TIBCO ActiveMatrix® BPM
Client Application Management Guide

Software Release 4.2
August 2017

Document Update: December 2017

Two-Second Advantage®

TIBCO®

Important Information

Contents

TIBCO Documentation and Support Services

How to Access TIBCO Documentation

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

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

Product-Specific Documentation

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

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

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

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

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

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

How to Contact TIBCO Support

You can contact TIBCO Support in the following ways:

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

How to Join TIBCO Community

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

Client Application Development

TIBCO ActiveMatrix BPM provides Application Development to create, develop, and test custom client applications hosted in ActiveMatrix BPM.

Upload application files to Application Development, and then edit, test, and verify changes. For example, keep the service logic of the work list, but completely change the appearance of the layout.

Application Development provides sample applications, components and component services as the starting point for a custom application, and avoids the need to consider complex API interactions. Use Application Development to customize applications with layout and style sheet changes made to HTML, CSS and JavaScript files. For instance, add a company logo, and incorporate the company color scheme. Then publish the custom application, and it is immediately available to users.

The video Creating First Application - Demo shows how to create a new application from a cloned and customized sample application.

Application Development includes the Workapp application. This includes a logon screen for a user to connect to an ActiveMatrix BPM runtime. They can then access business actions and work items. The video My Work Application Demonstration shows how a user would use the Workapp application.

For more information about Application Development, see:

- *TIBCO ActiveMatrix® BPM Client Application Management Guide*
- *TIBCO ActiveMatrix® BPM Client Application Developer's Guide*

TIBCO Mobilespace

TIBCO Mobilespace is available from the Apple App Store and the Google Play Store. Mobile users can use Mobilespace to access business services, and process work items from a smart phone or tablet.

A mobile user visits the app store for their mobile device, and downloads Mobilespace. When they first open Mobilespace, they will need to enter the URL for the TIBCO ActiveMatrix BPM runtime, and their logon credentials. They only need to enter this information once because Mobilespace stores the information between sessions. However, if they choose to log out, they will need to enter this information again when they next log on.

An example URL is: *https://amx-bpm-server/apps*.

Mobilespace includes a standard application: Mobile Case Manager. The mobile user opens Mobile Case Manager, or a custom mobile application, if available. Through these applications, they can manage cases and access business services and work items to complete their daily tasks. They can choose and configure their favorite application.

See: Mobilespace User Video.

A mobile user needs business services and work items that are applicable for a mobile format. TIBCO Business Studio™ includes a **Publish** option for business services to set the **Target Device** to either **Desktop** or **Mobile**. Mobile users can access business services that have the **Target Device** set to **Mobile**.

Use Application Development to customize an application for use with Mobilespace. The process is exactly the same as for desktop applications. Mobile applications use two particular settings in the `ApplicationName.app.desc.json` file. Without one of these settings, the application will not be available to Mobilespace.

Mobilespace is a fully featured mobile app for Apple iOS and Google Android, it has access to the device native API, and does not require mobile specific forms. By contrast, Openspace Mobile uses the system web browser, it does not have access to the device native API, and does require mobile specific forms.

For more information, see:

