional requests that can be made to progress a work item after opening it, see [Processing a Work Item](#).

## Processing a Work Item

A number of functions are provided that are used to process a work item.

Each of these functions is applicable only to work items in a particular state or states—for example, you cannot perform an [openWorkItem](#) on a work item that is already Opened. These functions will also change the state of the affected work item—to take [openWorkItem](#) as an example again, a successful [openWorkItem](#) call changes the state of the work item from Allocated or Pended to Opened. See [Work Item States](#) and [Accessing Hidden Work Items](#) for information about the possible states of work items and the possible changes between states.

The following functions are used for progressing work items:

- [openWorkItem](#)
- [closeWorkItem](#)
- [completeWorkItem](#)
- [cancelWorkItem](#)
- [skipWorkItem](#)
- [saveOpenWorkItem](#)

- [pendWorkItem](#)
- [rescheduleWorkItem](#)
- [setWorkItemPriority](#)

To identify which work item is affected, the requests for these operations require the unique ID of the work item as input. You can obtain the work item ID by several methods, for example from the response to a previous [getWorkListItems](#) or [getAllocatedWorkListItems](#) function

### **openWorkItem**

Use this function to open a work item and to retrieve the associated data. The work item must be Allocated or Pended, and its state is changed to Opened. The function returns the body of the work item.

Note that this function is available in two different services:

- [WorkPresentationService](#) - The [openWorkItem](#) function in this service should be used in the following situations:
  - If the user task is designed to display a form. This is typical for most client applications. (If the work item is designed to display a custom form, rather than a TIBCO Business Studio form, also see [Custom Forms](#).)
  - If the user task is designed to start a pageflow. The response from the [openWorkItem](#) function contains the pageflow details, which you use with the [startPageFlow](#) function in the [PageFlowService](#) to start the pageflow.
- [WorkItemManagementService](#) - The [openWorkItem](#) function in this service is used only if the user task does not open a form nor start a pageflow. This would be used only in special use-case client applications.

### **closeWorkItem**

Use this function to close a work item and to update the associated data with any changes the user has made while it was open. This function is for work items that are not completed, but on which work has been paused.

The work item must be Opened. The function sets the state to Pended.

Note that this function is available in two different services:

- [WorkPresentationService](#) - The [closeWorkItem](#) function in this service should be used to close a form that was opened when the work item was opened with the [openWorkItem](#) function in [WorkPresentationService](#). This is typically called in response to a user clicking the **Close** button on a work item form.
- [WorkItemManagementService](#) - The [closeWorkItem](#) function in this service is used only if the user task did not open a form when the work item was opened. This would be used only in special use-case client applications.

### **completeWorkItem**

Use this function when work on a work item or pageflow has completed, and to update the associated data with any changes the user has made while the work item or pageflow was open.

The work item must be Opened, and its state is changed to Completed.

Note that this function is available in two different services:

- [WorkPresentationService](#) - The [completeWorkItem](#) function in this service should be used in the following situations:

- To complete a work item when a form was opened when the work item was opened with the [openWorkItem](#) function in the [WorkPresentationService](#). Input and output data associated with the work item is saved to the database. This is typically called in response to a user clicking the **Submit** button on a work item form.
- To complete the parent work item when a pageflow was started after the work item was opened with the [openWorkItem](#) function in the [WorkPresentationService](#). The work item form is closed and input and output data associated with the work item (and pageflow) is saved to the database.
- [WorkItemManagementService](#) - The [completeWorkItem](#) function in this service is used only if the user task does not open a form nor start a pageflow. This would be used only in special use-case client applications.

### **cancelWorkItem**

Use this function to cancel any changes that a user has made on a work item form. It closes the work item form, discards any changes on the form, and leaves the work item in the user's work list.

The [cancelWorkItem](#) function is available in the [WorkPresentationService](#).

### **skipWorkItem**

Use this function to skip a work item, so that no action is carried out on it.

The [skipWorkItem](#) function (which is in the [WorkItemManagementService](#)) has a very similar effect to completing a work item, in that it means that no further work is required. This function can only be carried out:

- By a user who has authorization for the **skipWorkItem** system action, and
- on work items for which all required output and in/out data fields either:
  - have default values defined, or
  - in the case of input/output fields, have been give values before this work item, by a script or another activity.

The work item must be Allocated, and once skipped its state is changed to Completed.

### **saveOpenWorkItem**

Use this function to save an open work item, updating the associated data with any changes the user has made while it was open. The work item must be Opened. Its state is not changed by this function.

The [saveOpenWorkItem](#) function is in the [WorkItemManagementService](#).

### **pendWorkItem**

Use this function to set aside an unfinished work item as pending. The work item to be pended must already be in an Allocated, Pended or PendHidden state.

The [pendWorkItem](#) function is in the [WorkItemManagementService](#).

Optionally, you can specify a **hiddenPeriod** parameter to make the work item inaccessible for a specified period of time, or to change the duration of or cancel an existing hiddenPeriod. See [Work Item State Transitions](#) and [Accessing Hidden Work Items](#) for more information.

### **rescheduleWorkItem**

Use this function to reschedule an existing work item.

The [rescheduleWorkItem](#) function (which is in the [WorkItemManagementService](#)) can be used to update the work item's schedule period and/or update the work item's data.

For example, if customer details need to be updated. Any work item data can be changed. However, you cannot unset existing data. This function can be carried out when the work item is in any state other than Completed. The work item's state is not changed when it is rescheduled.

The request is made up of:

- the **itemSchedule**. This is the work item schedule period to be associated with the work item. If no Schedule period is specified, the item's schedule period is left unmodified. The **itemSchedule** can optionally include the **startDate** and either the **maxDuration** or **targetDate**.
- the **itemBody**. This is the work item body definition. This specifies the work item data to be changed. If no **itemBody** is specified, the work item data is not changed. It contains each data field (as a name/value pair) required by the data model defined in the work item's work type. The associated work type specifies each data field's type and whether it is an INPUT, OUTPUT or INOUT parameter. The **itemBody** must also include the parameter and either the complex value or value.

If a user is using an application to enter data in a work item form either before or at the same time a `rescheduleWorkItem` function is performed, how the work item data is modified depends on:

- how you have configured your process definition in TIBCO Business Studio. The modification of work item data is configured using the **Overwrite data already modified in work item** check box. See *TIBCO Business Studio Process Modeling Guide* for more information.
- whether the work item form is open or closed at the time the `rescheduleWorkItem` function is performed.

If the work item form is closed and a user has made one or more changes to one or more fields on the form:

- If the **Overwrite data already modified in work item** check box is selected, when the `rescheduleWorkItem` function is performed, any data that has been updated by a user is overwritten with the new data from the `rescheduleWorkItem` function. The remaining fields rescheduled for update are also updated.
- If the **Overwrite data already modified in work item** check box is not selected, any changes a user has made to a form remain, but the other fields are overwritten with the new data from the `rescheduleWorkItem` function.

If the work item form is open when the `rescheduleWorkItem` function is performed, when the user clicks **Submit** or **Close**, a dialog displays that allows them to do the following:

- If the **Overwrite data already modified in work item** check box is not selected, the following operations are available.
  - **Override** - the user can also override any changes to the data. In this case, the changes that the user has made to the form remain and the changes that have been made by TIBCO ActiveMatrix BPM are lost.
  - **Reload** - reload the form with the new changes displayed. In this case, any changes that the user has made to the form are lost. The user can then decide whether to re-enter the data or submit it as it is.
  - **Cancel** - cancel the changes made to the work item form. In this case, all the data the user has entered is lost but the changes made by TIBCO ActiveMatrix BPM are retained.
- If the **Overwrite data already modified in work item** check box is selected, the Override option is not available.

If a `rescheduleScript` has been defined, it is run each time the `rescheduleWorkItem` function is called. See the *TIBCO BPM Implementation Guide* for more information about `rescheduleScript`.

This function can only be carried out by a user who has authorization for the **rescheduleWorkItem** system action.

## setWorkItemPriority

Use this function to set the priority of a work item.

Setting the work item priority with the [setWorkItemPriority](#) function (which is in the [WorkItemManagementService](#)) allows a user to sort their work list by Priority.

Also, scripts and processes can check the priority value and therefore perform actions based on the work item priority.

All work items have a priority. The priority is a numeric value indicating the relative importance of the work item. By default, the work item priority is 50. The valid range for work item priority is 1 - 100. (You must specify a value in the range of 1 - 100 (inclusive). If you enter a value outside this range, an exception is thrown.)

You can change the priority of an individual work item or multiple work items. Depending on your requirements, you have a choice of one of the following:

- **absPriority**. A numeric value from 1 to 100.
- **offsetPriority**. Offset a work item priority by a specific numeric value. For example, an employee is away on holiday and you want to offset the priority on all of the employee's work items by 20.

The request is made up of:

- **workItemID**: ID of the work item for which information is required.
- **version**: (Optional) If it is omitted, the latest version is used.
- work item priority, for example, **abspriority** or **offsetPriority**.

To use this function:

- A work item can be in any state except Open and Completed. If you reset the work item's priority, its state is not changed.
- You must have authorization for one or both of the following system actions:
  - **changeAllocatedWorkItemPriority** - This system action allows you to reset the priorities of work items with a status of Allocated and that are allocated to the resource performing the action.
  - **changeAnyWorkItemPriority** - This system action allows you to change the work item priority of any work items.

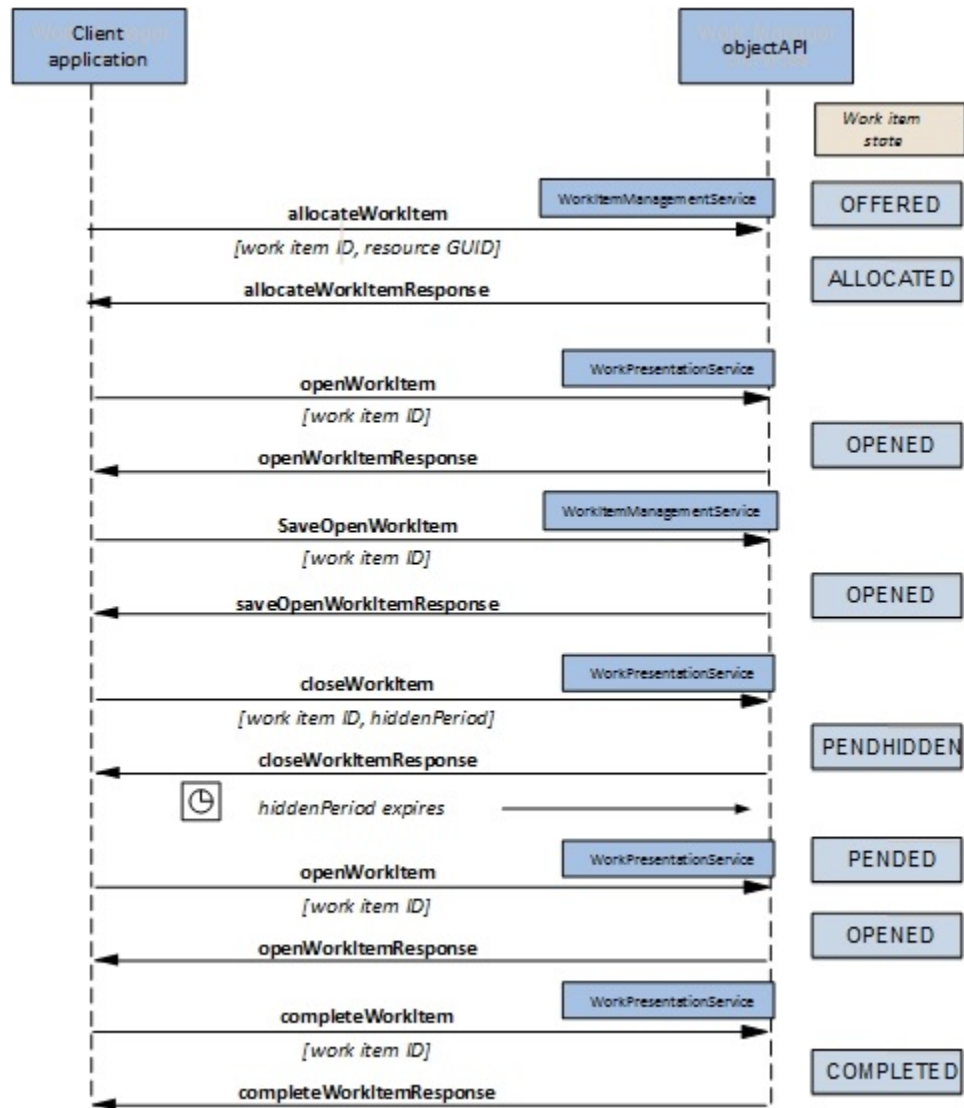
This means if a user wants to change the priority of a work item in their work list whose status is Allocated, but it is not allocated to them, then they must also have authorization for the **changeAnyWorkItemPriority** system action.



## Example of Processing a Work Item

This shows a simple example of the possible flow for a work item.

### Processing a Work Item



In this example:

### Procedure

1. A work item appears in an organizational entity's work list. Its initial state is Offered.
2. To direct it to a resource (user) within that organizational entity, use the [allocateWorkItem](#) call, specifying the ID of the work item and the GUID of the resource as input parameters. The state of the work item changes to Allocated.
3. When the user opens the work item from their client user interface, the client application calls the [openWorkItem](#) function, with the work item ID as an input parameter. The status of the work item changes to Opened.
4. The client application may make provision for the user to save the work item part way through completing it. This would be particularly useful if the work item displays a form containing a large number of fields or that may take some time to complete. In such circumstances the client



application can call the [saveOpenWorkItem](#) function. This saves the data associated with the work item, but does not change its state—it remains Opened.

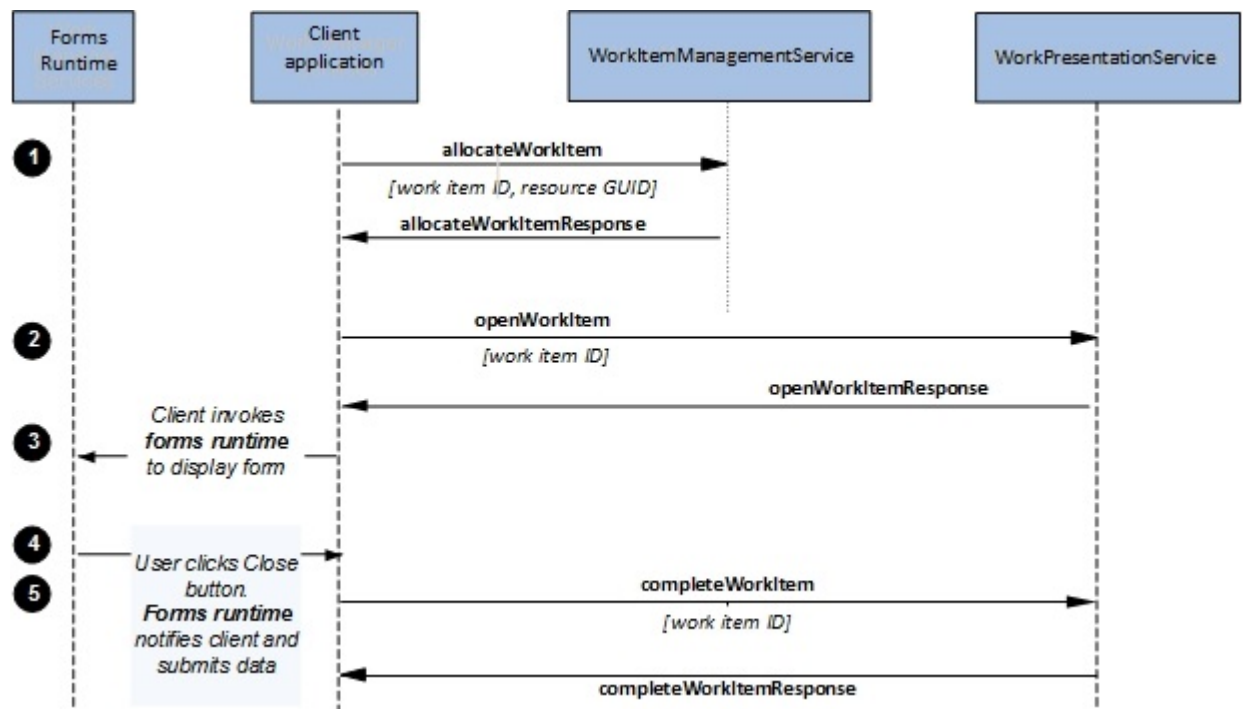
5. In some applications a user may have to wait some time before finalizing a work item, for example to wait for another process to complete or for information to come in from an outside source. In that case the work item can be pended. If it is in an Opened state as in this example, that can be achieved by calling the [closeWorkItem](#) function in [WorkPresentationService](#) and specify a hiddenPeriod. That work item cannot be operated on for the specified period of time, or until the specified date and time is reached. Until then, it is in the PendHidden state.
6. When the pendHidden deadline is reached, a timer expires and the work item is put in the Pended state.
7. The user can now resume work on it. This is done by calling [openWorkItem](#) again, and returning the work item to the Opened state.
8. When the user has finished with the work item, and indicates this by pressing a **Submit** button or similar in their client application's user interface, the [completeWorkItem](#) function is called and the state of the work item changed to Completed.

### Example of Processing a Work Item with a Form

This topic shows a simple example of the possible flow for a work item that presents information to the user using a custom form.

For additional information, see [Displaying a Work Item Form](#).

#### *Processing a Work Item with a Form*



In this example:

#### Procedure

1. A work item appears in an organizational entity's work list. To direct it to a resource (user) within that organizational entity, use the [allocateWorkItem](#) call from [WorkItemManagementService](#), specifying the ID of the work item and the GUID of the resource as input parameters.
2. When the user opens the work item from their client user interface, the client application calls the [openWorkItem](#) function from [WorkPresentationService](#), with the work item ID as an input

parameter. The [WorkPresentationService](#) operation is used instead of the operation of the same name in the [WorkItemManagementService](#), because the work item makes a visual presentation to the user, and [WorkPresentationService](#) is optimized for the presentation of forms.

3. The client application invokes the **Forms runtime** to display the work item's initial form. It may make other calls to the Forms runtime for subsequent forms within the work item. See [Forms](#) for further information on using the Forms runtime.
4. When the user has finished with the work item, and indicates this by pressing a **Close** button or similar in their client application's user interface, the Forms runtime notifies the client application.
5. The client then invokes the [completeWorkItem](#) function from [WorkItemManagementService](#).



Throughout this example the state of the work item changes as described in [Example of Processing a Work Item](#).

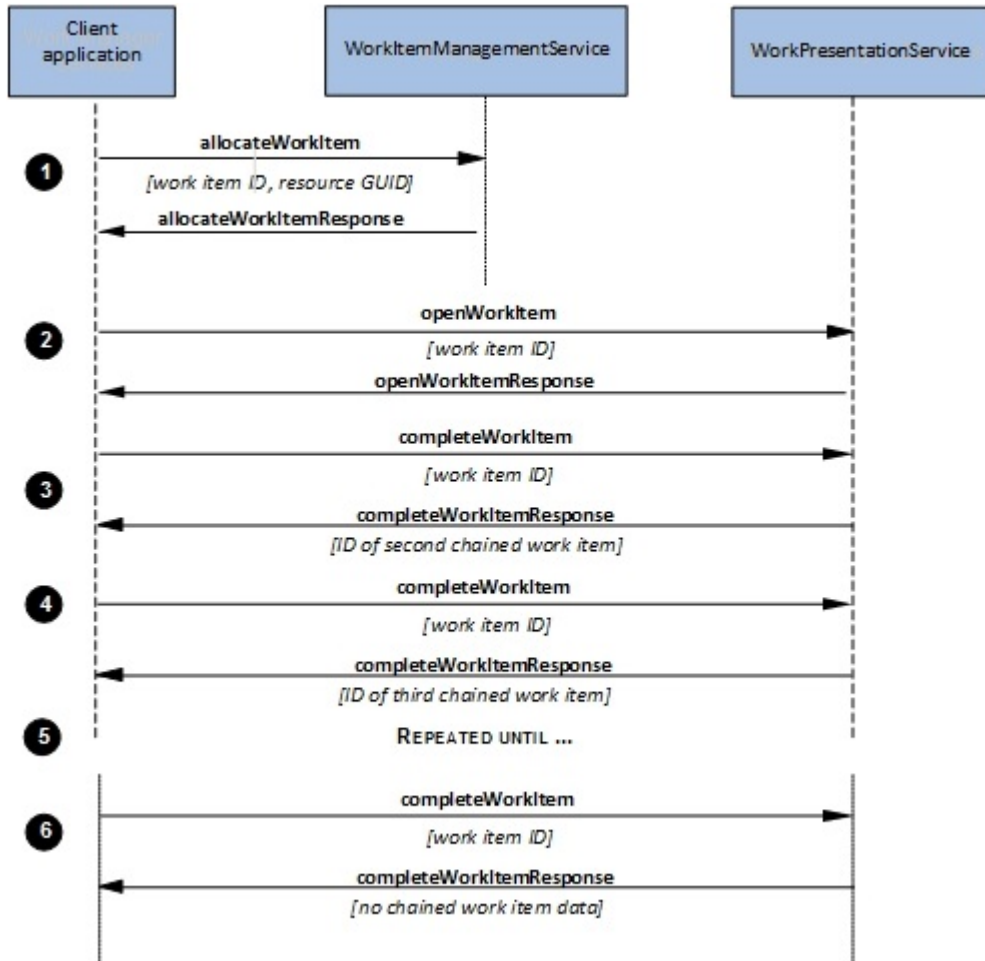
## Processing Chained Work Items

When a business process is designed, several work items may be grouped together in an embedded sub-process and defined as chained. This means that they must be carried out in succession by the same user.