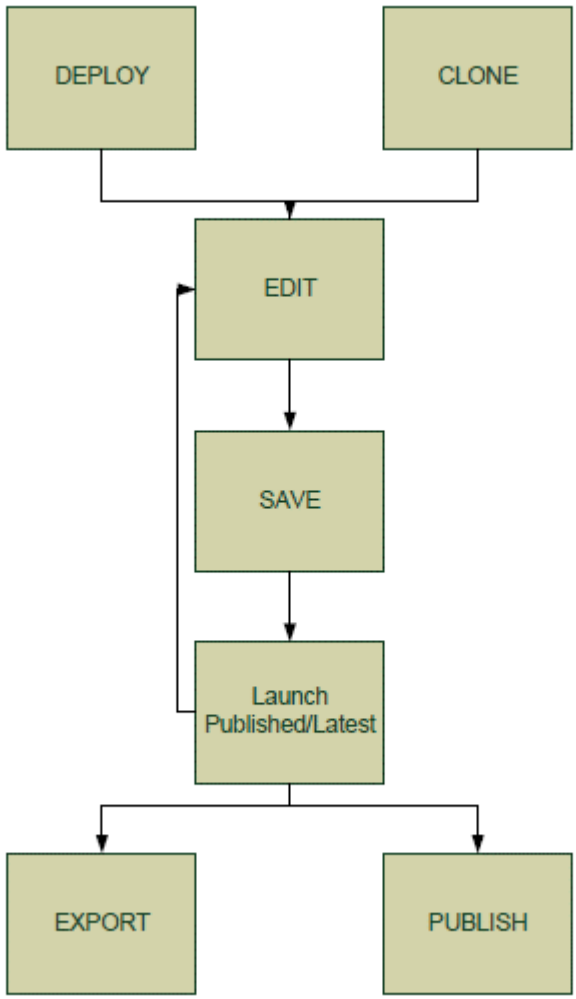
- *TIBCO Business Studio - BPM Implementation Guide*.
- *TIBCO ActiveMatrix® BPM Client Application Management Guide*.
- *TIBCO ActiveMatrix® BPM Client Application Developer's Guide*.

Applications

Custom user interface 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.

Application Lifecycle

An application can be created, edited, tested, published - and can be again re-edited and published. Different users can do this, so you can have multiple versions of an application tailored to different users' requirements. The latest published application version will be served to end user.



You can do the following as part of the lifecycle of an application:

- Create an application (generally by deploying a new or cloning an existing application).
- Clone an application (**CLONE**).
- Edit application files in a file editor (**EDIT**). See Live Editing and Publishing.
- Save changes (**SAVE**)

- Launch and test the application (**Launch** either **Published** or **Latest**, or click  on the application from the Cards view.)
- Export an application to another location, where you can edit it locally before deploying it (**EXPORT**)
- Publish the application (your changes are reflected to others) (**PUBLISH APPLICATION**). Typically, you only do this once you are satisfied with the changes you have made and have tested them.
- Any user can now edit the published application and use their version locally
- **UNDO** reverts the most recent changes.
- Redo changes after an undo (**REDO**)
- Rollback to go back to the previously published version (**ROLLBACK**)
- Upload a new application that you want to include in Application Development (**UPLOAD APPLICATION**) .

The application to be uploaded should have `.app.zip` extension.

Application Version Control

Application Development supports version control for applications. The user always sees the latest published version of an application.

Version Control starts the numbering of released versions of an application at version 1.0. It then increments the version number by 0.1 for each new release. The version numbers are, therefore, 1.0, 1.1, 1.2, 1.3, and so on.

Application Development lists all the published versions of an application, allows reversion to a preceding version, and deletion of any version except the latest. Reversion to a preceding version creates a new version, with a copy of the preceding version. For instance, if an application has five versions, 1.0 to 1.4 inclusive, reversion to version 1.2 creates a new version, 1.5, that is a copy of version 1.2.

Application Development provides two means to access the versions: from the expansion pane on the right, and from the **App Config** tab on the left. The expansion pane and the **App Config** tab allow reversion. Only the **App Config** tab allows deletion, and shows the author for each version.

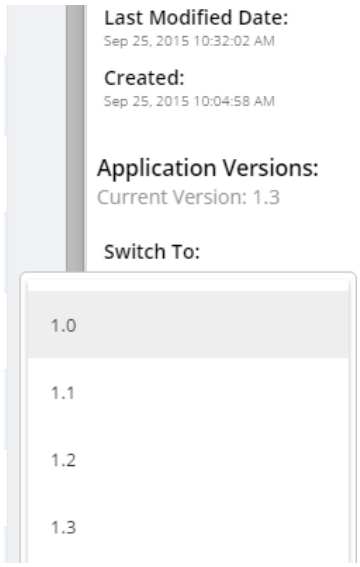
Application Development does not support the simultaneous publication of more than one version of an application. For example, it is not possible to publish versions 1.0 and 1.1.