See "Chained Execution" in the *TIBCO Business Studio Process Modeling User's Guide* for more information on chaining.

In order to implement chained execution at runtime, when the first work item of a chained group is completed, the next work item in the chain must immediately be opened and allocated to the same resource who executed the first one. To achieve this, use the [completeWorkItem](#) function in [WorkPresentationService](#).

## Chained Work Items



In this example:

### Procedure

1. A work item is assigned to a resource (user) by the [allocateWorkItem](#) call from [WorkItemManagementService](#), as in [Example of Processing a Work Item](#).
2. When the user opens the work item from their client user interface, the client application calls the [openWorkItem](#) function from [WorkPresentationService](#), with the work item ID as an input parameter.
3. When the user has finished with the work item, the client application invokes the [completeWorkItem](#) function from [WorkPresentationService](#). The response to [completeWorkItem](#) returns the ID of the next work item in the chained subprocess.



Do not use the [WorkItemManagementService](#) to open and complete work items that are being chained; use the [WorkPresentationService](#).

4. The [completeWorkItem](#) response automatically opens the next chained work item; the client application does not need to call [openWorkItem](#) again.



The next chained work item must arrive within the timeout period specified by the `CHAINED_TIMEOUT` property in the `WPProperties.properties` file. If no response is received within that period, the chain "breaks", and no further chained work items are executed.

For more details of this and other properties, see the `WPProperties.properties` file in the "BPM Properties Files" section of the *TIBCO ActiveMatrix BPM - BPM Administration guide*.

5. The `completeWorkItem` function is called for as many times as necessary, each time returning the ID of the next chained work item.
6. Eventually `completeWorkItem` returns no chained work item information, which indicates that the chained subprocess is completed.

## Handling a Work Item That Contains Business Data

Business processes running in the BPM runtime may use their own business data. This consists of user-defined data types representing business-domain objects, such as a **customer** or an **address**.

Business data types are defined in a Business Object Model in TIBCO Business Studio, after which they can be used as part of processes, pageflows or business services that are either part of or reference that project.

To enable your client application to handle a work item (or pageflow page) that contains such business data, you must provide it with the details of the underlying schemas used to define that data.

You cannot obtain this information programmatically. Instead, you must perform the following procedure.

### Procedure

1. In TIBCO Business Studio, identify every project that contains:
  - a process (business, pageflow or business service) with which the client application will interact.
  - a Business Object Model (that defines or uses business data) that is used by one of these processes.
2. Export a Work Data Model for each of these projects.
3. Modify your client application to use the data definitions provided by the Work Data Models.

## Accessing Hidden Work Items

If a **hiddenPeriod** is specified when calling `pendWorkItem` (in `WorkItemManagementService`), the work item transitions to the `PendHidden` state, rather than `Pended`.

Functions that work on the `Pended` state (for example, `reallocateWorkItem`) cannot access the work item until either of the following occurs:

- the `hiddenPeriod` expires.
- `pendWorkItem` is called with a **hiddenPeriod** of 0. This cancels the current `hiddenPeriod`. (A negative `hiddenPeriod` duration or a date that is in the past have the same effect.)

When one of these events occurs, the work item reverts to the state it was in when it was hidden (`Allocated` or `Pended`), if it was hidden using `pendWorkItem`.

The duration of a `hiddenPeriod` can also be extended or reduced by calling `pendWorkItem` again with a new `hiddenPeriod`. The new `hiddenPeriod` overrides the existing `hiddenPeriod` and is calculated from the current date/time.

For example, suppose that `pendWorkItem` was used to pend an allocated work item at 14:30, with a `hiddenPeriod` of 30 minutes. If `pendWorkItem` is called again at 14:45 with:

- a `hiddenPeriod` of 2 hours, the work item will remain hidden until 16:45.
- a `hiddenPeriod` of 5 minutes, the work item will remain hidden until 14:50.
- a `hiddenPeriod` of 0, the work item will immediately revert to Allocated state.
- a `hiddenPeriod` of -5 minutes, the work item will immediately revert to Allocated state.

## Case Data Models

Case data is any business related concept that many BPM-based applications can operate on or be associated with.

For example, in an insurance company, case data may be both an insurance claim and the actual policy. For more information, see the *TIBCO ActiveMatrix BPM Concepts Guide*.

The [GlobaldataAdminService](#) API is used to administer case data models.

Applications with case data use case model database schema generation. This means that when an application that has case data is deployed, a complete set of database tables is generated for that application.

Case data is defined in business object models in TIBCO Business Studio. For more information, see *TIBCO Business Studio Modeling Guide*.

You can configure your system to automatically execute the CREATE/UPDATE/DROP database scripts created from the successful deployment of an application that uses case data.

If you have configured the system so that you must manually execute the database scripts, you can use the [GlobaldataAdminService](#) functions to retrieve these scripts. You can then take these scripts, and optionally, make a number of supported changes on them, depending on your requirements.

Note that the case model must be in the correct state for the database script action you want to perform, as follows:

Case Model State	Database Script Actions
INITIAL	None
PENDING_DBA_TO_INSTALL	<ul style="list-style-type: none"> <li>• Create</li> <li>• Update</li> </ul>
INSTALLED	<ul style="list-style-type: none"> <li>• Freeze</li> <li>• Unfreeze</li> </ul>
PENDING_DBA_UNINSTALL	Drop
FAILED_TO_UNINSTALL	Cleanup
DELETED	None

Once you have verified the scripts and executed them in your database, you can notify the system using the following functions, depending on the script action executed:

- [notifyCleaned](#)

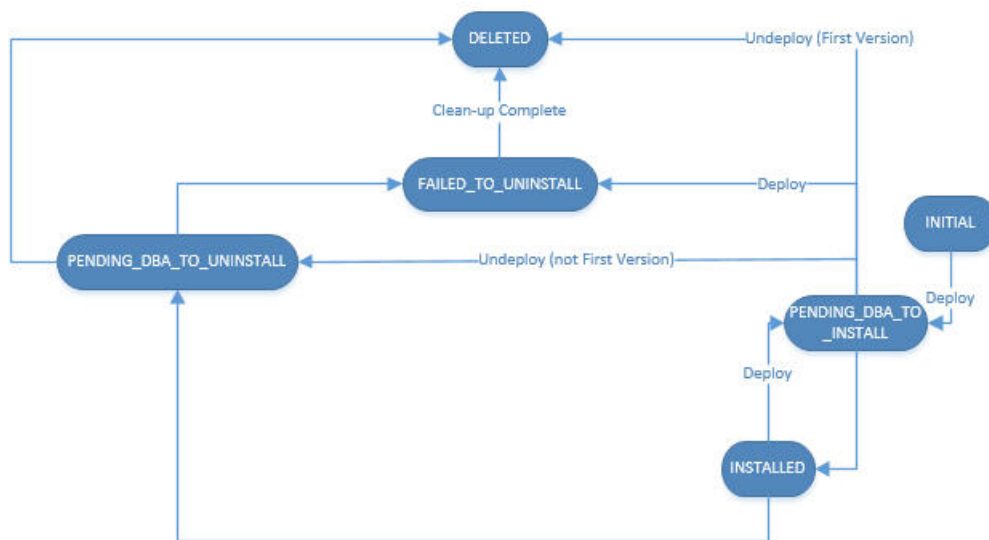
- [notifyCreated](#)
- [notifyDropped](#)
- [notifyFreeze](#)
- [notifyUnfreeze](#)
- [notifyUpdated](#)

When the system has been notified that the database schema is generated, process instances can be run from the application and access the case data defined in the new case data model.

### Deployment Life Cycle of a Case Model

Case models have a deployment lifecycle. In other words, you can deploy new versions and obtain UPDATE database scripts. Similarly, you can use the DROP scripts to remove the case data from the database. For more information about application deployment lifecycles, see the *TIBCO ActiveMatrix BPM Deployment Guide*. You can tell where a case model is in its deployment lifecycle by its state.

The following diagram illustrates the deployment cycle of a case model.



- **INITIAL.** This describes the initial state of the application. The application is either still in the process of deploying or is undeployed.
- **PENDING\_DBA\_TO\_INSTALL.** The CREATE/UPDATE scripts are ready for retrieval. You must retrieve the scripts and manually execute them on your database to generate the case model database schema. Once the scripts have been executed, you must notify the system.

If the first version of an application is undeployed before the UPDATE scripts for a new version of the application have been executed, then the case model's state changes to **PENDING\_DBA\_TO UNINSTALL** so the DROP scripts can be retrieved.

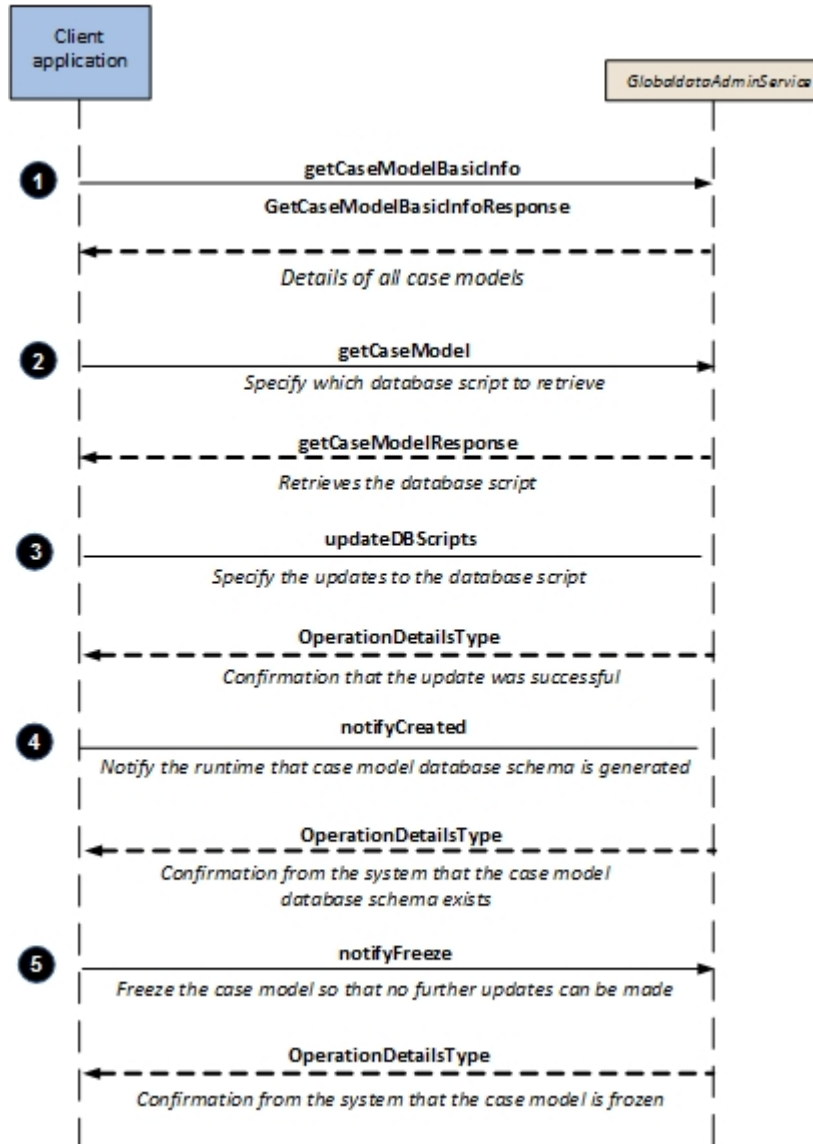
- **INSTALLED.** A case model has a state of **INSTALLED** if the CREATE/UPDATE scripts have been executed and the system has been notified. Process instances may exist for the application.
- **PENDING\_DBA\_TO\_UNINSTALL.** The DROP script is ready for retrieval. You must retrieve the scripts and manually execute them on your database to drop the case model database schema. Once the scripts have been deleted or dropped, you must notify the system.
- **FAILED\_TO\_UNINSTALL** The scripts have not executed correctly on the database. You must clean-up your database and then, you must notify the system when the clean-up is complete.

- **DELETED.** The DROP script has been executed and the case model database tables deleted. The case model is removed.

## Retrieving Case Model Database Scripts

You can manually execute the case model database scripts.

The following diagram shows an example of the calls to the [GlobaldataAdminService](#) to do this.



### Procedure

1. Find out the case models that are deployed by calling [getCaseModelBasicInfo](#).

If you do not specify an application name, all case models are returned.


Information about the deployed case models is returned, for example, case model ID and major version number.

2. Use [getCaseModel](#) to retrieve the database scripts.

Specify the name of the application whose scripts you want to retrieve. If you do not specify an application name, all database scripts are retrieved.



The response returns:

- the case model details. For example, `appName`, `caseModelID` and `currentVersion`.
  - the version number of the currently installed case model. If a case model is installed, it means all the database scripts have been run and case data may exist.
  - The last time the case model has been updated and the date and time that this occurred. After initial deployment, it shows the deployment time.
  - The `CREATE` database script. If the application has been updated, the `UPDATE` database script is also returned. If the case model is in the `INSTALLED` state, the `DROP` script is also returned.
  - Whether or not the case model is frozen and the operation details. For example, who performed the operation, when and why.
  - The case model's state.
3. (Optional) Use [updateDBScripts](#) to make any required changes to the database scripts.
- 

Although editing the scripts is possible, this not recommend unless it is essential and there are only a few changes that you are allowed to make.
4. Once you have verified your database scripts, you must manually execute them on your database to generate the case model database schema. Once the case model database schema has been generated, you must notify the system using [notifyCreated](#).
5. You can freeze case data models so that no further changes can be made to it. In other words, the case model cannot be upgraded or undeployed. This prevents another database administrator from making any further changes to the case data model. To do this, use [notifyFreeze](#).

## Forms

A client application may need to display a form to a user when that user opens a work item. opens a work item that starts a pageflow, or starts a business service.

The client application can render two types of forms:

- a TIBCO form, which provides a web-based user interface to the work item data. See [TIBCO Forms](#).
- a custom form developed as part of the client application. See [Custom Forms](#).

## TIBCO Forms

TIBCO forms provide web-based user interfaces for business processes. At design-time, process analysts and solution designers can use TIBCO Business Studio to design forms and associate them with user tasks in business processes or pageflow processes.

See the TIBCO Business Studio documentation for more information.

TIBCO Business Studio generates run-time implementations of these forms based on the use of channel types and presentation channels:

- A *channel type* defines a method employed to deliver and display a form to a user. The specification of a channel type defines:
  - a delivery mechanism (for example, web client or email).
  - a rendering technology (Google Web Toolkit (GWT)).
- A *presentation channel* is a container for a selection of available channel types; it defines the ways in which a form can be delivered and presented to users.

TIBCO Business Studio automatically generates an implementation of each form defined in a project for each channel type defined in the project's presentation channel(s).





If no form has been defined for a user activity, default form implementations are automatically generated for the default presentation channel.

When a project is deployed, the generated form artifacts are deployed as well. (**bpmresources** is the name of the BPM runtime component that provides access to the form resources of deployed BPM applications.)

A client application can use the [WorkPresentationService](#) to access the form artifacts that have been deployed, and to display an appropriate TIBCO form for a work item or pageflow/business service page.

## Binding the Client Application to a Channel

A client application must be bound to a single channel (identified by a `channelId`).

The selected channel must be one that is used by the presentation channel of any TIBCO Business Studio process that the client application will process.



The binding can be done declaratively or programmatically.

The runtime uses this channel to select the correct form artifact to use to display a form.

## Identifying the Client Channel in a Service Call

Most `WorkPresentationService` calls require you to identify the channel to be used.

The call from [WorkPresentationService](#) passes the following items in the request:

- **channelId** is the identifier of the channel to which the client is bound.
- **channelType** is an enumerated value associated with that channel type.

The following table shows:

- the channel types supported by TIBCO Business Studio.
- the format of different `channelIds` used to identify individual channels of each presentation type. *ChannelName* is the label of the parent presentation channel - for example, `DefaultChannel` or `CustChannel1`.
- the `channelType` enumeration that should be specified for each `channelId`.

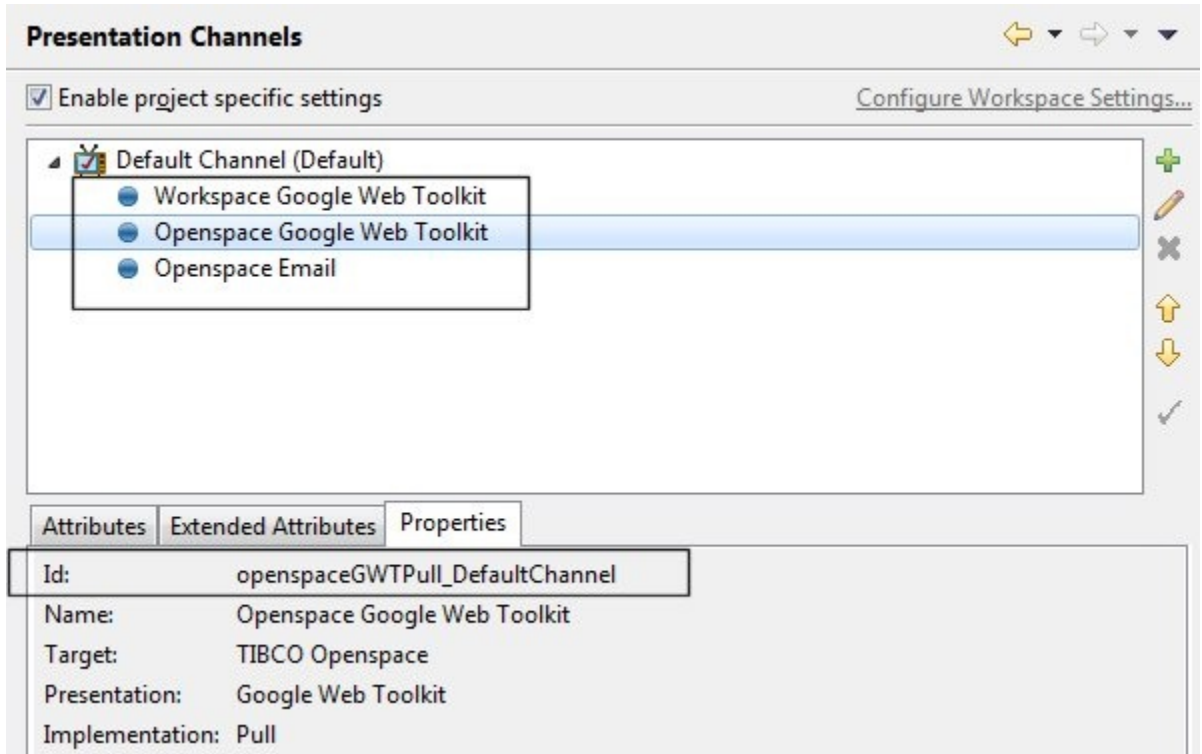
TIBCO Business Studio channel type	channelId	channelType
Openspace Google Web Toolkit	<code>openspaceGWTPull_ChannelName</code>	<code>openspaceChannel</code>
Workspace Google Web Toolkit	<code>GIGWTPull_ChannelName</code>	<code>openspaceChannel</code>
Openspace Mobile	<code>MobileGWTPull_ChannelName</code>	<code>MobileChannel</code>
Openspace Email	<code>openspaceEmailPush_ChannelName</code>	<code>EmailChannel</code>
Workspace Email	<code>EmailGIPush_ChannelName</code>	<code>EmailChannel</code>



The `channelType` enumerations `JSPChannel`, `PageflowChannel` and `RssChannel` are currently not supported.

You cannot obtain `channelIds` programmatically. Instead, you must obtain them directly from TIBCO Business Studio. The following screenshot shows:

- the channel types used by the default presentation channel. (Form artifacts are automatically generated for each of these channel types.)
- how to obtain the channelId of a particular channel. In this example, openspaceGWTPull\_DefaultChannel identifies the channel in the Default Channel presentation channel that uses the Openspace Google Web Toolkit channel type.



For more information, see "Using Presentation Channels to Display Tasks to Users" in *TIBCO Business Studio BPM Implementation*.

## Rendering a TIBCO Business Studio Form

The Forms Runtime Adapter is a JavaScript API provided as part of the runtime.

A client application can use the Forms Runtime Adapter to display TIBCO Business Studio forms for work items, pageflow pages and business service pages.