Revert to a Previous Application Version

The expansion pane on the right allows reversion to a previous application version.

Procedure

1. On the Applications page, double-click the required application.
2. Click  on the expansion pane on the right.
   The Application Information pane opens, and lists the current version.
3. Under **Switch To:**, click **Select Version**.
   This opens a list of published versions.



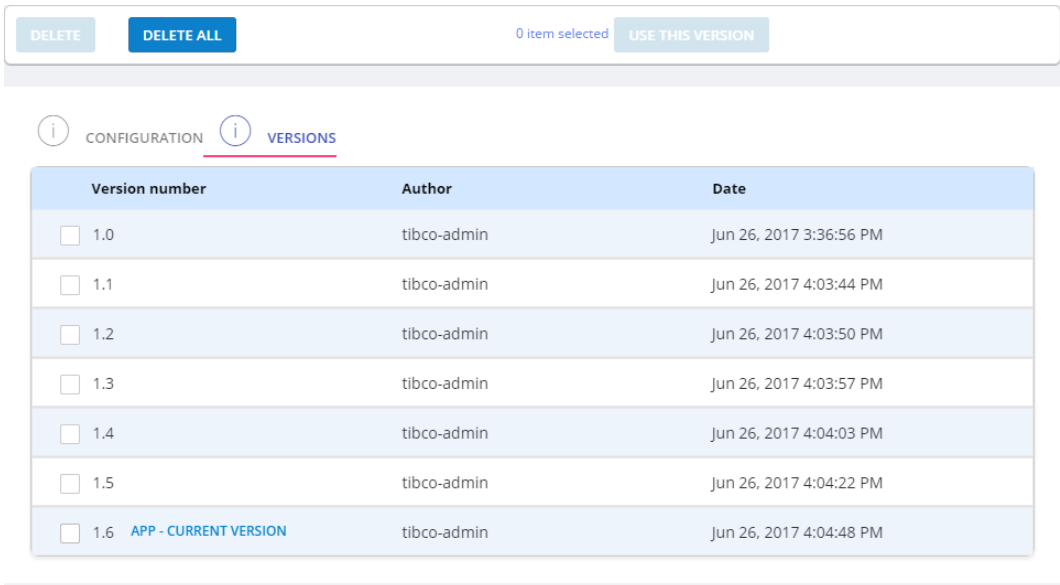
4. Select the version to revert to.
   The Revert window opens.
5. Click **YES** to revert to the chosen version.
   A message opens: Version *n* is cloned as a new version and the cloned version is used as current version.

Revert or Delete an Application Version

The **App Config** tab on the left allows reversion to a previous application version, deletion of a one or more application versions, and deletion of all previous application versions. It also shows the author for each release.

Procedure

1. On the Applications page, click the required application.
2. Click the **App Config** tab.
3. Click **VERSIONS**
   This opens a list of published versions, that includes the author for each version.



4. To revert to a previous version, do the following:
   a) Select the check box for the required version.
   b) Click **USE THIS VERSION**.
   A message opens: Version *n* is cloned as a new version and the cloned version is used as current version.
5. To delete a previous version, do the following:
   a) Select the check box for one or more required versions.
   b) Click **DELETE**.
   The Confirm delete window opens.
   c) Click **YES** to delete the chosen versions.
   A message opens: Version(s) *n* has been deleted.
6. To delete all previous versions, do the following:
   a) Click **DELETE ALL**.
   The Confirm delete window opens.
   b) Click **YES** to delete all previous versions.
   A message opens: Version(s) *n*, *n + 1*, *and so on*, has been deleted.

Editing and Publishing

You can edit deployed application files, and publish the application to make the revised application available to all users.

More than one user can edit the same file, but if this happens when you **SAVE**, you see an alert that the version you are trying to save is not the latest and you have the options to overwrite or cancel.