To display a TIBCO form, the client application must:

1. Load the Forms Runtime Adapter in the browser control. The Forms Runtime Adapter provides an API that will display the form.
2. Pass in the URL of the JSON representation of the form.
3. Pass in the JSON payload (representing the payload associated with the user task of the displayed form) that will be used to populate the form.
4. Create action handlers for the **Submit**, **Cancel** and (if required) **Close** form actions. These will be used to notify the client application when that action takes place, passing back any updated data.
5. Render the form using the Forms Runtime Adapter loadForm method.



For more information about how to perform these steps, see the following references:

- [Integrating TIBCO Forms with Custom Client Applications](#) provides detailed information about the Forms Runtime Adapter and how to use it.
- [Rendering a TIBCO Business Studio Form](#) provides an example of how to use the Forms Runtime Adapter in a custom client application that uses the web service API to access BPM services.

The Forms Runtime Adapter handles the user's interaction with the form. When the user submits, closes or cancels the form, the Forms Runtime Adapter notifies the client application that this has happened (using the callback handler registered) and returns the data from the form.



For more information about how to integrate use of the Forms Runtime Adapter with calls to the BPM services, see:

- [Displaying a Work Item Form](#)
- [Displaying a Form in a Pageflow](#)
- [Displaying a Form in a Business Service](#)

## Custom Forms

Development and use of custom forms is entirely the responsibility of the client application.

The same principles apply as for using TIBCO Business Studio forms, but you should note the items described in the following sub-topics.

## Presentation Details

Use the `WorkPresentationService`, `PageFlowService`, and `BusinessService` APIs to display forms for work items and pageflow/business service page activities.

Note, however, you:

- should ignore all elements concerned with presentation channel artifacts, as these can only be used to display TIBCO Business Studio forms.
- cannot use the presentation details returned by `openWorkItem` when [Displaying a Work Item Form](#) or [Displaying a Form in a Pageflow](#).

See also: [WorkPresentationService](#), [PageFlowService](#) and [BusinessServiceService](#).

## Data Format and Structures

You should use XML instead of JSON as the data payload format. (The JSON returned by the BPM runtime is tailored for use with the Forms Runtime Adapter.)

Your application must interpret and render the XML data payloads returned by the BPM runtime, which will require knowledge of their underlying data structures. You cannot obtain this information programmatically. Instead, you must obtain it from TIBCO Business Studio:

- Export a **Work Data Model** for every project that the client application will need to handle. This model contains a number of artifacts, including:
  - a work type definition file (`wt.xml`), which defines the data structures of all work types used in the project. (Each user activity in a business process has a work type associated with it.)
  - a pageflow activity definition file (`pfActivityType.xml`), which defines the data structure for each user activity in a pageflow process in the project.



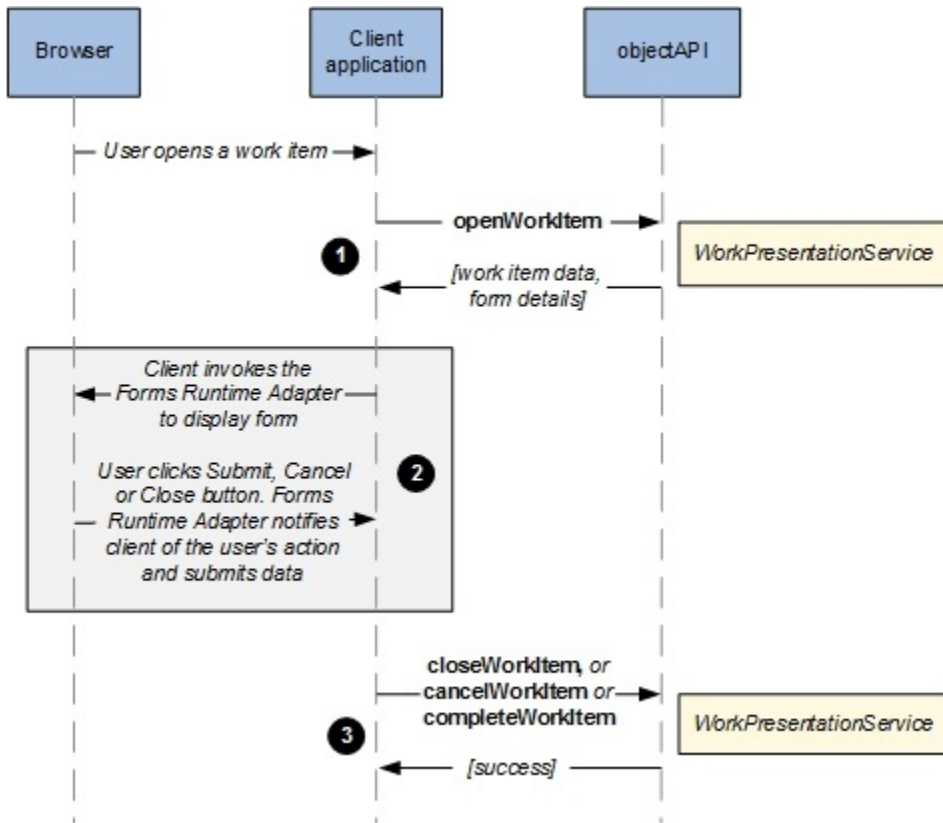
For more information about how to generate a Work Data Model, see "Exporting Projects to a Work Data Model" in *TIBCO Business Studio Process Modeling*.

- Note any formal parameters required by business processes or pageflow processes. When you start or inject data into a pageflow process or business service that requires formal parameters, you must

specify the data for those formal parameters. (See the following element definitions: [startPageFlow](#), [injectPageFlowEvent](#), [startBusinessService](#), [injectBusinessServiceEvent](#).)

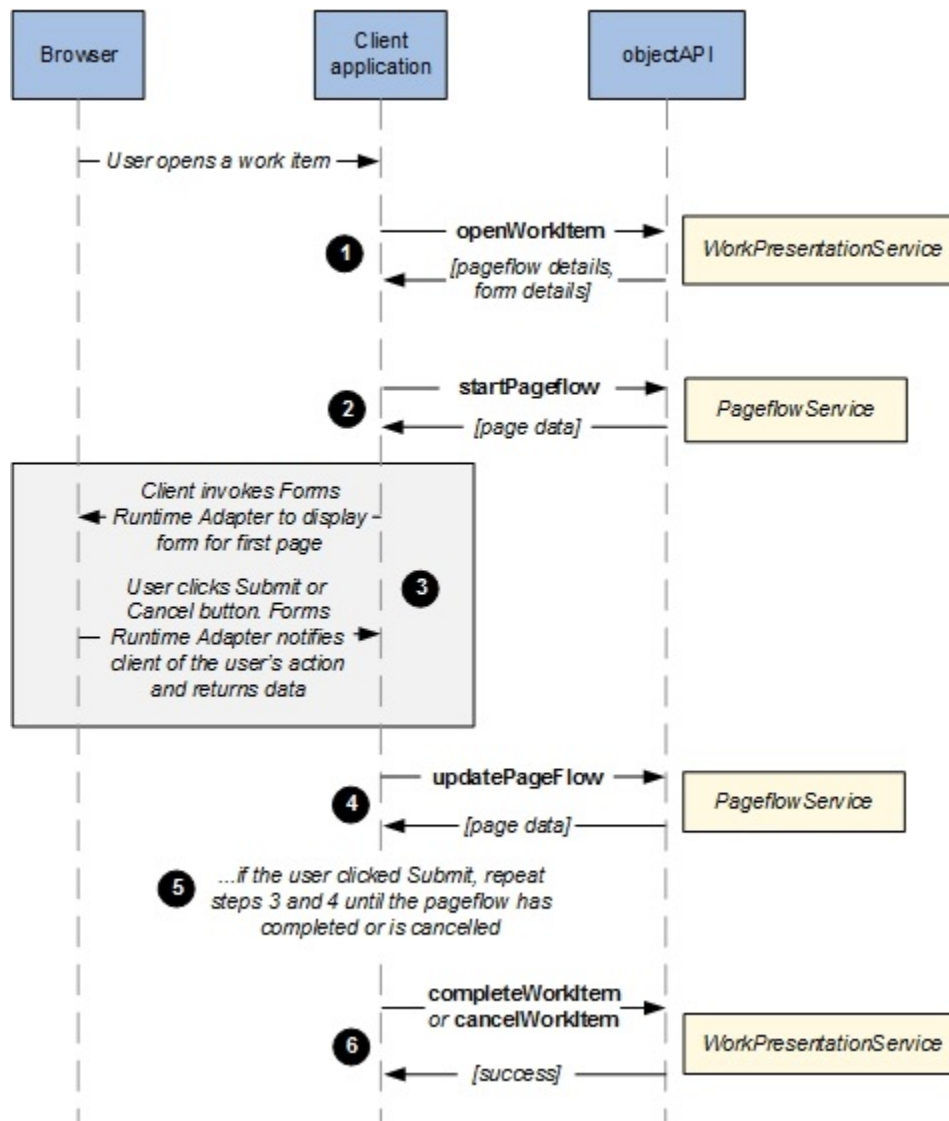
## Displaying a Work Item Form

This topic provides an example that shows the sequence of calls a client application should make to display a work item form.



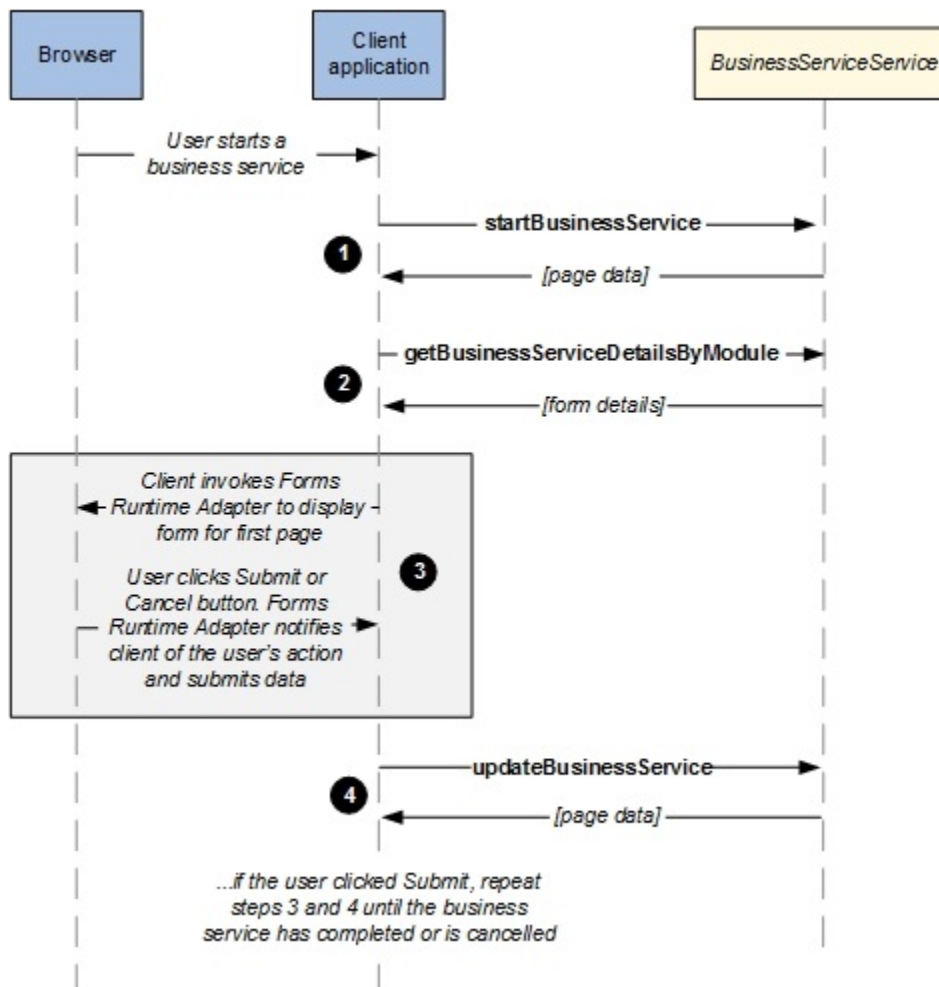
## Displaying a Form in a Pageflow

This topic provides an example that shows the sequence of calls a client application should make to display a form in a pageflow.



## Displaying a Form in a Business Service

This topic provides an example that shows the sequence of calls a client application should make to display a form in a business service.



## Integrating TIBCO Forms with Custom Client Applications

TIBCO forms are a part of projects are deployed to the runtime.

The key top-level components that are involved in making TIBCO Forms accessible from custom client applications are as follows:

- **BPM Resources component** - This is a component provided by the BPM runtime. This component fulfills two functions:
  - First, it provides access to the [Forms Runtime Adapter](#). This is a JavaScript API that client applications can load in the browser and use to render forms in their applications. See [Injecting the Forms Runtime Adapter in the Browser](#) to know about loading the forms runtime adapter.
  - Second, this component provides access to the form resources of BPM applications deployed by the user. Each form that is deployed makes use of a number of resources including: the form definition, CSS, resource bundles for localization, and JavaScript files that provide form actions as well as implementations of Business Object Model objects referenced by the form.
- **BPM public web service API** - BPM exposes its functionality through comprehensive work management and process management APIs, which are exposed as web services. The main services

that are used in the form applications are: [WorkPresentationService](#), [PageFlowService](#), and [BusinessServiceService](#).

A client application can access the web services using either the BPM public web service API, the Java Service Connector API, or the REST API.

- **Client Application** - The client application is typically a web application that bundles together the non-Form UI resources that are used by the application, and a controller servlet that acts as the conduit between the browser and the BPM public web service API.

## Injecting the Forms Runtime Adapter in the Browser

The Forms Runtime Adapter is hosted by the BPM runtime. To use this, you first need to load the Forms Runtime Adapter file in the page. This is typically done via a `<script>` tag in the `<head>` section of the HTML document.

```
<script src="http://<host-name>:<port>/bpmresources/formsclient /
formsclient.nocache.js"/>
```

See [Rendering a TIBCO Business Studio Form](#) for more information. Once the JavaScript API loading is complete, it notifies the client application by invoking a function called `onTIBCOFormRunnerLoad`. The client application can define this function on the page and receive notification of the availability of `com.tibco.forms.client.FormRunner`.

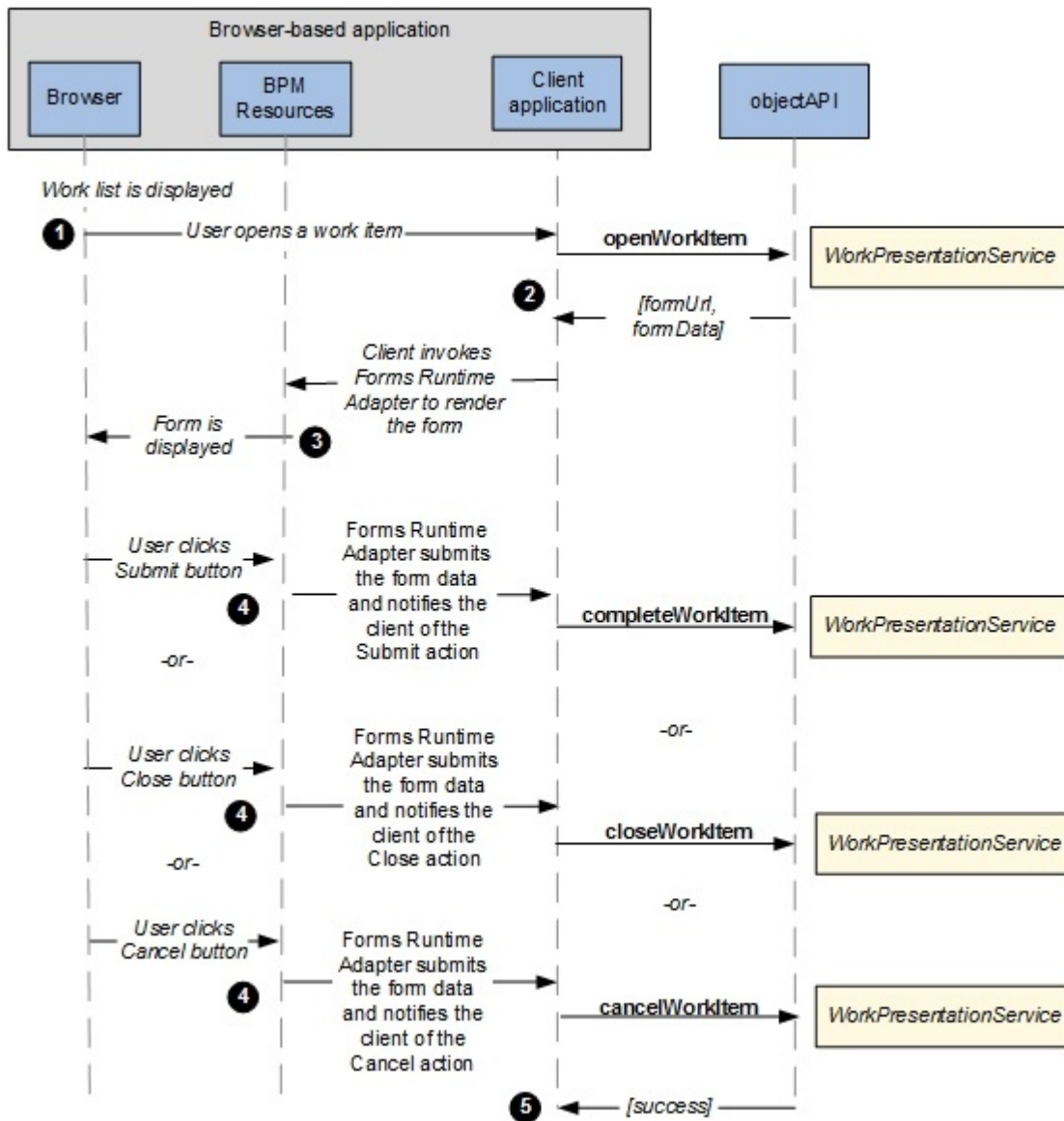
For example:

```
function onTIBCOFormRunnerLoad() {
 // Forms Runtime Adapter is now available for use on the page.
 // com.tibco.forms.client.FormRunner.loadForm() can be accessed from now on
 // to load the form.
}
```



## Typical Flow of Events

This topic describes the typical flow of events when a user opens a work item from the work list displayed in the browser of the client application.



The details of the events are as follows:

1. To open a work item, the client application invokes the [openWorkItem](#) function in the [WorkPresentationService](#).
2. In response to the [openWorkItem](#) request, the `formUrl` and `formData` are returned to the client application. The client application decides the `locale` for rendering the form and provides the identifier of the DOM node under which the form will be rendered. The `channelId` is the identifier of the presentation channel to which the client application is bound, for example, `GIGWTPull_DefaultChannel` or `openspaceGWTPull_DefaultChannel`. For more information, see [Binding the Client Application to a Channel](#).

The client application invokes the `FormRunner.loadForm()` method, which accepts the above values along with two arguments: `onSuccess` and `onError`. See [com.tibco.forms.client.FormRunner](#) for details of the `FormRunner.loadForm()` method.



The following code snippet shows how the form is loaded using the `FormRunner.loadForm()` method.

```
var formURL; // Form URL obtained from the work item.
var formData; // The initial form data obtained from the work item.
var bomJSPath; // BOM JavaScript root path obtained from the workitem.
var locale = "en_US"; // locale to use
var parentId; // Identifier of a node to which the form is added.
var submitHandler = function(actionName, form) {
 var formData = form.getSerializedParameters();
 // submit the work item
 form.destroy();
};
var closeHandler = function(actionName, form) {
 // close the work item
 form.destroy();
};
var cancelHandler = function(actionName, form) {
 // cancel the work item
 form.destroy();
};
var onSuccess = function(form) {
 form.setActionHandler(com.tibco.forms.client.Form
 .ACTION_SUBMIT, submitHandler);
 form.setActionHandler(com.tibco.forms.client.Form
 .ACTION_CLOSE, closeHandler);
 form.setActionHandler(com.tibco.forms.client.Form
 .ACTION_CANCEL, cancelHandler);
};
var onError = function(e) {
 alert("An error occurred while loading the form: " + e);
};
com.tibco.forms.client.FormRunner.loadForm(formURL, formData, bomJSPath,
 locale, parentId, onSuccess, onError, JSONP);
```



Avoid using the `com.tibco.forms.client.FormRunner.loadForm()` method on the page `onLoad` event, as the `com.tibco.forms.client.FormRunner` class might not be fully loaded and the API methods being used might not be available. To avoid these errors, the client application can define the `onTIBCOFormRunnerLoad` function on the page and can be notified about the availability of `com.tibco.forms.client.FormRunner`. For more details, see [Injecting the Forms Runtime Adapter in the Browser](#).

3. The form is displayed in the browser. The `FormRunner.loadForm()` method is asynchronous, so any post-processing that is done on the loaded form object happens within the `onSuccess` callback handler. In the above code snippet, three action handlers are set that take care of the submit, close, and cancel operations that are provided on most forms.
4. When the form is submitted, the `submitHandler` handles the submit action. In response to form submission, `form.getSerializedParameters()` method retrieves the `formData`. See [com.tibco.forms.client.Form](#) for details of the `form.getSerializedParameters()` method. This method returns a JSON (JavaScript Object Notation) serialization of the data within the form.
5. After successful submission of the form, the client application invokes [completeWorkItem](#) to pass the form data back to the [WorkPresentationService](#). This function is used to update the work item.

## iframe Integration

A form can be loaded in an inline frame (`iframe`) in a custom application. The `loadForm` APIs in the `com.tibco.forms.client.FormRunner` class support the parent node ID of an `<iframe>` element.

To make this integration easy, the file `IFrameSource.html` is provided in the forms client (which is bundled inside the [Forms Runtime Adapter](#)). An `iframe` in the custom application can use the following source URL:

```
<iframe class="tf-form-not-loaded" id="formContainer" src="/bpmresources/
formsclient/IFrameSource.html">
</iframe>
```

Using `IFrameSource.html` is optional. You can provide your own `iframe` source page if desired.

The forms runtime loaded in a parent window talks to the forms runtime loaded in the `iframe`. For this communication to happen, the `formsclient.nocache.js` file must be loaded. The `IFrameSource.html` page loads `formsclient.nocache.js` by default. However, if you provide your own `iframe` source page, ensure that it loads `formsclient.nocache.js` (using a `<script>` tag in the HTML; see [Injecting the Forms Runtime Adapter in the Browser](#)).

When the form is loaded in the `iframe`, the Forms Runtime Adapter removes the class selector `tf-form-not-loaded` from the `<iframe>` element and adds another class, `tf-form-loaded`. Similarly, when the form is destroyed by the Cancel, Close, or Submit actions, the Forms Runtime Adapter adds the class `tf-form-not-loaded` back to the `<iframe>` element. This can be used to control the visibility of the `iframe` within a custom application.

Note that changing the CSS class name based on whether a form is currently loaded or not is taken care of by the `formsclient` loaded on the parent window (therefore, your `iframe` element can initially have the class `tf-form-not-loaded`, which is automatically removed when a form is actually loaded).

The area of the `iframe` where the form is loaded, is based on the following conditions:

- if the same `parentNodeId` is available within the `iframe`, the form is loaded within the `iframe`.
- if the `iframe` has an element with the ID `tfFormContainer`, the form is loaded in that element.
- if both of the above are missing, the form is loaded directly under the `body` element within the `iframe`.

## Forms Runtime Adapter

The Forms Runtime Adapter provides access to methods for instantiating, accessing, and measuring the performance of forms.

The Forms Runtime Adapter defines the following three classes:

- [com.tibco.forms.client.FormRunner](#)
- [com.tibco.forms.client.Form](#)
- [com.tibco.forms.client.LoadStat](#)

### ***com.tibco.forms.client.FormRunner***

The `com.tibco.forms.client.FormRunner` class provides static utility methods for instantiating the `Form` class in the custom client application. It also provides access to the Forms logger, which can be used even when no forms are actually loaded.

For more details on the `logger()` method, see the table on the bottom of this page.

The following methods are supported:

- `loadForm()`
- `loadFormWithRemoteData()`
- `renderStaticView()`

The `loadForm()` and `loadFormWithRemoteData()` methods have the same set of input parameters but they differ in the way the initial data are passed. Both the methods are void.

- In `loadForm()`, the initial data are passed directly as a JSON (JavaScript Object Notation) string.
- In `loadFormWithRemoteData()`, a URL of the initial JSON data is passed using the `initDataURL` parameter. The FormRunner retrieves the data from the specified URL.

The details of these methods are as follows:

*com.tibco.forms.client.FormRunner Class*

Method	Description
<pre>loadForm( String url, String initialData, String bomJSPPath, String locale, String parentNodeId, Function onSuccess, Function onError, Boolean JSONP)</pre>	<p>Loads the form at the specified URL. The parameter details are as follows:</p> <ul style="list-style-type: none"> <li>• <code>url</code> - Specifies the URL to the form JSON representation, e.g. "http://&lt;hostname:port&gt;/bpmresources/1.0.0.201107291435/openspaceGWPull_DefaultChannel/FindAddress/GetAddress/Getuserdetails/Getuserdetails.gwt.json"</li> <li>• <code>initialData</code> - Specifies the JSON representation of the initial data that are provided to the form.</li> <li>• <code>bomJSPPath</code> - Specifies the root folder path used for loading the BOM JavaScript files used by the form, e.g. "http://&lt;hostname:port&gt;/bpmresources/1.0.0.201107291435"</li> <li>• <code>locale</code> - Specifies the locale to be used in the form runtime with format &lt;lc&gt;[_&lt;CC&gt;] where [] denotes optionality, e.g. "en_US". The locale needs to be represented such that &lt;lc&gt; is a valid two-character lowercase ISO-639 language code and if present the optional &lt;CC&gt; is a valid two-character uppercase ISO-3166 country code. Both '_' and '-' are supported as delimiters.</li> <li>• <code>parentNodeId</code> - Specifies the DOM identifier of the node to which the form is added. The value cannot be null.</li> </ul> <p>If you are using an <code>iframe</code>, you can pass the ID of the <code>iframe</code> element on the page. For more information, see <a href="#">iframe Integration</a>.</p> <ul style="list-style-type: none"> <li>• <code>onSuccess</code> - A function that is called once the form is successfully initialized. The Form object is passed into this function. This function can be used to add custom callback handlers that implement lifecycle events such as <code>submit</code>, <code>close</code>, and <code>cancel</code>.</li> <li>• <code>onError</code> - A function that is called if any errors are encountered in initializing the form. The function will receive any exception that was encountered during the initialization.</li> <li>• <code>JSONP</code> - Informs the Forms Runtime Adapter to use JSON with Padding (JSONP) when loading JSON resources. The default value is <code>false</code>. When the custom client and Forms Runtime Adapter are hosted on different servers, set the <code>JSONP</code> parameter as <code>true</code>. In this scenario, there are SOP (Single Origin Policy) issues while loading JSON resources. By using the JSONP technique, the JSON response is wrapped to a call to a function by the server and send to the client. A JSON resource can then be loaded using a script tag to avoid any SOP violations.</li> </ul>

Method	Description
<code>loadFormWithRemoteData(String url, String initDataURL, String bomJSPPath, String locale, String parentNodeId, Function onSuccess, Function onError, Boolean JSONP)</code>	<p>Loads the form at the specified URL. The parameter details are as follows:</p> <ul style="list-style-type: none"> <li><code>url</code> - Specifies the URL to the form JSON representation, e.g. "http://&lt;hostname:port&gt;/bpmresources/1.0.0.201107291435/openspaceGWTPull_DefaultChannel/FindAddress/GetAddress/Getuserdetails/Getuserdetails.gwt.json".</li> <li><code>initDataURL</code> - Specifies the URL to the form initial data.</li> <li><code>bomJSPPath</code> - Specifies the root folder path used for loading the BOM JavaScript files used by the form, e.g. "http://&lt;hostname:port&gt;/bpmresources/1.0.0.201107291435"</li> <li><code>locale</code> - Specifies the locale to be used in the form runtime with format &lt;lc&gt;[_&lt;CC&gt;] where [] denotes optionality, e.g. "en_US". The locale need to be represented such that &lt;lc&gt; is a valid two-character lowercase ISO-639 language code and if present the optional &lt;CC&gt; is a valid two-character uppercase ISO-3166 country code. Both '_' and '-' are supported as delimiters.</li> <li><code>parentNodeId</code> - Specifies the DOM identifier of the node to which the form should be added. The value cannot be null.</li> </ul> <p>If you are using an <code>iframe</code>, you can pass the ID of the <code>iframe</code> element on the page. For more information, see <a href="#">iframe Integration</a>.</p> <ul style="list-style-type: none"> <li><code>onSuccess</code> - A function that is called once the form is successfully initialized. The Form object is passed into this function. This can be used to add custom callback handlers that implement lifecycle events such as <code>submit</code>, <code>close</code>, and <code>cancel</code>.</li> <li><code>onError</code> - A function that is called if any errors are encountered in initializing the form. The function will receive any exception that was encountered during the initialization.</li> <li><code>JSONP</code> - Informs the Forms Runtime Adapter to use JSON with Padding (JSONP) when loading JSON resources. The default value is <code>false</code>. When the custom client and Forms Runtime Adapter are hosted on different servers, set the JSONP parameter as <code>true</code>. In this scenario, there are SOP (Single Origin Policy) issues while loading JSON resources. By using the JSONP technique, the JSON response is wrapped to a call to a function by the server and send to the client. A JSON resource can then be loaded using a script tag to avoid any SOP violations.</li> </ul>

Method	Description
<code>renderStaticView()</code>	<p>Renders a tree representation of the data provided in the <code>InitialData</code> parameter.</p> <p>The API has following parameters:</p> <ul style="list-style-type: none"> <li><code>initialData</code>: specifies the JSON representation of the initial data that are provided to the form.</li> <li><code>bomJSPath</code>: specifies the root folder path used for loading the BOM JavaScript files used by the form.</li> </ul> <p>For example: <code>http://&lt;hostname:port&gt;/bpmresources/1.0.0.201107291435</code></p> <ul style="list-style-type: none"> <li><code>locale</code>: specifies the locale to be used in the runtime form with the <code>&lt;lc&gt;[_&lt;CC&gt;]</code> format, where <code>[]</code> denotes optionality.</li> </ul> <p>For example: <code>"en_US"</code>. The locale needs to be represented such that <code>&lt;lc&gt;</code> is a valid two-character lower case ISO-639 language code, and, if present, the optional <code>&lt;CC&gt;</code> is a valid two-character upper case ISO-3166 country code. Both <code>'_'</code> and <code>'-'</code> are supported as delimiters.</p> <ul style="list-style-type: none"> <li><code>parentNodeId</code>: specifies the DOM identifier of the node to which the form must be added. The value cannot be null.</li> </ul> <p>If you are using an <code>iframe</code>, you can pass the ID of the <code>iframe</code> element on the page. For more information, see <a href="#">iframe Integration</a>.</p> <ul style="list-style-type: none"> <li><code>onSuccess</code>: is a function that is called after the form is successfully initialized. The Form object is passed into this function. It can be used to add custom callback handlers that implement lifecycle events such as submit, close, and cancel.</li> <li><code>onError</code>: is a function that is called on encountering any error in initializing the form. The function receives the exception encountered during the initialization.</li> <li><code>provideCloseAction</code>: if it is set to <code>true</code>, the form is rendered with a button that closes the form and cleans up any resources used. If it is set to <code>false</code>, then it is the responsibility of the containing application to clean up the form when it is no longer needed.</li> <li><code>JSONP</code>: informs the Forms Runtime Adapter to use JSON with Padding (JSONP) when loading JSON resources. The default value is <code>false</code>. When the custom client and Forms Runtime Adapter are hosted on different servers, set the <code>JSONP</code> parameter to <code>true</code>. In this scenario, there are SOP (Single Origin Policy) issues while loading JSON resources. By using the JSONP technique, the JSON response is wrapped as a function by the server and is sent to the client. A JSON resource can then be loaded using a script tag to avoid any SOP violations.</li> </ul>

For the list of language codes and country codes required for specifying the `locale` parameter, visit the following web sites:



- List of language codes - <http://www.loc.gov/standards/iso639-2/langhome.html>
- List of country codes - [http://www.iso.org/iso/country\\_codes/iso\\_3166\\_code\\_lists.htm](http://www.iso.org/iso/country_codes/iso_3166_code_lists.htm)

The details of the `logger()` method are as follows:

*com.tibco.forms.client.FormRunner.logger Class*

Method	Return Value	Description
<code>fatal(String message)</code>	Void	Logs the given messages at the fatal logging level.
<code>error(String message)</code>	Void	Logs the given messages at the error logging level.
<code>warn(String message)</code>	Void	Logs the given messages at the warn logging level.
<code>info(String message)</code>	Void	Logs the given messages at the info logging level.
<code>debug(String message)</code>	Void	Logs the given messages at the debug logging level.
<code>trace(String message)</code>	Void	Logs the given messages at the trace logging level.
<code>isFatalEnabled()</code>	Boolean	Checks whether the Fatal logging level is enabled. It returns <code>true</code> if the logging level is enabled, and <code>false</code> otherwise.
<code>isErrorEnabled()</code>	Boolean	Checks whether the Error logging level is enabled. It returns <code>true</code> if the logging level is enabled, and <code>false</code> otherwise.
<code>isWarnEnabled()</code>	Boolean	Checks whether the Warn logging level is enabled. It returns <code>true</code> if the logging level is enabled, and <code>false</code> otherwise.
<code>isInfoEnabled()</code>	Boolean	Checks whether the Info logging level is enabled. It returns <code>true</code> if the logging level is enabled, and <code>false</code> otherwise.
<code>isDebugEnabled()</code>	Boolean	Checks whether the Debug logging level is enabled. It returns <code>true</code> if the logging level is enabled, and <code>false</code> otherwise.
<code>isTraceEnabled()</code>	Boolean	Checks whether the Trace logging level is enabled. It returns <code>true</code> if the logging level is enabled, and <code>false</code> otherwise.

***com.tibco.forms.client.Form***

The `com.tibco.forms.client.Form` class provides access to the runtime form object. This object enables you to access panes and controls within the form, register handlers for form actions, and access data to be submitted back to the server.


The `com.tibco.forms.client.Form` class has six fields that are used for setting action handlers. It also implements the methods listed in the table below.

*com.tibco.forms.client.Form Class - Field Details*

Field	Data Type	Description
<code>ACTION_APPLY</code>	String	Identifies the "apply" action.



Field	Data Type	Description
<code>ACTION_CANCEL</code>	String	Identifies the "cancel" action.
<code>ACTION_CLOSE</code>	String	Identifies the "close" action.
<code>ACTION_RESET</code>	String	Identifies the "reset" action.
<code>ACTION_SUBMIT</code>	String	Identifies the "submit" action.
<code>ACTION_VALIDATE</code>	String	Identifies the "validate" action.

*com.tibco.forms.client.Form Class - Method Details*

Method	Return Value	Description
<code>destroy()</code>	Void	Removes the form from its container and also releases its resources. This can be called by the client application to close the form.
<code>getFactory()</code>	Object	<p>Returns the factory object for the given form. This provides access to the BOM JavaScript factories associated with the BOM types used by this form, as documented under the <i>factory</i> variable available within form action and validation scripts.</p> <div>  <p>This method is not supported when an <code>iframe</code> is used to load the form using the built-in <code>iframe</code> integration support.</p> </div>

Method	Return Value	Description
<code>getLoadStats()</code>	LoadStat	<p>Returns an array of LoadStat objects. Each statistic represents the measurement of a particular phase of the form load. This can be used by applications to report this information in the user interface or otherwise log the information.</p> <p>To enable collecting load statistics at runtime, the URL used by the client application to load the form should contain the parameter <code>tibco_instr</code> with a value <code>true</code>. Otherwise <code>getLoadStats()</code> method would return an empty array.</p> <p>This method takes an optional callback function as an argument to support <code>iframe</code> integration, where the method returns asynchronously:</p> <pre>getLoadStats(Function callback)</pre> <p>The callback function is optional for non-<code>iframe</code> mode, but required for <code>iframe</code> mode.</p> <p>The method still returns the value in non-<code>iframe</code> mode for backward compatibility. However, use of the callback function is recommended for both <code>iframe</code> and non-<code>iframe</code> modes. For example:</p> <pre>form.getLoadStats(function(loadStats){   for (idx in loadStats)     alert(loadStats[idx].label + ": " +       (loadStats[idx].endTime - loadStats[idx].startTime) + "       ms"); });</pre> <p>To use the <code>getLoadStats()</code> method, the client application needs to subscribe to the TIBCO PageBus event <code>'com.tibco.forms.form.loaded'</code>. See <a href="#">com.tibco.forms.client.LoadStat</a> for more details.</p>
<code>getLocale()</code>	String	<p>Returns the string representation of the locale being used to render the form.</p> <p>This method takes an optional callback function as an argument to support <code>iframe</code> integration, where the method returns asynchronously:</p> <pre>getLocale(Function callback)</pre> <p>The callback function is optional for non-<code>iframe</code> mode, but required for <code>iframe</code> mode.</p> <p>The method still returns the value in non-<code>iframe</code> mode for backward compatibility. However, use of the callback function is recommended for both <code>iframe</code> and non-<code>iframe</code> modes. For example:</p> <pre>form.getLocale(function(locale) {   alert("Form Locale is: " + locale); });</pre>



Method	Return Value	Description
<code>getPackage()</code>	Object	<p>Returns the object that provides access to the BOM JavaScript package definitions associated with the BOM types used by this form, as documented under the <i>pkg</i> variable available within form action and validation scripts.</p> <div>  <p>This method is not supported when an <code>iframe</code> is used to load the form using the built-in <code>iframe</code> integration support.</p> </div>
<code>getResource()</code>	Object	<p>Returns the object that provides access to the resource bundles associated with this form, as documented under the <i>resource</i> variable available within form action and validation scripts.</p> <div>  <p>This method is not supported when an <code>iframe</code> is used to load the form using the built-in <code>iframe</code> integration support.</p> </div>
<code>getSerializedParameters()</code>	String	<p>Returns a JSON representation of the data being managed by the form. This is typically called from a submit handler in order to send the final results back to the server.</p> <p>This method takes an optional callback function as an argument to support <code>iframe</code> integration, where the method returns asynchronously:</p> <pre>getSerializedParameters(Function callback)</pre> <p>The callback function is optional for non-<code>iframe</code> mode, but required for <code>iframe</code> mode.</p> <p>The method still returns the value in non-<code>iframe</code> mode for backward compatibility. However, use of the callback function is recommended for both <code>iframe</code> and non-<code>iframe</code> modes. For example:</p> <pre>form.getSerializedParameters(function(data) {     alert('form data: ' + data); });</pre>

Method	Return Value	Description
<code>setActionHandler(String actionName, Function handler)</code>	Void	<p>Adds a handler to the form that is invoked when the specified action is invoked in the form. Note that any handler already registered for this action will be replaced by this handler. The parameter details are as follows:</p> <ul style="list-style-type: none"> <li><code>actionName</code> - Used to specify the name of the</li> <li><code>action</code> (For example, <code>ACTION_CLOSE</code>). If the action is <code>ACTION_SUBMIT</code>, then all the validations in the form will be invoked prior to invoking the callback handlers. If any of the validations fail, then the callback handler will not be invoked.</li> <li><code>handler</code> - This is the function that is invoked for the specified <code>actionName</code>. The form may have only one handler for each action. If this method is called more than once for the same <code>actionName</code>, the handler set previously will be replaced. Passing in <code>null</code> for this parameter will remove the handler for this particular action. The method signature of the handler function has two arguments: a string for the <code>actionName</code> and the form object.</li> </ul>
<code>setLocale(String locale)</code>	Void	Sets the string representation of the locale currently being used to render the form. For example, "en" or "en_US".

### ***com.tibco.forms.client.LoadStat***

The `com.tibco.forms.client.LoadStat` class helps you to measure the load-time performance of the form. Each statistic has three fields: a description, a start time, and an end time. The times are measured in milliseconds from when the form load began.

To use `com.tibco.forms.client.LoadStat`, the form loading has to be complete, including data loading and invocation of form open rules. The TIBCO PageBus event '`com.tibco.forms.form.loaded`' is published when the form loading is complete. The client application needs to subscribe to this event in order to be notified that the form loading is complete and then use the `getLoadStats()` method.

For example:

```
var callback = function(subject, form) {
 var stat = form.getLoadStats();
 for (var i=0; i<stat.length; i++) {
 alert(stat[i].label+" | End Time : "
 + stat[i].endTime+" | Begin Time : "
 + stat[i].startTime);
 }
}
PageBus.subscribe('com.tibco.forms.form.loaded', null, callback);
```

The details of the fields are as follows:

### *com.tibco.forms.client.LoadStat Class - Field Details*

Field	Data Type	Description
<code>label</code>	String	Describes what is being measured by this particular loadstat.
<code>startTime</code>	Number	The time, in milliseconds, when this particular measurement phase began. This time is relative to the instant when the form load began.
<code>endTime</code>	Number	The time, in milliseconds, when this particular measurement phase ended. This time is relative to the instant when the form load began.

## Events

Actions that take place in a BPM application are recorded as *events*. These include things like logging in, opening a work item, submitting a work item, suspending a process instance, etc.

Functions are provided to work with events using the [AuditService](#) API.

## Executing a Query

Events are viewed using *queries*.

Queries can be executed as either *dynamic* or *registered* queries:

- **Dynamic query:** A query object is passed in at execution time, where it is parsed and executed by Event Collector.  
This approach is recommended for queries which are either performed infrequently, or which change often when called, because the filter expression has to be parsed whenever the query is run.
- **Registered query:** A query object is defined and registered with Event Collector, at which point it is parsed and stored. Whenever the query is subsequently called, a reference is made to the passed query, which improves efficiency both in data transport and in parsing costs.

This approach is recommended for queries that are performed repeatedly over a period of time, because the filter expression only needs to be parsed once, but can be run many times.