You can clone applications, which means you can use a relevant application and tailor it for your own requirements. See Cloning Applications.

The following file formats are supported for editing:

- HTML
- CSS
- JavaScript
- XML
- PROPERTIES
- JSON

You can use your preferred third-party editors, however TIBCO supports only the in-built editors.

You can make changes to a file, save it, and refresh the browser to see the result of the changes within the deployed runtime application.

You can view and edit different published versions of an application (so it is always possible to revert to a previous published version if you do not want to use a more recent version). See Switching Between Application Versions.

Versioning is not available for individual files. For files, you can use **UNDO** to revert the most recent changes.

To enable editing of files and applications, see Editing Applications.

Setting Logging on Applications

Application Development records the messages that are returned from your browser. You can diagnose problems with your applications by configuring the different levels of messages.

The messages are written to the BPM log appender. See "Configuring TIBCO ActiveMatrix BPM Auditing" in *TIBCO ActiveMatrix BPM Administration Guide* for more information.

Procedure

1. Click **Settings**.
2. Click  to expand the debug settings.
   A list of available packages is displayed. If you hover over a package name, a description of the package is displayed.
3. To display the available message levels, click the arrow next to the package whose log level you want to set.
   There are five different levels of messages that you can record.

Action	Description
WARN	Messages recorded at this level mean minor potential problems or reduction in system performance. Investigation and rectification of the error is recommended at the first suitable opportunity.
INFO	INFO displays audit level information messages. Messages recorded at this level mean the system is running normally and no remedial action is required.
DEBUG	DEBUG records low-level diagnostic information.
TRACE	TRACE is the lowest level of message that you can record. This is the most verbose level and should only be turned on when diagnosing problems. Messages recorded at this level mean the system is running normally and no remedial action is required.
ERROR	Messages recorded at this level mean there could be potential data integrity problems, reduction in system performance, or a system component is experiencing a serious problem. Urgent investigation and rectification of the error is required.

4. Select the message level you require.
5. Click **REVERT** to set the log level. Repeat Step 3 - Step 5 for each package whose log level you want to set.
6. Click **SAVE CHANGES** to save your changes.
7. You can return the logging level for your packages to their default configuration by selecting **RESET ALL TO DEFAULT**.

Manage Client Applications

Application Development is used to browse, customize, save, preview and publish applications. Applications are hosted on the platform, so when users publish them, they are available to other users.

A number of third-party common libraries are available for use under `app_cdn`.

You can do the following:

- Edit the files in an application in a file editor, change, save, test and publish the application. Application Development displays when it was modified. The application can be re-edited and published again. See Editing Applications.
- The following properties can be passed to an application from the *`appName`*`.app.desc.json` file:

  For the properties marked as mandatory, you can modify their value, but do not unset the property, otherwise the application may not function properly.

  - `appImage` (mandatory for applications targeted for the mobile devices. Not applicable to the desktop applications.) - The path to the image depicting the application.
  - `category` (optional) - A category used to group your applications.
  - `configPage` (mandatory) - The path to the configuration file for the application.
  - `defaultLocale` (optional) - The locale in which the application is displayed.
  - `description` (mandatory property for applications targeted for mobile devices.) Short description for the application.
  - `errorPage` (optional) - The path to the error page for the application.
  - `indexPage` (mandatory) - The path to the `index.html` file for the application or the landing page for the application.
  - `library` (optional: default=false) - Boolean flag that describes if the application is a library that can be used by other applications.
  - `loginPage` (mandatory) - Application is directed to this page when the user logs in. Note that the out-of-the-box example application in Application Development use the login component provided by the framework -- `/apps/logincomponent/login.html`. Therefore, you have the option of using that same login component, or redirect to a custom login page using this property.
  - `mobile` (optional: default=false) - Boolean flag that describes if the application is for mobile devices.
  - `support` (optional) - Languages supported by mobile applications.

  Removing any of the mandatory properties may prevent the application from operating correctly.
- Select the **App Config** tab to view and edit the configuration file for the application.
- Upload/Import applications (**UPLOAD APPLICATION**)
- Clone applications (**CLONE**). You can specify a name for the cloned application. See Cloning Applications.

  You need to update the `app.desc.json` file.
- View the status of applications - for example, tested and published.

  Status is currently only displayed as **PUBLISHED**.

- Export/download applications (**EXPORT**).

  You can work offline and redeploy them.

- Preview the changes to applications (**Launch** - either **Latest** or **Published**). See Launching and Testing Applications.
- Use versioning to access previously-published versions of an application. See Application Versioning.
- Multiple users can access applications, but concurrent editing and merging changes on an application by multiple users are not allowed.

  More than one user can edit the same file, but if this happens when you **SAVE**, you see an alert that the version you are trying to save is not the latest and you have the options to overwrite or cancel.

- Control permissions using the **Permission** tab - which defines which applications are editable, and which are read-only. This is the context in which you can access the application. See Controlling Permissions.
- Publish applications to make them available to other users (**PUBLISH**). See Publishing Applications.

How to Access Application Development

Procedure

1. To access Application Development, type the login URL into your browser.
   The URL is: `protocol://host:port/apps/apa/login.html` where:
   - `protocol` is the communications protocol being used by Application Development, which is either http or https. This was determined at installation.
   - `host` is the DNS name or IP address of the server hosting the BPM runtime.
   - `port` is the port being used by the BPM server. The default value is 8080.
2. After the browser connects, the Login screen is displayed.

How to Set the Session Timeout

You can specify that if a user of an application developed with Application Development is inactive for a certain period of time, the user's session will time out and automatically log the user out. The default session timeout is 30 minutes.

Note that if the session times out, the user is not informed of the timeout. If the timeout occurs when a form is displayed, and the user performs an action such as clicking **Submit** or **Cancel**, the form is closed and the user is returned to the login screen. If data had been entered on the form during the session, it is lost.

Procedure

1. Open the following file using a text editor:

   CONFIG_HOME\bpm*bpm_app_name*\configuration\OSProperties.properties

   where:

   - *CONFIG_HOME* is where ActiveMatrix BPM configuration information is stored. On Windows, this defaults to:

     `C:\ProgramData\amx-bpm\tibco\data`

- *bpm_app_name* is the BPM application name specified when ActiveMatrix BPM was installed. The default is:

  `amx.bpm.app`

2. Change the value of the `sessionTimeout` property to the desired the number of seconds for the timeout.
3. Save and close the `OSProperties.properties` file.

Edit an Application

You can edit an application, test the changes, save it and publish it.

Procedure

1. Click  to display the menu options.
2. Click **OPEN** for the selected file/application.
3. Double-click on the file you want to edit in the file editor. You can now start editing.
   You can select **UNDO** or **REDO** if you want to undo or redo your most recent editing changes.

   You may want to edit more than one file in an application. You can use the  icon to move between files you are editing within an application.
4. **PREVIEW** your changes. If you are happy with the result, go on to the next step. If not, you can edit the file again.
5. **SAVE** your changes. The saved changes are visible to you, but not to other users.
6. Return to the List of Applications, select the application you have been editing, and select **PUBLISH APPLICATION**.
   Once you have published your changes, they are available to other users with permission to access this application.

Clone an Application

You can clone an existing application, then edit the cloned application, test the changes, save it and publish it.

Procedure

1. Click  to display the menu options.
2. Select the application you want to clone.
3. Click **CLONE**.
4. Enter a new application name and click **OK**.
5. You will see that your cloned application is successful. Click **OK**.
6. You can now start editing your cloned application.

Launch and Test an Application

You can launch applications either before or after publishing them. This allows you to test your application. If you still need to make changes after launching, edit the application again, re-launch and then publish.

To launch the published application, click  on the application from the Cards view.