The **target** option specified in a query (in the Query object) defines the Event Collector database table against which the query will execute, which in turn determines the type of information to be retrieved. See [Using Attributes in Query Filters](#) for more information, and see the *TIBCO ActiveMatrix BPM Event Collector Schema Reference* for details of the schema for all the Event Collector database tables.

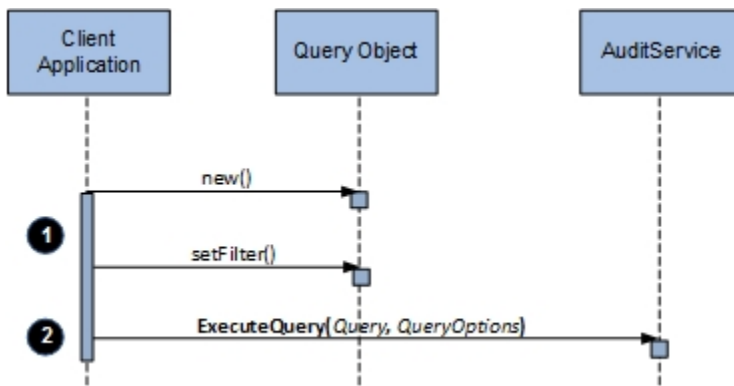


Be aware that queries against the Event Collector tables can take a long time to execute as the Event Collector tables can become very large.

## Dynamic Queries

This topic describes the execution of a dynamic query.

The following diagram shows how to execute a dynamic query.



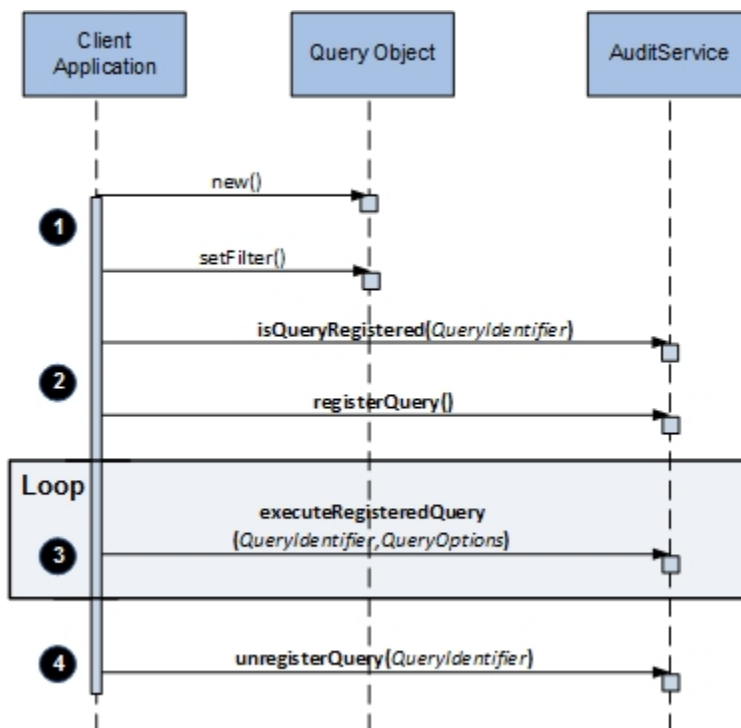
The client application does the following:

1. It instantiates a query object, which defines the filter expression and the list of attributes that are to be returned.
2. It performs an [executeQuery](#) function, which returns the list of events matching the filter criteria. The requested attributes and values are returned for each event.

### Registered Queries

This topic describes the execution of a registered query.

The following diagram shows how to create and use a registered query.



The client application does the following:

1. It instantiates a query object, which defines the filter expression and the list of attributes that are to be returned.
2. It uses [isQueryRegistered](#) to determine if the query is already registered, and if not, it calls the [registerQuery](#) operation to register the query, using the defined query object. Event Collector parses and stores the query for later use.

3. It performs an [executeRegisteredQuery](#) function each time the query is needed. *QueryOptions* can be used to dynamically alter the query parameters without having to alter and re-register the query itself. (See [Using Late-Bound Parameter Literals](#).) The function returns the list of events matching the filter criteria. The requested attributes and values are returned for each event.



The *QueryIdentifier* can identify the query using either its descriptive name (tag) or its GUID. Performance is improved when using GUID, so when calling a query many times it is recommended to look up the query's GUID using the [lookupQueryByTag](#) function.

4. It performs an [unregisterQuery](#) function to remove the query when it is no longer needed. This removes the query from the Event Collector database table.

## Correlated Events

Correlated events are those events in which the *CorrelationId* is the same—events with the same *CorrelationId* are all part of the same service call.

When a query is executed with the [executeQuery](#) function, or registered with the [registerQuery](#) function, the **correlate** parameter can be used to request that the query return not only the events that match the query, but also all *correlated* events.

This ability to return correlated events allows you to see all events that have been triggered within a service call. Note, however, that this can cause a large amount of data to be returned—use it only when necessary.

## Defining Query Filter Strings

Some *AuditService* functions can include an expression defining filter criteria to be applied to the query.

Those functions are:

- [executeQuery](#) (Request message) - Execute the specified query against the Event Collector database.
- [registerQuery](#) (Request message) - Register a query with Event Collector, so that it can be subsequently executed using the [executeRegisteredQuery](#) function.
- [getChart](#) (Request message) - Group audit information for one or more attributes for a specific object, for example, process instances, process templates, work items, case data and events.
- [getChartData](#) (Request message) - Return audit information for certain criteria relating to an object, for example, process instances, process templates, work items, case data and events.

## Filter Expression Format

The filter expression must use specific syntax.

The required syntax is:

```
(operand operator operand [operator operand...])
```

All queries must be supplied on a single line. All non-quoted whitespace will be ignored.

## Functions

This topic describes the functions that can be used in query filters.

Event Collector has some functions that can be used with the *operand operator filter expression* syntax. See [Filter Expression Format](#).

There are three types of functions:

- Functions that return data for commonly used search criteria.
- Functions that return audit trails for common objects, for example, cases, work items or process instances.

- Functions that return comments stored against objects, for example, cases, work items or process instances.

### Functions for Common Search Criteria

The format of a function described in the table below is:

*function(attribute)*

where:

- *function*: one of the functions described in the table below.
- *attribute*: identifies the type of information to be retrieved. The type of object (for example, process instance or work item) defines the Event Collector table against which the query will execute. The attribute you include with the function depends on the type of object you are interested in. See [Using Attributes in Query Language](#).

Function	Description
<i>isSet(attribute)</i>	Returns all events for the named attribute that has a value of NULL in the Event Collector audit table you are querying. This is useful for data that are returned even if the objects are deleted. For example, case objects as audit events linked to case objects are returned even if the case objects are deleted.
<i>currentDay(attribute)</i>	Returns all events for the named attribute that has a value in the Event Collector audit table you are querying occurring today. Attribute must be of <code>DateTime</code> type.
<i>currentWeek (attribute)</i>	Returns all events for the named attribute that has a value in the Event Collector audit table you are querying occurring this week. Attribute must be of <code>DateTime</code> type.
<i>currentMonth(attribute)</i>	Returns all events for the named attribute that has a value in the Event Collector audit table you are querying occurring this month. Attribute must be of <code>DateTime</code> type.
<i>currentYear(attribute)</i>	Returns all events for the named attribute that has a value in the Event Collector audit table you are querying occurring this year. Attribute must be of <code>DateTime</code> type.
<i>currentUser(attribute)</i>	Returns all events for the named attribute that has a value in the Event Collector audit table you are querying for the currently authenticated user.

## Functions for Common Objects

Function	Description
<code>processInstance(instanceId, details)</code>	<p>Returns the audit trail for the specified process instance where:</p> <ul style="list-style-type: none"> <li>• <b>instanceId</b>: identifies the process instance ID whose events you want to audit. Can be obtained from <a href="#">listProcessInstances</a>, <a href="#">queryProcessInstances</a> or <a href="#">queryProcessInstancesAlt</a>.</li> <li>• <b>details</b>: Not supported. Pass as an empty string.</li> </ul>
<code>workItem(workItemId, details)</code>	<p>Returns the audit trail for the specified work item where:</p> <ul style="list-style-type: none"> <li>• <b>workItemId</b>: identifies the work item ID whose events you want to audit. Can be obtained from <a href="#">getWorkListItems</a>.</li> <li>• <b>details</b>: Not supported. Pass as an empty string.</li> </ul>
<code>caseReference(caseRef, details)</code>	<p>Returns the audit trail for the specified case object where:</p> <ul style="list-style-type: none"> <li>• <b>caseRef</b>: The reference to the case object. You can obtain case references using <a href="#">findAllCases</a>.</li> <li>• <b>details</b>: Not supported. Pass as an empty string.</li> </ul>
<code>caseDetails(caseClass, modelMajorVersion, objectId, details)</code>	<p>Returns the audit trail for the specified case object where:</p> <ul style="list-style-type: none"> <li>• <b>caseClass</b>: The fully qualified name of the case class. For example, <code>com.example.gddemo.Car</code>. You can obtain class names by using <a href="#">getCaseClassInfo</a>.</li> <li>• <b>modelMajorVersion</b>: The version number of the case model. For example, 2 or 2.0.0.</li> <li>• <b>objectId</b>: The case model ID. For example, 1.</li> <li>• <b>details</b>: Not supported. Pass as an empty string.</li> </ul>

## Functions for Comments

Function	Description
<code>processInstanceComments(instanceId)</code>	<p>Returns all comments stored against the specified process instance where <b>instanceId</b> identifies the process instance ID whose comments you want to return. <b>instanceId</b> can be obtained from <a href="#">listProcessInstances</a>, <a href="#">queryProcessInstances</a> or <a href="#">queryProcessInstancesAlt</a>.</p>
<code>workItemComments(workItemId)</code>	<p>Returns all comments stored against the specified work item where <b>workItemId</b> identifies the work item ID whose comments you want to return. <b>workItemId</b> can be obtained from <a href="#">getWorkListItems</a>.</p>

Function	Description
<code>caseReferenceComments(caseRef)</code>	Returns all comments stored against the specified case object where <b>caseRef</b> identifies the reference to the case object. You can obtain case references using <a href="#">findAllCases</a> .
<code>caseDetailsComments(caseClass,modelMajorVersion,objectId)</code>	<p>Returns all comments stored against the specified case object where:</p> <ul style="list-style-type: none"> <li>• <b>caseClass</b>: The fully qualified name of the case class. For example, <code>com.example.gddemo.Car</code>. You can obtain class names by using <a href="#">getCaseClassInfo</a>.</li> <li>• <b>modelMajorVersion</b>: The version number of the case model. For example, 2 or 2.0.0.</li> <li>• <b>objectId</b>: The case model ID. For example, 1.</li> </ul>

### Example Queries

```
processInstanceComments('pvm:0a124') AND currentUser(tibco-admin)
```

Returns the comments stored against the specified process instance created by the currently authenticated user, in this case, tibco-admin.

### Operands

This topic lists the supported filter query operands.

Operand Type	Syntax	Description
Attribute name	<code>name</code>	The name of an attribute - see <a href="#">Using Attributes in Query Filters</a> for more information.
String literal	<code>'string'</code> <code>'str?ng'</code> <code>'str*ng'</code> <code>'str\?ng'</code> <code>'str\\ng'</code>	A string literal to be used for comparison.  See <a href="#">Using String Literals</a> for more information.
Integer literal	<code>12345</code>	A numeric literal.
Boolean literal	<code>TRUE</code> or <code>FALSE</code>	A boolean literal.
DateTime literal	<code>YYYY-MM-DDThh:mm:ss.SSS</code> or for a range: <code>{   [YYYY-MM-DDThh:mm:ss.SSS, YYYY-MM-DDThh:mm:ss.SSS} ! ]</code>	<p>A date-time literal consisting of the relevant components.</p> <p>For ranges, a curly bracket indicates an exclusive value; a square bracket indicates an inclusive value.</p> <p>See <a href="#">Using DateTime Literals</a> for more information.</p>
A query	<code>(query)</code>	A sub query to be evaluated before the main query.



Operand Type	Syntax	Description
Late-bound parameter literal	<code>::name</code>	A value which will be late bound to the expression.  See <a href="#">Using Late-Bound Parameter Literals</a> for more information.

### Using String Literals

A string literal can be used in filter queries to match strings.

The following wildcards can be used:

- `?` matches any single character
- `*` matches any number of characters.

Wildcards can be escaped using the `\` character. For example, the literal `'my\?text'` will match the string `'my?text'`.

Escape characters can be escaped using `\\` characters. For example, the literal `'my\\text'` will match the string `'my\text'`.

### Using DateTime Literals

A DateTime literal in filter queries allows the creation of an arbitrary DateTime which will be matched against any DateTime format AttributeValue.

The following DateTime variations can be used:

```
YYYY[timezone]
YYYY-MM[timezone]
YYYY-MM-DD[timezone]
YYYY-MM-DDThh[timezone]
YYYY-MM-DDThh:mm[timezone]
YYYY-MM-DDThh:mm:ss[timezone]
YYYY-MM-DDThh:mm:ss.SSS[timezone]
```

#### Timezones

*timezone* is optional, and can be specified as shown in the following table.

Syntax	Description
Z	Coordinated Universal Time (UTC)
+ -HH[:MM]	Offset from UTC

For example:

```
2010Z
2010-01-01T01:00+01:00
2010-01-01T01:00.000-04:30
```

#### Ranges

When specifying a range, a curly bracket indicates an exclusive value; a square bracket indicates an inclusive value. Square and curly brackets can be mixed in the same range.

Range	Description
[2010, 2011]	Greater than or equal to 01/01/2010 and less than or equal to 31/12/2011
{2010, 2011}	Greater than 31/12/2010 and less than 01/01/2011
{2010, 2011]	Greater than 31/12/2010 and less than or equal to 31/12/2011

### Using Late-Bound Parameter Literals

A late-bound parameter literal (`::name` operand) can be used in filter queries to assign a late bound parameter to the expression.

The **name** parameter must be provided as part of the “parameter” array in the QueryOptions element, which is passed as part of an executeQueryRequest or executeRegisteredQueryRequest message.

Late-bound parameters are useful when repeatedly executing the same query many times with different values. Using [executeRegisteredQuery](#) with late-bound parameters means that the filter will be parsed once, then executed many times with the different values.

### Operators

This topic lists the supported filter query operators.

Symbol	Description	Restrictions on use
=	Equal To	Both operators must be of the same type.
≠	Not Equal To	
>=	Greater than or equal to	
<=	Less than or equal to	
<	Less than	
>	Greater than	
AND	AND	Both operators must be booleans.
OR	OR	
NOT	NOT	
IN	IN	



Where brackets are used, the contents must evaluate to a boolean; a bracketed operation will be used as a boolean from that point on in the expression.

## Using Attributes in Query Filters

Each Event Collector database table contains specific attributes for the type of information held in that table. For example, the **ec\_event** table contains attributes that hold information about events, the **ec\_case\_status** table contains attributes that hold information about case objects, and so on.

When you submit a query, the filter expression itself can contain one or more attributes from the table. Plus, you can request that all attributes, or only specific attributes, and their values, be returned in the response.

A **target** option is also passed in the query, which specifies the Event Collector database table or view against which the query will execute. The attributes that you can include in a query filter expression is determined by the **target** option specified in the query. See the list of target options below.

The following shows a basic query against the **ec\_pe\_status** table (**target=PROCESS\_STATS**). It is requesting that all attributes (**requireAllAttributes=true**) from the **ec\_pe\_stats** table be returned for process instances in which **caseReference='BDS-1-com.example.hastings.Customer-1-0'**.

**URL:**

```
.../audit/query/execute?target=PROCESS_STATS&requireallattributes=true
```

**Request Body**

```
{
 "executeQueryObject": {
 "query": { "filter": "caseReference='BDS-1-com.example.hastings.Customer-1-0'" }
 }
}
```

The following are the available **target** options. The links display lists of the attributes in each table (or view) that can be used in filters, or returned in the response, when targeting that table:

- [AUDIT](#) - for queries against the **ec\_event** Event Collector database view, which stores information about events. (This is the default option if **target** is not specified in the query.)
- [CASE\\_STATS](#) - for queries against the **ec\_case\_status** Event Collector database table, which stores information about case objects.
- [PROCESS\\_STATS](#) - for queries against the **ec\_pe\_status** Event Collector database table, which stores measures about process instances.
- [PROCESS\\_TEMPLATES](#) - for use only with the [getChart](#) and [getChartData](#) operations, as well as the [bpm-highcharts](#) business component. These attributes cannot be used when executing queries from the [AuditService](#).
- [REGISTERED\\_ATTRIBUTES](#) - for queries against the **ec\_attribute** Event Collector database table, which stores information about attributes that have been registered with Event Collector.
- [REGISTERED\\_COMPONENTS](#) - for queries against the **ec\_component** Event Collector database table, which stores information about components that have been registered with Event Collector.
- [REGISTERED\\_QUERIES](#) - for queries against the **ec\_query** Event Collector database table, which stores information about queries that have been registered with Event Collector.
- [USER\\_ACTIVITY](#) - for queries against the **ec\_user\_activity** Event Collector database table, which stores information about user activity.
- [WORKITEM\\_STATS](#) - for queries against the **ec\_wi\_status** Event Collector database table, which stores measures about work items.



Also see the *TIBCO ActiveMatrix BPM Event Collector Schema Reference* for details of the schema for all the Event Collector database tables. (Note that not all attributes in the Event Collector tables can be used in queries. One of the properties of an attribute, **isFilterable**, defines whether or not you can use that attribute in query filters. The attributes listed in this document are filterable.)

## AUDIT Option Attributes

A query that uses the AUDIT option operates against the **ec\_event** Event Collector database view, which stores information about events.

The following table describes the attributes that you can include in a filter expression in a query that uses the AUDIT option.



When using the AUDIT option, you can use the [getAttributes](#) function to list all the attributes registered against Event Collector in the ec\_event view. All attributes can be returned in a query. However, only attributes with the `isFilterable` attribute set to `true` can be used in a filter expression.

Attribute	Description/Notes
applicationActivityInstanceId	Unique identifier of the process activity associated with the event from the activity instance (runtime identifier).
applicationActivityModelId	Unique identifier of the process activity associated with the event from the model (design time identifier).
applicationActivityName	Name of the process activity associated with the event. For example:  gateway, StartEvent, ScriptTask, DBTask, gateway, Success, Failure.
applicationName	Name of the application that generated the event.
attribute1 - attribute40	The 40 user-defined custom work item attribute fields associated with the event. For a description of these attributes, see <a href="#">Using Work Item Attribute Fields</a> . Also see <a href="#">Sort Filter Criteria</a> for the data type of each attribute.
caseReference	Unique reference to a case object created by ActiveMatrix BPM.
caseSearchTag	Unique identifier of case data.
channelId	Unique identifier of the presentation channel associated with the event.
componentId	Identifier of the component that generated the event.
contextId	The contextID is used with the correlationID and parentContextID to determine the series of events that have occurred.  The contextID is set to a new value for each new service call made in a chain of events - thus identifying the contents of a specific service call.

Attribute	Description/Notes
correlationId	<p>CorrelationID of the event.</p> <p>The correlationID is used with the contextID and parentContextID to determine the series of events that have occurred.</p> <p>The correlationID is set when a service request is received from an external source. It remains the same for the duration of that request (including any child service calls made).</p>
creationTime	Timestamp that the event occurred.
entityId	Identifier of the organization model entity associated with the event.
entityType	<p>Type of the organization model entity associated with the event. This must be one of the following values:</p> <ul style="list-style-type: none"> <li>• ORGANIZATION</li> <li>• ORGANIZATIONAL_UNIT</li> <li>• GROUP</li> <li>• POSITION</li> <li>• PRIVILEGE</li> <li>• CAPABILITY</li> <li>• RESOURCE</li> <li>• LOCATION</li> <li>• ORGANIZATION_TYPE</li> <li>• ORGANIZATIONAL_UNIT_TYPE</li> <li>• POSITION_TYPE</li> <li>• LOCATION_TYPE</li> <li>• ORGUNIT_RELATIONSHIP_TYPE</li> <li>• POSITION_HELD</li> <li>• ORGUNIT_RELATIONSHIP</li> <li>• ORGUNIT_FEATURE</li> <li>• POSITION_FEATURE</li> <li>• PARAMETER_DESCRIPTOR</li> </ul>
environmentId	The reference to the ec_environment table.
extendedMessage	Extended message for the message. This will generally contain more information, for example, the reason for a failed activity.
hostAddress	Address of the host on which the event occurred.
hostName	Name of the host on which the event occurred.