Procedure

1. Click  to display the menu options.
2. Select the application/s you want to launch.
3. Click **Launch** and then select either **Published** or **Latest**.
4. The application opens in a new tab. Test whatever you require, and close the tab when you have finished.
5. You can then either make more edits to the application, or publish it if you have finished working on it and want to make it available to other users. See Publishing Applications

Upload an Application

Upload an application to include in Application Development.

Application names are generated using the `title` key in the message properties files in the `l10n` folder. If no `title` key is defined, then the name of the application zip file uploaded to Application Development is used.

Procedure

1. Click **UPLOAD APPLICATION**.
2. Click on the plus sign and browse to select the application zip file to upload.

The application to be uploaded must have the `.app.zip` extension.

Delete an Application

You can delete applications using Application Development.

Procedure

1. Click  to display the menu options.
2. Select one or more applications you want to delete.
3. Click **DELETE**.
   You will be asked to confirm that you want to delete the selected applications.
4. Click **YES** to confirm the deletion.
   You will be informed that the deletion was successful.
5. Click **OK**.

Result

The applications you have deleted will no longer be visible in the list of applications.

Publish an Application

You can publish applications, meaning that they will be available to other users.
Until you publish an application it is only available locally to the user who is editing it.

Procedure

1. Click  to display the menu options.
2. Select the application you want to publish.
3. Click **PUBLISH**.
4. You will be informed that the application is published. Click **OK**. The application is now available to other users.

Mobile Application Settings

Mobile applications use two particular settings. One of these settings defines the application as a mobile application, and makes it available to Mobilespace.

The two settings are:

- The setting: `"mobile" : true`. This defines a mobile application, and makes it available to Mobilespace.
- The setting: `"appImage" : filepath`. This is the path to an icon that represents the application.

These settings are in the *ApplicationName*`.app.desc.json` file:

```
{
   "indexPage" : "ApplicationName/index.html",
   "mobile" : true,
   "appImage" : "ApplicationName/img/iconFile.svg",
   "configPage" : "ApplicationName/ApplicationName.config.json",
   "description" : "Application description"
}
```

For best results, the icon that represents the application, `"appImage"`, should meet the following criteria:

- It is typically an SVG image file, but raster images are acceptable.
- The image should be square, with a 1:1 width - height ratio.
- A raster image should be one of these formats: PNG, JPG, JPEG, BMP, or GIF.
- A raster image should have a minimum size of 512 x 512 pixels.

Procedure

1. Follow: Edit an Application
2. Choose the application, locate and open the *ApplicationName*`.app.desc.json` file.
3. Add or edit the two settings for `"mobile"` and `"appImage"`.

   `"mobile" : true`

   `"appImage" : filepath`
4. Click **SAVE**.

Control Application Permissions

Control application permissions with the **Permission** tab. This defines which applications are editable, and which are read-only. This is the context in which you can access the application.
Permissions can be set to one of the following:

- System: This permission belongs to an application provided with the product , and does not allow any changes to it. Clone an application with the System permission, and then assign to it an alternative permission.
- Full Access.
- Hidden. Use this permission for an application that is not included in the list of applications.

  The application still exists and is available from the appropriate URL, for example, `http://localhost:9999/apps/visibleTest/index.html`.
- Read-only.
- Library. Use this permission for common files and folders that different applications may require. Library applications are hidden by default. They are not available on the applications page.