Attribute	Description/Notes
id	Primary key of the event.
managedObjectId	Valid values for this attribute depend on the value of the <code>messageCategory</code> attribute - see <a href="#">Message Categories and Attribute Contents</a> .
managedObjectName	Valid values for this attribute depend on the value of the <code>messageCategory</code> attribute - see <a href="#">Message Categories and Attribute Contents</a> .
managedObjectType	Valid values for this attribute depend on the value of the <code>messageCategory</code> attribute - see <a href="#">Message Categories and Attribute Contents</a> .
managedObjectStatus	The status of the managed object. This is only applicable to the <code>WORK_ITEM</code> message category.
managedObjectUrl	Valid values for this attribute depend on the value of the <code>messageCategory</code> attribute. See <a href="#">Message Categories and Attribute Contents</a> .
managedObjectVersion	Valid values for this attribute depend on the value of the <code>messageCategory</code> attribute - see <a href="#">Message Categories and Attribute Contents</a> .
message	Description of the event.
messageCategory	Identifies the entity which was being operated on causing this event to be produced. See <a href="#">Message Categories and Attribute Contents</a> .
messageId	Identifier of the event. For example: <code>messageId= 'BX_INSTANCE_PROCESS*'</code>
modelMajorVersion	Major version number of a case data model.
modelVersion	Version number of a case model.
moduleName	Process package name of the deployed process associated with the event.
nodeName	Name of the node on which the event occurred.
parentActivityInstanceId	Identifier of the parent activity associated with the event.
parentContextId	<p>The <code>parentContextID</code> is used with the <code>correlationID</code> and <code>contextID</code> to determine the series of events that have occurred.</p> <p>The <code>parentContextID</code> is set to a new value each time an internal (to a component) service call is made, and is set to the <code>contextID</code> of the calling service, thus identifying the parent of this service call.</p>

Attribute	Description/Notes
parentObjectId	Valid values for this attribute depend on the value of the messageCategory attribute - see <a href="#">Message Categories and Attribute Contents</a> .
parentProcessInstanceId	Identifier of the parent process associated with the event.
principalId	Unique identifier of the security principal associated with the event.
principalName	Name of the security principal associated with the event.
priority	Priority of the event. This must be one of the following: <ul style="list-style-type: none"> <li>• LOW</li> <li>• MEDIUM</li> <li>• HIGH</li> </ul>
priorStepId	Identifier of the previous activity (to the one associated with the event).
processPriority	Priority of the process instance associated with the event.
resourceId	Identifier of the resource associated with the event.
resourceName	Name of the resource associated with the event.
roleName	The rolename given to a link between two case data classes.
severity	Severity of the event. This must be one of the following: <ul style="list-style-type: none"> <li>• TRACE</li> <li>• DEBUG</li> <li>• INFO</li> <li>• SERVICE</li> <li>• AUDIT</li> <li>• WARN</li> <li>• ERROR</li> <li>• FATAL</li> </ul>
subprocessInstanceId	Identifier of the sub-process associated with the event.
subprocessName	Name of the sub-process associated with the event.
subprocessVersion	Version of the sub-process associated with the event.
systemActionComponentId	Identifier of the component that requested a system action check. (Typically this will be BRM.)

Attribute	Description/Notes
systemActionId	Unique identifier of the requested system action.
retryTime	The time when the failed activity will be retried again.
workItemPriority	The priority of the work item.
workItemScheduleEnd	Scheduled end date of the work item associated with the event.
workItemScheduleStart	Scheduled start date of the work item associated with the event.

### CASE\_STATS Option Attributes

A query that uses the CASE\_STATS option operates against the **ec\_case\_status** Event Collector database table, which stores information about cases.

The following table describes the attributes that you can include in a filter expression in a query that uses the CASE\_STATS option.

Attribute	Description/Notes
caseClass	A case class is a template for a case object.
caseDuration	Total duration (in milliseconds) for the case object.
caseId	Unique identifier of the case class that is specified at design time.
caseRef	Unique reference (or pointer) to a case object, created by ActiveMatrix BPM when the case object is created.
caseState	A case state is a special type of attribute, available only on case classes, that can be used to uniquely identify a set of business-specific states that a case can be in.
createdTime	Date/time that the case object was created.
deletedTime	Date/time that the case object was deleted; this is null if the case is not yet deleted.  To return all active cases, use: <code>NOT(isSet(deletedTime))</code>
lastModified	Timestamp that the case object was last modified
modelMajorVersion	The major version number of the case model.
modelVersion	The version number of the case model.
userGuid	GUID of the resource associated with this case object.
userId	Identifier of the organization model entity associated with the case object.
version	The version number of the case object.



## PROCESS\_STATS Option Attributes

A query that uses the PROCESS\_STATS option operates against the ec\_pe\_status Event Collector database table, which stores measures about process instances.

The following table describes the attributes that you can include in a filter expression in a query that uses the PROCESS\_STATS option.

Attribute	Description/Notes
attribute1 - attribute40	The 40 user-defined custom work item attribute fields associated with the event. See <a href="#">Using Work Item Attribute Fields</a> for a description of these attributes, and see <a href="#">Sort Filter Criteria</a> for the data type of each attribute.
caseReference	Unique reference to a case object created by ActiveMatrix BPM.
currentActivity	The current activity that is in progress for this process instance.
currentActivityStart	The time the current activity started.
endTime	Completion time of this process instance.
moduleName	Module name related to the parent process template.
parentProcessInstanceId	Identifier of the parent process associated with the event.
pvmId	The numeric ID of the process instance. This ID can be used to join against the process engine tables. Also see the ref attribute below.
priority	Priority of this process instance.
processTemplateId	Unique identifier of the process template. It includes the process template name, module name, and module version. This is stored in the ec_proc_template table.
processTemplateName	Name of the parent process template.
ref	The string representation of the identifier for the process instance (beginning with "pvm:"). Also see the pvmId attribute above
rootProcessInstanceId	The root of the process instance hierarchy. This points to an instance that has the parent as null.
startTime	Start time of this process instance.
status	Status of this process instance.
statusChanged	Timestamp of the last status change of this process instance.
timeTaken	Total time taken (in milliseconds) by this process instance between its startTime and endTime.

Attribute	Description/Notes
type	Identifies the type of process instance, for example, a sync-attached process instance.
userGuid	Guid of the resource associated with this process instance.
userId	Resource associated with this process instance.
version	Version number of the parent process template.

### PROCESS\_TEMPLATES Attributes

The PROCESS\_TEMPLATES attributes are only applicable when using the [getChart](#) and [getChartData](#) operations, as well as the [bpm-highcharts](#) business component. These attributes cannot be used when executing queries from the [AuditService](#).

The PROCESS\_TEMPLATES attributes contain process template information stored in the **ec\_proc\_template** database table.

The following table lists the PROCESS\_TEMPLATES attributes.

Attribute	Description/Notes
appName	The name of the deployed application.
moduleName	Module name that the process template belongs to.
moduleVersion	Version of the module.
pvmId	The numeric ID of the process instance. This ID can be used to join against the process engine tables.
pvmRef	The process engine ID of the process template.
processTemplate Name	Process template name.

### REGISTERED\_ATTRIBUTES Option Attributes

A query that uses the REGISTERED\_ATTRIBUTES option operates against the ec\_attribute Event Collector database table, which stores information about attributes that have been registered with Event Collector.

The following table describes the attributes that you can include in a filter expression in a query that uses the REGISTERED\_ATTRIBUTES option.

Attribute	Description/Notes
category	Category of this attribute.
componentId	Unique identifier of the attribute's parent component.

Attribute	Description/Notes
id	Primary key of this attribute.
ref	Name of this attribute.
type	Type of this attribute. This must be one of the following: <ul style="list-style-type: none"> <li>• INT</li> <li>• LONG</li> <li>• DOUBLE</li> <li>• BOOLEAN</li> <li>• STRING</li> <li>• DATE</li> </ul>

### REGISTERED\_COMPONENTS Option Attributes

A query that uses the REGISTERED\_COMPONENTS option operates against the **ec\_component** Event Collector database table, which stores information about components that have been registered with Event Collector.

The following table describes the attributes that you can include in a filter expression in a query that uses the REGISTERED\_COMPONENTS option.

Attribute	Description/Notes
componentClass	Component Logging Meta Data (CLMD) class file used by this component.
componentName	SCA component name - for example, implementation.brm. This defaults to the ref value if it is an internal (non-SCA) component.
componentType	SCA component type - for example, TIBCO-IT-SPRING or TIBCO-IT-WEBAPP.
description	Description of this component.
id	Primary key of this component.
parentId	Identifier of the parent component.
ref	Unique identifier of this component. For example, BRM or DE.
revision	Deployed plugin revision number of this component.
version	Deployed plugin version number of this component.

### REGISTERED\_QUERIES Option Attributes

A query that uses the REGISTERED\_QUERIES option operates against the **ec\_query** Event Collector database table, which stores information about queries that have been registered with Event Collector.

The following table describes the attributes that you can include in a filter expression in a query that uses the REGISTERED\_QUERIES option.

Attribute	Description/Notes
attributes	Required attributes to be fetched when executing this registered query.
correlate	Boolean value defining whether correlation is required for this registered query.
filter	Filter string used by this registered query.
id	Primary key of this registered query.
ref	Unique tag defined for this registered query.
sort	Attributes defining the sort criteria for this registered query.
target	Target of this registered query, defining the database table against which the query executes - for example, AUDIT or REGISTERED_ATTRIBUTES.

### USER\_ACTIVITY Option Attributes

A query that uses the USER\_ACTIVITY option operates against the **ec\_user\_activity** Event Collector database table, which stores information about user activity.

The following table describes the attributes that you can include in a filter expression in a query that uses the USER\_ACTIVITY option.

Attribute	Description/Notes
actionDuration	Total duration (in milliseconds) for this user activity.
actionEnd	End date/time for this user activity.
actionStart	Start date/time for this user activity.
activityInstanceId	Instance identifier of the activity from which the work item was derived. This value is unique, even if the activity is executed multiple times, for example, as part of a loop.
activityName	Name of the activity from which the work item was derived.
id	Unique identifier for this user activity.
processInstance	Identifier of the parent process instance from which this work item was generated.
processTemplate	Identifier (name) of the process template from which the parent process instance was generated.
processTemplateId	The reference to the process template, module name, and module version referenced in the ec_proc_template table.
ref	The work item associated with this user activity.
userAction	The user action for this work item.

Attribute	Description/Notes
userGuid	GUID of the resource associated with this work item.
userId	Resource associated with this work item.
wiStatus	Status of the work item.

### WORKITEM\_STATS Option Attributes

A query that uses the WORKITEM\_STATS option operates against the ec\_wi\_status Event Collector database table, which stores measures about work items.

The following table describes the attributes that you can include in a filter expression in a query that uses the WORKITEM\_STATS option.

Attribute	Description/Notes
actionDuration	Total duration (in milliseconds) that this work item was being actioned - that is, the time between its firstOpenTime and its completionTime.
activeDuration	Total duration (in milliseconds) that this work item was active - that is, the time between its firstOfferTime and its completionTime, disregarding any intermediate states.
activityInstanceId	Instance identifier of the activity from which the work item was derived. This value is unique even if the activity is executed multiple times - for example, as part of a loop.
activityName	Name of the activity from which the work item was derived.
attribute1 - attribute40	The 40 user-defined custom work item attribute fields associated with the work item. For a description of these attributes, see <a href="#">Using Work Item Attribute Fields</a> . Also see <a href="#">Sort Filter Criteria</a> for the data type of each attribute.
caseReference	Unique reference to a case object created by ActiveMatrix BPM.
completionTime	Completion time of this work item.
completingUserId	Resource completing this work item.
completingUserGuid	GUID of the resource completing this work item.
firstOfferTime	First time that this work item was offered or allocated.
firstOpenTime	First time that this work item was opened.
lastOpenTime	Last time that this work item was opened.
processInstance	Identifier of the parent process instance from which this work item was generated.

Attribute	Description/Notes
processTemplate	Identifier (name) of the process template from which the parent process instance was generated.
processTemplateId	The reference to the process template, module name, and module version, referenced in the ec_proc_template table.
pvmId	The numeric ID of the process instance. This ID can be used to join against the process engine tables.
pvmTaskId	The numeric version of the "pvm" ID that relates to the activity within the process.
ref	Unique identifier of this work item.
rootProcessInstanceId	The root of the process instance hierarchy. This points to an instance that has the parent as null.
scheduleEnd	Scheduled end date/time (in UTC) for this work item.
scheduleStart	Scheduled start date/time (in UTC) for this work item.
statusChanged	Time that the status of this work item was last changed.
userGuid	Guid of the resource associated with this work item.
userId	Resource associated with this work item.
waitDuration	Total duration (in milliseconds) that this work item was waiting - that is, the time between its firstOfferTime and its firstOpenTime.
wiStatus	Status of this work item.
workingTimeDuration	Total working time duration (in milliseconds) for this work item i.e. the cumulative time for between first and last (form submission) activities.

### Message Categories and Attribute Contents

When constructing a filter expression for a query against the **ec\_event** view (where the query uses the AUDIT option), there are several attributes for which the possible valid values change depending on the value of the **messageCategory** attribute.

For example:

- If **messageCategory**='PROCESS\_INSTANCE', **managedObjectName** must be the name of a process template.
- If **messageCategory**='ORGANIZATIONAL\_ENTITY', **managedObjectName** must be the name of an organization model entity.

For a list of these categories and attributes, see the "Auditable Messages" topic in the *TIBCO ActiveMatrix BPM Administration Guide*.

## Example Queries

The following are example filter queries against the **ec\_event** Event Collector database view (target=AUDIT).

Description	Query
Returns all events referring to process instance pvm:0a121.	<pre>"query": { "filter": "messageCategory='PROCESS_INSTANCE' AND managedObjectId='pvm:0a121'"}</pre>
Returns all events referring to a process instance value that is defined by the ProcID late bound parameter literal.  The <b>executeGenericQueryRequest</b> or <b>executeRegisteredGenericQueryRequest</b> message must include a value for ProcID as part of the QueryOptions element.	<pre>"query": { "filter": "messageCategory='PROCESS_INSTANCE' AND managedObjectId=:ProcID"}</pre>
Returns all events for work items generated by process instance pvm:0a126.	<pre>"query": { "filter": "messageCategory='WORK_ITEM' AND parentObjectId='pvm:0a126'"}</pre>
Returns all events referring to process instance pvm:0a121 where the messageID is not BX_INSTANCE_PROCESS_COMPLETED.	<pre>"query": { "filter": "messageCategory='PROCESS_INSTANCE' AND managedObjectId='pvm:0a121' AND messageId &lt;&gt; 'BX_INSTANCE_PROCESS_COMPLETED'"}</pre>
Returns all events for work items associated with the resource (user) Clint Hill.	<pre>"query": { "filter": "messageCategory='WORK_ITEM' AND resourceName='Clint Hill'"}</pre>
Returns all events where the message attribute value begins with 'Start', followed by any number of characters.	<pre>"query": { "filter": "message = 'Start*'"}</pre>
Returns all events for the current user.	<pre>"query": { "filter": "currentUser(resourceName)"}</pre>
Returns all events referring to process instance pvm:0a121 created between 9 AM on July 10, 2017 and 5 PM July 14, 2017, inclusive. Times are local.	<pre>"query": { "filter": "messageCategory='PROCESS_INSTANCE' AND managedObjectId = 'pvm:0a121' AND creationTime=[2017-07-10T09,2017-07-14T17]"}</pre>
Returns all events referring to process instance pvm:0a121 created between 8 AM July 10, 2017 and 6 PM July 14, 2017, exclusive. Times are Eastern Daylight Time (-04 hour offset from UTC).	<pre>"query": { "filter": "messageCategory='PROCESS_INSTANCE' AND managedObjectId = 'pvm:0a121' AND creationTime={2017-07-10T08-04,2017-07-14T18-04}"}</pre>

Description	Query
Returns all events referring to process instance pvm:0a121 created between 8 AM July 10, 2017 (exclusive) and 5 PM July 14, 2017 (inclusive). Times are Indian Standard Time (+5 1/2 hour offset from UTC).	<pre>"query": { "filter": "messageCategory='PROCESS_INSTANCE' AND managedObjectId = 'pvm:0a121' AND creationTime={2017-07-10T08+05:30,2017-07-14T17 +05:30]"}</pre>

For information about dates and times being inclusive or exclusive, based on bracket type, see [Using DateTime Literals](#).

## Event Measures

The `StatisticsService` API provides a function for accessing statistical data for both processes and work items.

The [getMeasure](#) function is used to request measures for processes or work items from the Event Collector database tables.

## Purging Audit Data

The audit entries for a process instance are not required once the process instance completes, is cancelled, or has failed.

The [AuditService](#) API provides functions to manage purges of audit data from the Event Collector database tables—that is, to delete the audit entries and statistical data for completed, cancelled, and failed process instances.

The [AuditService](#) provides the following functions for purging:

- [doPurge](#) - This function purges data from the Event Collector database tables. Its options enable you to define the scope of the purge.
- [stopPurge](#) - This function stops an ongoing purge.
- [checkPurge](#) - This function checks the status of an ongoing purge.



- The functionality provided by this service is equivalent to that of the administrator-initiated stored procedures described in the section "Clearing Audit Entries and Statistical Data" in the *TIBCO ActiveMatrix BPM - Administration* guide.

The performance of the data purges described in this section will be greatly improved if you create the same database indexes that you are recommended to use with these stored procedures. See the section "Clearing Audit Entries and Statistical Data" in the *TIBCO ActiveMatrix BPM - BPM Administration* guide for the syntax to use for each of the supported database types.

- TIBCO recommends that you back up your database, according to whatever backup strategy your organization has implemented, before running [doPurge](#) or initiating the equivalent stored procedure. Refer to the documentation supplied with your database server for information on how to do this.

## Calendars

Calendars are used to hold information about working times, which can be used in scheduling work and calculating deadlines.

You can have multiple calendars. This allows you to handle working times and deadlines for locations in different time zones, for groups with different working hours, and so on.

Two services are provided to manage calendars:



- [WorkCalService](#) - This service includes functions to create, delete, edit, and get information about base and overlay calendars and the time zones and calendar entries that they use.
- [BusinessDeadlineService](#) - This service contains a function used to calculate the deadline for completing a specified task, based on a given calendar.

## Base and Overlay Calendars

There are two kinds of calendars: base and overlay.

These calendar types work together, as follows:

### Base calendars

A base calendar defines the normal working hours. It defines which hours in a standard seven-day week are available as working hours, repeating every week throughout the year, but does not map these working hours to specific calendar dates.

Base calendars can also define exceptions to the normal defined working hours. These *working-day exclusions* do apply to specific dates. For example, a base calendar might define working hours as 08:00 to 17:00 on Mondays to Fridays; but public holidays are not working days, so the calendar might contain an exclusion for 01 January. Exclusions have a defined duration—an exclusion for a public holiday would last all day, while one for a company meeting might last a couple of hours—and may be defined as recurring after a specified interval, such as every week or every month.

A default base calendar, with the name SYSTEM, is provided. This calendar is used when no other calendars apply.

A base calendar specifies the time zone to which it applies.



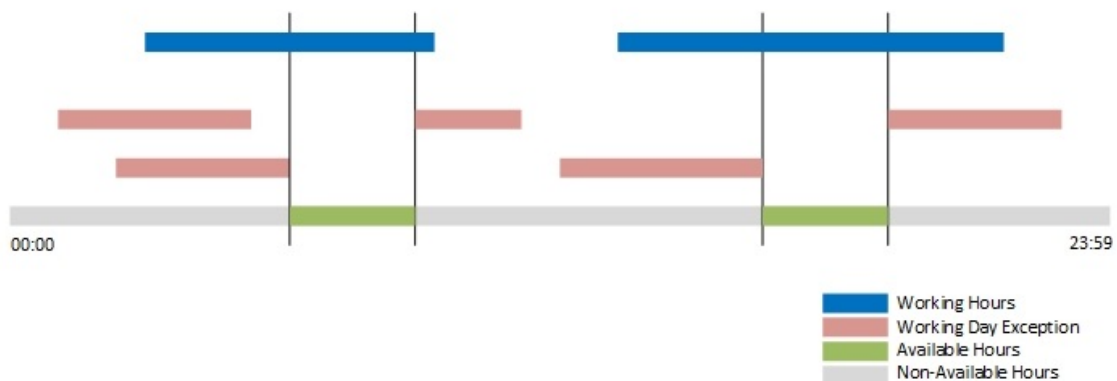
If no time zone is specified on a base calendar, it uses the time zone of the BPM server. However, if for example the end user is in a different time zone from the server, this can lead to incorrect times being generated. It is therefore best practice always to specify a time zone for a base calendar.

### Overlay calendars

An overlay calendar is applied at runtime on top of the base calendar. An overlay calendar contains only working-day exclusions. These are the same as the exclusions defined in base calendars, in that they apply to specific dates, have a defined duration, and may be defined as recurring.

Overlay calendars do not have a time zone defined, but take the time zone of the base calendar on top of which they are applied.

The following illustration shows the effect of combining base and overlay calendars. The working hours defined in the base calendar, and the working day exclusions defined (in this example) in both the base and the overlay calendars, are combined to determine the available working hours.



## Exclusions

Calendar entries may be one-off working-day exclusions, defining one specific instance that diverges from the normal working days laid down in the base calendar, or may be recurring entries.

Recurring entries repeat at intervals, either indefinitely or for a prescribed time.

This recurrence is governed by a rule that follows the industry standard definition given in [RFC-5545](#).

## Time Zones

Base calendars each have a time zone specified. This enables you to use different base calendars in different locations around the world. Overlay calendars do not specify a time zone, so that an overlay can be applied on to any base calendar without any conversion.

Time zones are specified using the **time-zone** parameter in the [saveBaseCalendar](#) function

All date-time entries are stored, whether on a base or an overlay calendar, with a time zone of UTC attached. This does not mean that they are necessarily intended to represent UTC times; in effect times are treated as time zone-neutral. For example, working hours entered as 08:00 to 17:00 on a base calendar with its time zone set to PST (UTC minus 8 hours) are actually stored as 08:00 to 17:00 UTC, not as 16:00 to 01:00 UTC. In most cases this is completely transparent to the user, but you need to be aware of how the system handles time zones and date-time values if you are making calls directly to the API.

Changing a base calendar's **time-zone** value does not cause any adjustments to existing or future calendar entries. So if you define the start of the working week as 08:00 on Monday, it is still 08:00 on Monday even if the time zone of the calendar changes.

Although the date-time values are time zone-neutral, it is best practice to specify a time zone of z (UTC) for values such as the **time-slot start** and **end** times that define working hours entries in a base calendar. If no time zone is specified, some functions will automatically convert the value to the time zone used by the server, regardless of the calendar's **time-zone** specification. This may not always be the time zone required. For example, if a user in Paris accesses a server in California and enters a calendar time of 11:00, expecting it to be 11:00 CET (10:00 UTC), the time would be interpreted as 11:00 PST and stored as the UTC equivalent of that time, 19:00 UTC. Entering 11:00Z, however, is stored as 11:00 UTC.

Times should be treated in this way whether they are for base or for overlay calendars. This enables the entries to be applied as if they were time zone-neutral, so that any overlay calendar can be applied on top of any base calendar.



When calculating deadlines, start times must be specified (using [calcDeadline](#)) as actual UTC times. For information on how time zones are treated when calculating deadlines, see [Scheduling](#).

## Calendar References

Business processes and timer events can have *calendar references* that link them to appropriate calendars. This allows you to reflect situations where different parts of the organization have different working patterns.

Calendar references (also called *calendar aliases*) are created in TIBCO Business Studio, then assigned to a calendar using the API. For information about assigning a calendar reference to a calendar, see [Assigning a Calendar Reference to a Calendar](#).

A calendar reference consists of:

- a namespace, a name, and a type (Base or Overlay). These are used by TIBCO Business Studio user to identify the calendar required, but do not necessarily match the namespace and name of any actual calendar,

- a GUID generated automatically by TIBCO Business Studio, that provides a unique identifier for this one reference.

The [WorkCalService](#) contains the following calendar reference-related functions:

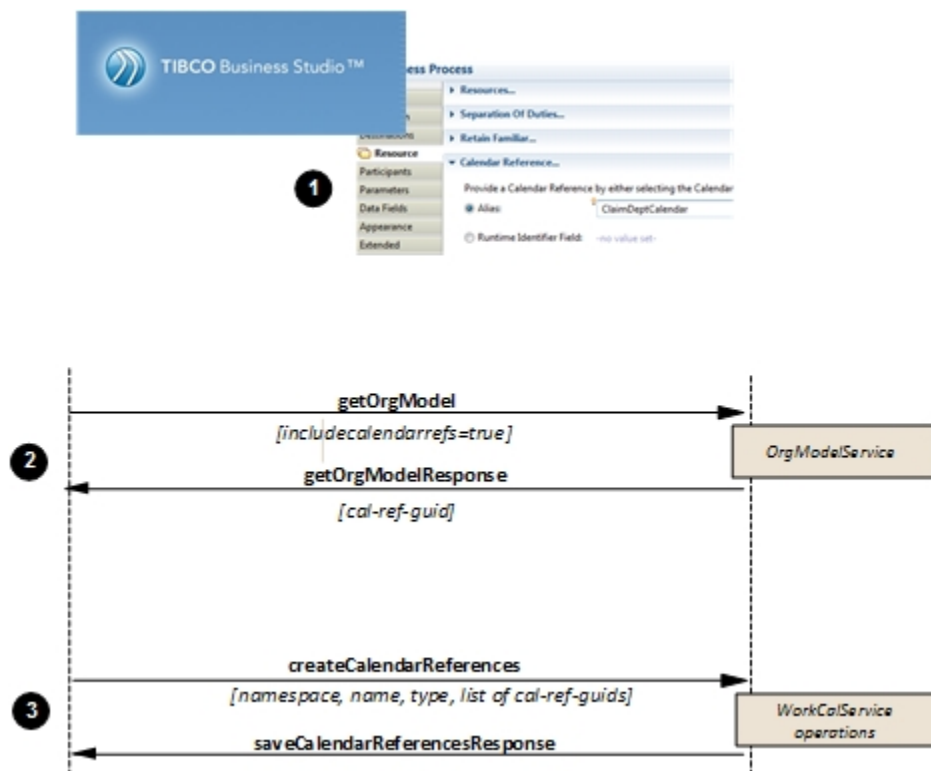
- [createCalendarReferences](#) - Associates a calendar reference to a calendar.
- [deleteCalendarReferences](#) - Deletes the specified calendar references from the identified calendar.
- [resolveReferences](#) - Resolves a collection of specified calendar reference GUIDs, that is, it tells you the calendars to which the specified calendar references are associated.

For more information about calendar references, see the "Intermediate Events" section in the *TIBCO Business Studio Process Modeling User's Guide*.

## Assigning a Calendar Reference to a Calendar

A *calendar reference* is created in TIBCO Business Studio and is then assigned to a calendar using the API.

### Creating and Assigning a Calendar Reference



### Procedure

1. In TIBCO Business Studio, on the **Resource** tab of the **Properties** view for the process or the event, enter one of the following:
  - a reference in free text, that identifies the calendar required,
  - or a process field which will provide that reference at runtime.

For more information on calendar references in Business Studio, see the "Intermediate Events" section in the *TIBCO Business Studio Process Modeling User's Guide*.

When you deploy the process, a GUID is generated automatically. This is referred to as the **cal-ref-guid**, and it uniquely identifies the reference.

2. Call `getOrgModel` with `includeCalendarRefs` set to `true` to retrieve the calendar references. .
3. Map a calendar reference, or multiple references, to an actual calendar using `createCalendarReferences`.

## Result

Subsequently, you can use `resolveReferences` to check what references are mapped to which calendars.

## Calendars and User Tasks

In TIBCO Business Studio, processes and timer events on the boundary of user tasks can be associated with references to base calendars. That base calendar is then used to calculate deadlines for any user tasks in the process that do not have participants (and so cannot derive a calendar from any organizational entity).

See the "Intermediate Events" section in the *TIBCO Business Studio Process Modeling User's Guide* for details of how to do assign a calendar reference to a timer event.

The calendars to use are determined as follows:

1. If a user-task timer event has a base calendar reference, that reference is considered first.
2. If the user-task timer event does not have a base calendar reference, the user task's participant is considered. If the user-task participant references a single organizational entity, that entity is used to look up a calendar as described in [Time Zones](#).



If you use an RQL query to define a participant, there is no guarantee that the query will resolve to a single organizational entity. Therefore, RQL participants cannot be used to derive a calendar.

3. If neither the user-task timer event nor the participant provides a base calendar reference, the process is considered.
4. If none of these produce a reference to a base calendar, the default System calendar is used as the base.
5. The same sequence is used to find an overlay calendar to apply to whichever base calendar has been selected.
6. The resulting base calendar, and overlay calendar if any, are used to calculate deadlines.



An overlay calendar must be applied to a base calendar—it cannot be used on its own. If the overlay cannot be applied to any other base calendar, the default System calendar is used, as set out in the procedure above. This may not always give the desired results, particularly if the server holding the System calendar is in a different country or a different time zone from where the process will be carried out. When specifying calendar references in TIBCO Business Studio, you should therefore be very cautious when specifying references that resolve to overlay calendars.

## Creating Calendars

To set up a system of calendars for your business, you need to plan which calendars are needed to meet your business requirements.

First, since a Base calendar is associated with a time zone, you will want at least one Base calendar for each time zone in which your business operates. If you have multiple offices or other business units in the same time zone but with different working weeks—perhaps in different countries, or perhaps offices that work five-day weeks and transport depots that work six or seven days—you will want more than one Base calendar with the same time zone, but different **working-days** properties.

For example, in an environment that provides 24-hour support, you might have:

- An **Early** Base calendar, defining working hours as from 06:00 to 14:00 per day, which would be referenced by the organization unit **EarlyShift**,
- A **Late** Base calendar, defining working hours as from 14:00 to 22:00, which would be referenced by the organization unit **LateShift**,
- A **Night** Base calendar, defining working hours as from 22:00 to 06:00, which would be referenced by the organization unit **NightShift**.

Alternatively, the same functionality could be provided by defining a 24-hour working day on the Base calendar and using three overlay calendars to modify it.

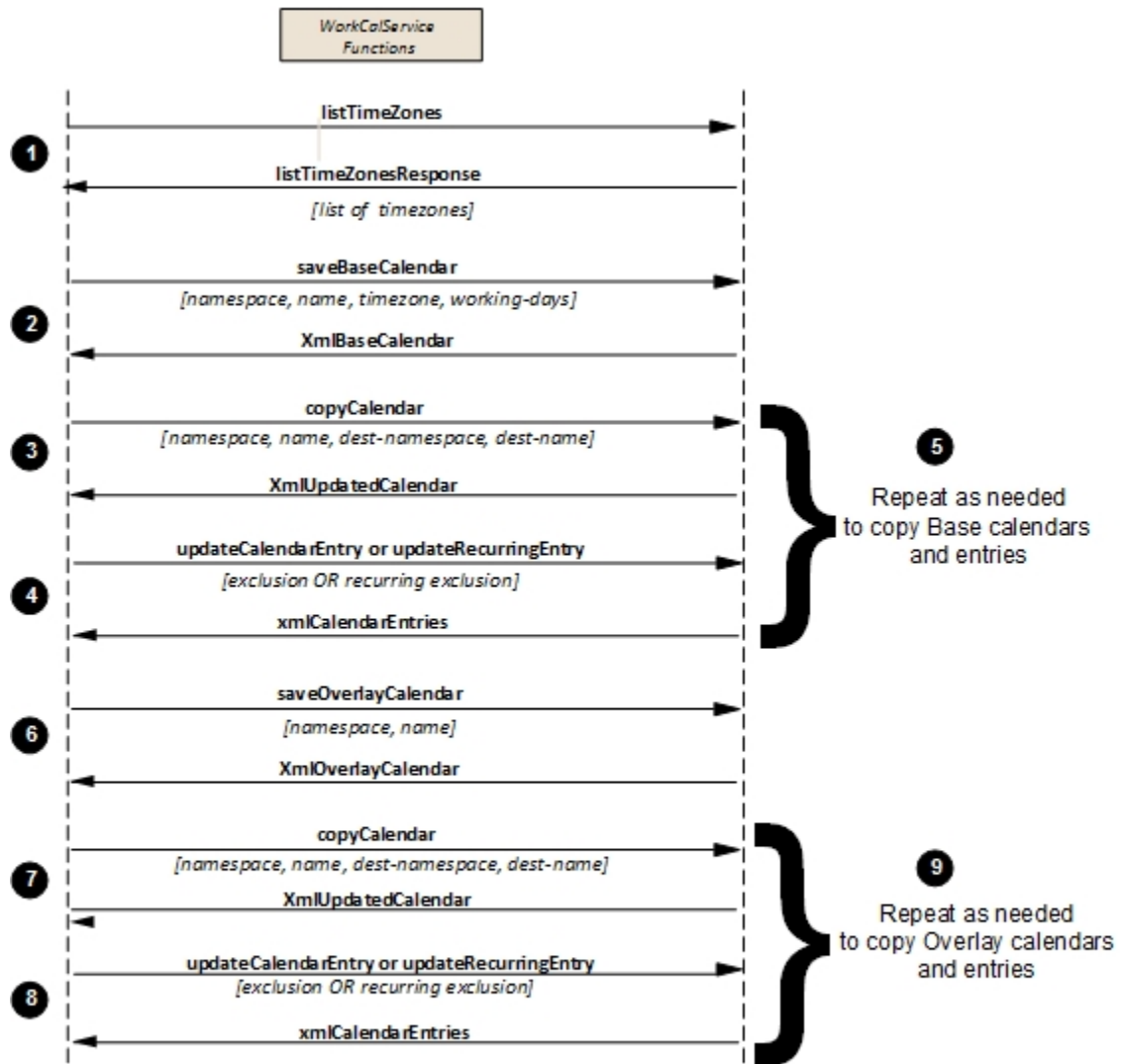
You might also use Overlay calendars:

- On a national level, to define public holidays, if you choose not to do so in the Base calendar,
- To restrict working hours for shifts. For example, a claims assessment department in an insurance company works from 08:00 to 20:00. But no individual works a 12-hour day; the department is divided into two shifts that work 08:00 to 16:00 and 12:00 to 20:00. Therefore the department uses:
  - a Base calendar that defines the whole working day, that is referenced by the claims assessment department Organization Unit,
  - two Overlay calendars, that each exclude the non-working hours for one of the shifts, and each of which is referenced by the appropriate shift's Organization Unit.

While organizational entities such as organization units or groups will use these Overlay calendars, your system of calendars need not correspond directly to the organization model. The same Overlay calendar might be referenced by several different organizational entities.

Having planned your calendars, you can follow a procedure similar to that shown in the figure below.

## Creating Base and Overlay Calendars



### Procedure

1. Use [listTimeZones](#) to generate a list of the permitted time zone values to be assigned to Base calendars.
2. Create an initial Base calendar using [saveBaseCalendar](#). You can set the time zone of the new calendar, using one of the values returned by the [listTimeZones](#) response. Use the **WorkingDays** parameter to specify the normal working hours for each day of the working week.
3. You can use [copyCalendar](#) to copy your initial Base calendar to create a second.



Note that [copyCalendar](#) does not allow you to change the time zone of the new calendar. If your strategy is to have a Base calendar for each time zone in which your business operates, for example, you could subsequently use [saveBaseCalendar](#) to change the time zone of each copy.

4. Use [updateCalendarEntry](#) or [updateRecurringEntry](#) to create the entries on your second calendar.
5. Repeat [Step 3](#) to [Step 4](#) as necessary until you have all the Base calendars you want.

6. Use [saveOverlayCalendar](#) to create an Overlay calendar.
7. Create another Overlay calendar using [copyCalendar](#) to copy the first one.
8. Create the entries on your second Overlay calendar using [updateCalendarEntry](#) or [updateRecurringEntry](#).
9. Repeat [Step 7](#) to [Step 8](#) as necessary until you have all the Overlay calendars you want.

## Identifying Calendars

A calendar is identified by a *namespace* and a *name*.

These are defined as follows:

- **namespace** - a convenient grouping for calendars. A namespace is not compulsory; for example, the default Base calendar SYSTEM does not have a namespace.
- **name** - a name for the individual calendar. It must be unique within the namespace, but not necessarily across the whole system.

A calendar also has a **version** number. The calendar is created at version 0 and the version number is incremented by one each time a change is made. However the version does not form part of the calendar's identity, in the sense that it does not have to be passed when retrieving a calendar using the [getCalendar](#) function

The version number is used to implement optimistic locking (see <http://c2.com/cgi/wiki?OptimisticLocking> for details) when calling save and update operations, in order to prevent multiple clients from changing the same calendar.

## Adding Recurring Exclusions

Sometimes regular events, such as regular seminars or training days, cause predictable exclusions from normally programmed working hours.

You can enter such recurring exclusions using [updateRecurringEntry](#). A recurring exclusion is defined by:

- **start** and **end** datetimes - **start** defines the date and time of the first occurrence. The **end** datetime records the end of the last event occurrence, and is optional.
- **duration** - defines the length of each occurrence. It corresponds to the standard notation given in [RFC-5545](#).
- **recurrence rule** - the rule that sets out how the start datetime of each occurrence of the exclusion is calculated. This is given in the standard notation set out in [RFC-5545](#), and is defined in [Recurrence Rule](#).

There is also an optional **description**.

## Recurrence Rule

The recurrence rule consists of a collection of rule parts, which define how to calculate an occurrence. The recurrence is defined in RFC-5545.

- **Frequency** defines the type of recurrence rule. Valid values are:
  - **SECONDLY** for events that repeat at an interval of a second or more.
  - **MINUTELY** for events that repeat at an interval of a minute or more.
  - **HOURLY** for events that repeat at an interval of an hour or more.
  - **DAILY** for events that repeat at an interval of a day or more.



- **WEEKLY** for events that repeat at an interval of a week or more.
- **MONTHLY** for events that repeat at an interval of a month or more.
- **YEARLY** for events that repeat at an interval of a year or more.
- **Interval** a positive integer (defaulting to 1) that defines the intervals at which the events occur. For example, a **Frequency** of **DAILY** and an **Interval** of 8 means "every eight days".
- **Week Start** specifies the day on which the working week starts. This is significant when a **WEEKLY** rule type has an **Interval** greater than 1, and a **BYDAY** rule part is specified. This is also significant when a **YEARLY** rule type has a **BYWEEKNO** rule part specified. The default value is **MO** [Monday].
- **Count** defines the number of occurrences after which the recurrence rule finishes. A rule can have either a **Count** or **Until**, but not both. If neither **Count** nor **Until** are specified, the recurrence rule is infinite.
- **Until** defines the start datetime of the last occurrence of the rule. A rule can have either a **Count** or **Until**, but not both. If neither **Count** nor **Until** are specified, the recurrence rule is infinite.
- **BY??? rules** such as **BYDAY=MO**. A collection of rule parts that describe how a date should be adjusted in order to calculate the start of each event. Each type of **BY??? rule** describes how one element of a date or time is adjusted. Possible rules are **BYSECOND**, **BYMINUTE**, **BYHOUR**, **BYDAY**, **BYMONTHDAY**, **BYEARDAY**, **BYWEEKNO**, **BYMONTH** and **BYSETPOS**.

These rule parts are applied in a fixed order, and recursively. As the rule parts are applied, the date output by one rule part becomes the input to the next rule part. When all rule parts have been applied, the initial input date is incremented, according to the **frequency** and **interval**, and the process repeated. This continues until either the number of calculated events equals the **Count**, or until the next event would start after the **Until** date.

The recurrence rule is persisted as an expression, that is, as a string using the grammar defined in [RFC-5545](#). Some examples are:

- `FREQ=WEEKLY ; INTERVAL=2 ; UNTIL=19971224T000000Z ; WKST=SU ; BYDAY=MO , WE , FR`

The exclusion occurs Monday, Wednesday, and Friday of every fortnight, until 24 December 1997. Weeks start on Sunday.

This rule is broken down, and interpreted as follows:

- `FREQ=WEEKLY` — The rule applies on a weekly basis.
- `INTERVAL=2` — The frequency interval is 2, making it, every 2 weeks.
- `UNTIL=19971224T000000Z` — The rule ends on 24 December 1997.
- `WKST=SU` — The rule states that Sunday is the first day of a week.
- `BYDAY=MO , WE , FR` — The rule applies on first Monday, Wednesday and Friday of the weeks identified by the **frequency** and **interval**.
- `FREQ=WEEKLY ; INTERVAL=2 ; COUNT=8 ; WKST=SU ; BYDAY=TU , TH`  
Tuesday and Thursday, of every fortnight, for the next eight fortnights. Week starts on Sunday.
- `FREQ=YEARLY ; INTERVAL=4 ; BYMONTH=11 ; BYDAY=TU ; BYMONTHDAY=2 , 3 , 4 , 5 , 6 , 7 , 8`  
The event applies on the first Tuesday, between 2 November and 8 November, of every fourth year, indefinitely.
- `FREQ=MONTHLY ; INTERVAL=2 ; COUNT=10 ; BYDAY=1SU , -1SU`  
The first and last Sunday, of every other month, for the next 20 (2 x 10) months.
- `FREQ=WEEKLY ; INTERVAL=1 ; COUNT=8 ; BYDAY=FR`  
Every Friday, for the next eight weeks.



## Scheduling

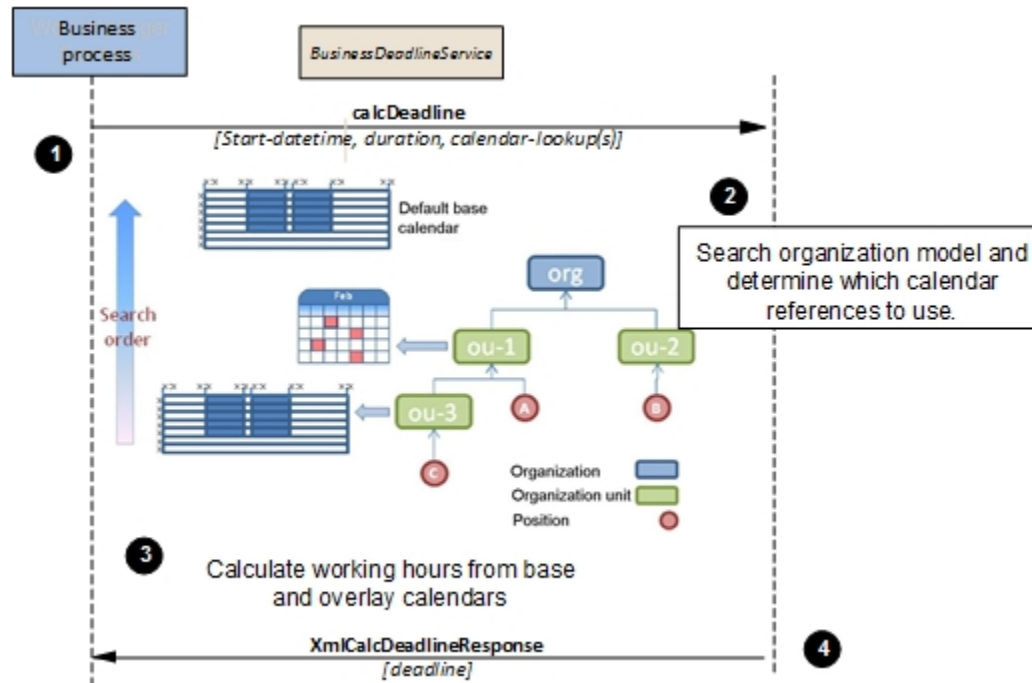
Once calendars are defined, calendar references are assigned and are mapped to calendars, the Deadline and Calendar (DAC) component can work out the deadline for a work item.

This is done using the [calcDeadline](#) function. The process followed is like that shown in the figure below.



Deadlines of less than one hour are calculated without using calendars or invoking the Calendar services components. In these situations, the work item's duration is simply added to its start time.

### Calculating a Deadline



### Procedure

1. Call the [calcDeadline](#) function.

This function specifies when the work item starts, how long it should take, and one or more items of calendar lookup information — either calendar references or identifiers of organizational entities involved in the work item.

2. Perform an examination of the organization model to determine which base calendar reference and which (if any) overlay calendar reference to use.

For more information, see [Identifying Calendars](#)

3. Once the calendars have been determined, the available working hours are calculated from the information in the calendar entries.
4. The combination of the starting time, the duration of the work item, and the working hours available is used to calculate the earliest date and time at which the given work item can be completed.

- The **duration** parameter may be interpreted as indicating elapsed time or a number of working days, depending on how it is expressed. For details, see [Duration Definitions](#).
- If the **duration** parameter is defined in working days, it ignores any day that has a number of available hours less than the **min-hours** parameter defined for the base calendar being used. For more information, see [Minimum Hours in Working Day Calculations](#).

The value set for the `calendarLookAhead` property in the `dac.properties` file determines how far ahead the calculator should look when working out the deadline. If the calculation shows that there is not enough working time available in the period defined by `duration`, plus `calendarLookAhead`, an error is returned.

The deadline is calculated using UTC, but displayed using the local time zone of the client. For example, assume that a work item is offered to the users in an organization unit, and that the base calendar for that organization unit is based upon the London time zone. However, the work item being offered resides on a BPM node based in New York. Calculation of the deadline is based on the time zone and the working times of the London organization unit, and not on the New York time zone of the BPM node.

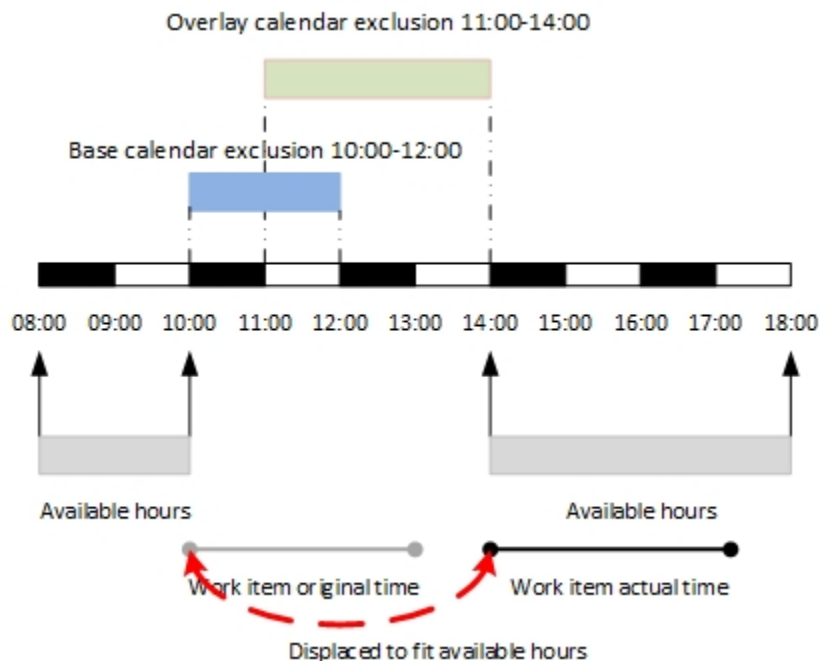


These calculations may not be performed correctly if the base calendar in use does not explicitly specify a time zone. For details, see [Base and Overlay Calendars](#).

## Scheduling Example

In the example shown in this topic, a work item is to be performed in a unit based in California, using a base calendar with the local PST timezone.

The base calendar defines the working day as 08:00 to 18:00. On this particular day the base calendar also defines an exclusion from 10:00 to 12:00. An overlay calendar also applies, defining an exclusion from 11:00 to 14:00. This leaves available working hours for this particular day as 08:00 to 10:00 and 14:00 to 18:00.



A work item arrives. Its start time was defined using `calcDeadline` as 18:00 UTC, and its duration as 3 hours. The start time is converted to 10:00 PST. This puts it in the exclusion defined on the calendars, so the start time of the work item is shifted to the start of available working hours. It is therefore scheduled for 14:00 to 17:00 PST.

## Comments

Comments can be stored against any process instance, work item or case object. You can use these comments to create a custom audit trail, for example.

### Adding Comments

The [AuditService](#) contains the following operations to add comments to an entity:

- [addCaseComment](#) - Adds a comment to a case.
- [addProcessInstanceComment](#) - Adds a comment to a process instance.
- [addWorkItemComment](#) - Adds a comment to a work item.

The request must identify the case, process instance, or work item against which to store the comment, the comment to be stored, and any options for the comment.

The response from the add operations is an audit ID for the comment. You can use this ID to identify the parent comment, which allows you to create a hierarchical view of comments for a BPM process instance, work item or case.

### Retrieving Comments

There are a number of ways to retrieve comments that have been stored against an entity.

- Use the [AuditService](#) operations available to generate an audit trail (which includes the comments) for an entity:
  - [getCaseReport](#)
  - [getProcessInstanceReport](#)
  - [getSearchCaseReport](#)
  - [getWorkItemReport](#)
- Use the [AuditService](#) operations available to retrieve comments from an entity:
  - [getCaseComments](#)
  - [getProcInstComments](#)
  - [getSearchCaseComments](#)
  - [getWorkItemComments](#)
- Use the available Event Collector functions that return comments stored against a process instance, work item, or case. For information about these functions, see [Functions for Comments](#).

## System Actions

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

The table below lists the names, parent components, and default value of all defined system actions.

- **Name** is the name of the system action.
- **Required to execute** lists the web service operations (and Service Connector methods, which use the same names) that require the system action.
- **Component** identifies the BPM component that owns this system action:
  - **BDS** - Business Data Services
  - **BRM** - Business Resource Management
  - **DAC** - Deadline and Calendar
  - **DE** - Directory Engine
  - **EC** - Event Collector
  - **PE** - Process Manager
  - **WSB** - Workspace
- **Default value** is the system-wide default value applied to this system action:
  - **Allowed** - The system action can be performed by any user without authorization.
  - **Denied** - The system action cannot be performed by any user unless they have the correct authorization.

Both the component and the system action name must be specified when calling some [SecurityService](#) functions (for example, [listActionAuthorisedEntities](#) and [listAuthorisedOrgs](#)).

### *System action names and components*

System action	Required to execute	Comp.	Default value
accessGlobalDataScripts	<a href="#">getAuditInfo</a> <a href="#">getCaseModel</a>	BDS	Denied
administerGlobalDataScripts	<a href="#">updateDBScripts</a>	BDS	Denied
applicationConfiguration	Not currently used	WSB	Denied
autoOpenNextWorkItem	<a href="#">deleteOrgEntityConfigAttributes</a> <sup>5</sup> <a href="#">setOrgEntityConfigAttributes</a>	BRM	Allowed
browseModel	<a href="#">getCalendarReferences</a> <a href="#">getOrgModel</a> <a href="#">listOrgModelVersions</a>	DE	Allowed

<sup>5</sup> This system action is required only if you are using these functions to set or to delete the **WorkItemAutoOpen** attribute.

System action	Required to execute	Comp.	Default value
bulkCancelProcessInstances	<a href="#">cancelProcessInstances</a>	PE	Denied
bulkPurgeProcessInstances	<a href="#">checkPurgeTerminatedProcessInstances</a> <a href="#">purgeTerminatedProcessInstances</a> <a href="#">saveTerminatedProcessInstances</a> <a href="#">stopPurgeTerminatedProcessInstances</a>	PE	Denied
bulkResumeProcessInstances	<a href="#">resumeProcessInstances</a>	PE	Denied
bulkSuspendProcessInstances	<a href="#">suspendProcessInstances</a>	PE	Denied
cancelProcessInstance	<a href="#">cancelProcessInstance</a>	PE	Denied
cmisAdmin	<a href="#">deleteDocument</a> <a href="#">deleteOrphanedFolders</a> <a href="#">unlinkDocument</a>	BDS	Denied
cmisUser	<a href="#">createDocument</a> <a href="#">findDocuments</a> <a href="#">getDocumentContent</a> <a href="#">getDocumentMetadata</a> <a href="#">getFolderContent</a> <a href="#">linkDocument</a> <a href="#">moveDocument</a>	BDS	Allowed
cancelWorkItem	Not currently used	BRM	Allowed
changeAllocatedWorkItemPriority	<a href="#">setWorkItemPriority</a>	BRM	Allowed
changeAnyWorkItemPriority	<a href="#">setWorkItemPriority</a>	BRM	Denied
closeOtherResourcesItems	<a href="#">closeWorkItem</a> (in <a href="#">WorkItemManagementService</a> )  Note - This system action is needed only when closing a work item that is allocated to another user.	BRM	Denied
createGlobalData	<a href="#">createCase</a>	BDS	Allowed
createResourceAdmin	<a href="#">createResource</a> <a href="#">updateResource</a>	DE	Denied

System action	Required to execute	Comp.	Default value
deleteGlobalData	deleteCaseByCID deleteCaseByRef	BDS	Denied
deleteCalendars	deleteCalendarEntries purgeCalendarEntries	DAC	Denied
deleteLDAPAdmin	deleteContainer	DE	Denied
deleteResourceAdmin	deleteContainer deleteResource purgeDeletedResources	DE	Denied
executeBusinessService	cancelBusinessService cancelPageFlow injectBusinessService injectPageFlow startBusinessService startPageFlow updateBusinessService updatePageFlow	BIZSVC	Allowed
exportLDAPAdmin	exportResources	DE	Denied
haltedProcessAdministration	getAvailableProcessInstanceVariables ignoreProcessInstance ignoreProcessInstances resumeHaltedProcessInstance resumeHaltedProcessInstances retryProcessInstance retryProcessInstances setAvailableProcessInstanceVariables	PE	Denied
handleProcessMigration	clearMigrationRules getMigrationPoints isSetMigrationRule listMigrationRules setMigrationRules unsetMigrationRules	PE	Denied

System action	Required to execute	Comp.	Default value
importLDAPAdmin	<a href="#">importResources</a>	DE	Denied
LDAPAdmin <sup>6</sup>	<a href="#">executeLdapQuery</a> <a href="#">getCandidateDetail</a> <a href="#">getLdapEntry</a> <a href="#">listAttributeNames</a> <a href="#">listCandidateResources</a> <a href="#">listContainers</a> <a href="#">listLdapConnections</a> <a href="#">saveContainer</a> <a href="#">setExtensionPoint</a> <a href="#">setGroupCandidateQuery</a> <a href="#">setPositionCandidateQuery</a>	DE	Denied
listBusinessServices	<a href="#">listPageFlows</a> <a href="#">listBusinessServices</a> <a href="#">listCaseAction</a> <a href="#">listCategories</a> <a href="#">queryBusinessServices</a> <a href="#">queryCategories</a>	BIZSVC	Allowed
listProcessTemplateAuditTrail	Not currently used	EC	Allowed
manageDataViews	<a href="#">createDataView</a> <a href="#">editDataView</a> <a href="#">deleteDataViewByID</a> <a href="#">deleteDataViewByName</a>	BDS	Denied
openOtherResourcesItems <sup>7</sup>	<a href="#">openWorkItem</a>	BRM	Denied
openWorkItemAuditTrail	Not currently used	EC	Allowed
organizationAdmin <sup>8</sup>	<a href="#">setExtensionPoint</a>	DE	Denied
pendWorkItem	<a href="#">pendWorkItem</a>	BRM	Allowed
purgeProcessInstances	Not used	PE	Denied

<sup>6</sup> For certain [DirectoryService](#) functions, this system action gives the caller access to all organizations, regardless of the organization relationships that are set up. For more information, see [Overriding Organization Relationships](#).

<sup>7</sup> This system action is required only when opening a work item allocated to another user.

<sup>8</sup> This system action is used to override organization relationships when calling [OrgModelService](#) functions. Users possessing this system action can see all organizations when calling [OrgModelService](#) functions that return organization elements, regardless of the organization relationships that are set up. For more information, see [Overriding Organization Relationships](#).

System action	Required to execute	Comp.	Default value
queryAudit	<a href="#">addCaseComment</a> <a href="#">addProcessInstanceComment</a> <a href="#">addWorkItemComment</a> <a href="#">getChart</a> <a href="#">getChartData</a> <a href="#">getCaseComments</a> <a href="#">getProcInstComments</a> <a href="#">getSearchCaseComments</a> <a href="#">getWorkItemComments</a>	EC	Allowed
queryProcessInstance	<a href="#">decodeProcessId</a> <a href="#">getActivityInstanceStatus</a> <a href="#">getParameterValue</a> <a href="#">getProcessInstanceStatus</a> <a href="#">getProcessInstanceSummary</a> <a href="#">listProcessInstanceAttributes</a> <a href="#">listProcessInstances</a> <a href="#">queryDone</a> <a href="#">queryFirstPage</a> <a href="#">queryHaltedProcessInstances</a> <a href="#">queryLastPage</a> <a href="#">queryNextPage</a> <a href="#">queryPreviousPage</a> <a href="#">queryProcessInstanceCount</a> <a href="#">queryProcessInstances</a> <a href="#">queryProcessInstancesAlt</a>	PE	Allowed
queryProcessTemplate	<a href="#">getStarterOperationInfo</a> <a href="#">listProcessTemplateAttributes</a> <a href="#">listProcessTemplates</a> <a href="#">listServices</a> <a href="#">listStarterOperations</a> <a href="#">queryProcessTemplateCount</a> <a href="#">queryProcessTemplates</a> <a href="#">queryProcessTemplatesAlt</a>	PE	Allowed



System action	Required to execute	Comp.	Default value
readCalendars	<a href="#">calcDeadline</a> <a href="#">getCalendarEntries</a> <a href="#">getCalendar</a> <a href="#">getCalendarReferences</a> <a href="#">listCalendars</a> <a href="#">resolveReferences</a>	DAC	Allowed
readGlobalData	<a href="#">getCaseModelBasicInfo</a> <a href="#">readCase</a> <a href="#">navigateCase</a> <a href="#">navigateCaseByCriteria</a> <a href="#">findAllCases</a> <a href="#">findCaseByCID</a> <a href="#">findCaseByExample</a> <a href="#">getCaseReferencesForDataViewByAdhocView</a> <a href="#">getCaseReferencesForDataViewById</a> <a href="#">getCaseReferencesForDataViewByName</a> <a href="#">getDataViewDetailsByApp</a> <a href="#">getDataViewDetailsByCaseClass</a> <a href="#">getDataViewDetailsByCategory</a> <a href="#">getDataViewDetailsById</a> <a href="#">getDataViewDetailsByName</a> <a href="#">getDataViewDetailsByUncategorized</a> <a href="#">getDataViewCategories</a>	BDS	Allowed
readParameters	Not currently used	DE	Allowed
readPushDestinations	Not currently used	DE	Allowed
reallocateToOfferSet	<a href="#">reallocateWorkItem</a> <sup>9</sup> <a href="#">reallocateWorkItemData</a>	BRM	Denied

<sup>9</sup> The [reallocateWorkItem](#) and [reallocateWorkItemData](#) functions allow you to reallocate work items to users in the original offer set if you have the [reallocateToOfferSet](#) system action, or to any user in the organization model if you have the [reallocateWorkItemToWorld](#) system action.

System action	Required to execute	Comp.	Default value
reallocateWorkItemToWorld	<a href="#">reallocateWorkItem</a> <a href="#">reallocateWorkItemData</a>	BRM	Denied
rescheduleWorkItem	<a href="#">rescheduleWorkitem</a>	BRM	Denied
resolveResource	<a href="#">getResource</a> <a href="#">archivedResources</a> <a href="#">findByEntity</a> <a href="#">findByName</a> <a href="#">findByDN</a>	DE	Allowed
resourceAdmin	<a href="#">createResource</a> <a href="#">deleteResource</a> <a href="#">updateResource</a> <a href="#">purgeDeletedResources</a> <a href="#">listCandidateResources</a> <a href="#">setGroupCandidateQuery</a> <a href="#">setPositionCandidateQuery</a> <a href="#">deleteContainer</a>	DE	Denied
resumeProcessInstance	<a href="#">resumeProcessInstance</a>	PE	Denied
scheduleWorkItem	Not currently used by any public API	BRM	Allowed
setDeadlineExpiration	<a href="#">setDeadlineExpiration</a>	PE	Denied
setPriority	<a href="#">setPriority</a>	PE	Denied
setResourceOrderFilterCriteria <sup>10</sup>	<a href="#">getResourceOrderFilterCriteria</a>	BRM	Denied
showProcessInstanceAuditTrail	Not currently used	EC	Allowed
skipWorkItem	<a href="#">skipWorkItem</a>	BRM	Denied
startAndCancelAdHocActivity	<a href="#">runAdhocActivity</a> <a href="#">cancelAdhocActivity</a>	PE	Allowed
startBusinessService	This system action is no longer used. It is superseded by the <a href="#">executeBusinessService</a> system action.	WSB	Allowed

<sup>10</sup> Only the [setResourceOrderFilterCriteria](#) system action at the organization model level is used; the *scoped* [setResourceOrderFilterCriteria](#) system action (i.e., the one at the group, organization unit, and position level) is not currently used.

System action	Required to execute	Comp.	Default value
startprocess	<a href="#">createProcessInstance</a>	PE	Allowed
suspendProcessInstance	<a href="#">suspendProcessInstance</a>	PE	Denied
suspendWorkItem	Not currently used by any public API	BRM	Allowed
updateGlobalData	<a href="#">updateCase</a> <a href="#">linkCase</a> <a href="#">unlinkCase</a>	BDS	Allowed
userAdmin <sup>11</sup>	<a href="#">deleteUserSettings</a> <a href="#">getUserSettings</a> <a href="#">listUserSettingIds</a> <a href="#">saveUserSettings</a>	DE	Allowed
viewWorkList	<a href="#">addCurrentResourceToView</a> <a href="#">deleteCurrentResourceFromView</a> <a href="#">deleteWorkListView</a> <a href="#">getAllocatedWorkListItems</a> <a href="#">getEditableWorkListViews</a> <a href="#">getPublicWorkListViews</a> <a href="#">getViewsForResource</a> <a href="#">getWorkListItems</a> <a href="#">getWorkListItemsForView</a> <a href="#">getWorkListViewDetails</a> <a href="#">unlockWorkListView</a>	BRM	Denied
viewGlobalWorkList	<a href="#">getWorkListItems</a> (when requesting for <i>all</i> resources) <a href="#">getWorkListItemsForGlobalData</a> <a href="#">getWorkListItemsForView</a>	BRM	Denied
workItemAllocation	<a href="#">allocateAndOpenNextWorkItem</a> <a href="#">allocateAndOpenWorkItem</a> <a href="#">allocateWorkItem</a> <a href="#">unallocateWorkItem</a>	BRM	Denied

<sup>11</sup> Note that the userAdmin system action appears as "User Settings" in TIBCO Business Studio, rather than "User Admin". When passed in API calls, pass "userAdmin".

System action	Required to execute	Comp.	Default value
writeCalendars	<a href="#">copyCalendar</a> <a href="#">createCalendarEntry</a> <a href="#">createRecurringEntry</a> <a href="#">createCalendarReferences</a> <a href="#">renameCalendar</a> <a href="#">saveBaseCalendar</a> <a href="#">saveOverlayCalendar</a> <a href="#">updateCalendarEntry</a> <a href="#">updateRecurringEntry</a>	DAC	Denied
writeParameters	Not currently used	DE	Allowed
writePushDestinations	<a href="#">setPushDestination</a> <a href="#">updatePushDestinations</a>	DE	Denied