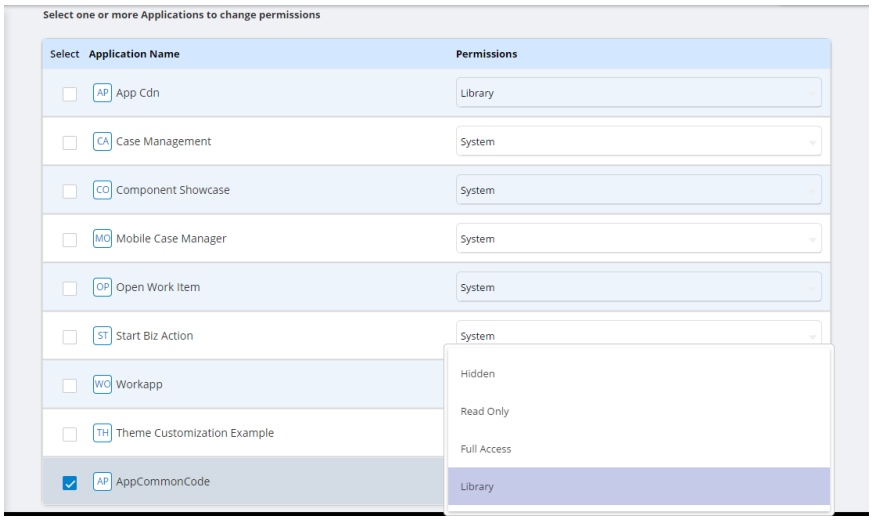
To create the Library permission, edit the *`ApplicationName`*`.app.desc.json` file, and add the line `"library":true`. The Library permission is optional, so if it is false, do not add a second Library permission.

```
{
"indexPage":"small/index.html",
"configPage":"small/small.config.json",
"library":true
}
```

Use the following procedure to change the permission to Library from **Permission**.

Procedure

1. Select **Permission**.
2. Select  to edit permissions.
3. Select the application whose permissions you want to edit.
4. Select the **Library** permission from the list.



5. Click **SAVE PERMISSIONS**.

After the **Save Permissions Successful!** dialog box, return to the Applications list. The application no longer appears in the list. To reinstate it, change the permission again to either **Read-Only** or **Full Access**.

Export an Application

You can export an application to another location, where you can edit it locally before uploading it back into Application Development.

Procedure

1. Click  to display the menu options.
2. Select the application you want to export.
3. Click **EXPORT**.
   When you have exported the application, it will have the suffix `.app.zip`. You can extract the contents of the zip file and edit it locally.
   See Uploading Applications for details on how to upload your application back into Application Development when you have finished editing it.

Work Offline With Application Development

You can deploy an application, and work offline in RAD (rapid application development)/JAD (joint application development) mode to edit and test.

Once you enter RAD/JAD mode, all the static resources continue to be served from the RAD/JAD host. You remain in RAD/JAD mode within your browser session until the RAD/JAD mode is disabled. The RAD JAD session is effective in all the tabs or windows opened within a given browser.

If you want to view the content from the BPM server location, you need to access it using a different browser. For example, you can use a RAD/JAD session in the Firefox browser and you can access the content from the BPM server in the Chrome browser.

Trying to use multiple tabs in a given browser session to access the content in the RAD/JAD mode and accessing the content directly from the BPM server does not work. In this case, these tabs work as if they are set in the RAD/JAD session.

When accessing an application in RAD/JAD mode, you must use a local machine URL. If you access an application using a remote URL, RAD/JAD mode is disabled.

To use RAD/JAD mode, you must be logged in as a user that has the EditApplication system action assigned. Also see: Application Development System Actions.

Procedure

1. Export the application.
2. Unzip the exported application to either the local file system, for example:

   `file:///<local drive>:/<apps>`

   or to an external web server so that the application can be served from the web server.
3. Access the runtime.

   In the URL, provide parameters to run in RAD/JAD mode, as follows:

   `http://127.0.0.1:<port>/apps/<app name>/<landing page>?bpm_app_rad_enable=true&bpm_app_rad_artifacts_url=<local file directory or external server URL>`

- Set the `bpm_app_rad_enable` parameter to `true` to invoke RAD/JAD mode: `bpm_app_rad_enable=true`
- Set the `bpm_app_rad_artifacts_url` parameter to a local file system path or a *URL* to an external web server hosting the resource artifacts for the client application.

If you specify a directory in the **`bpm_app_rad_artifacts_url`** query parameter, that directory must be on the local file system.

4. Do one of the following:
   - If you are hosting the application from an external web server, modify static resources files like HTML, CSS styles sheets, JavaScript, images, XML and JSON files.
   - If the application is in the local file system, modify the files (HTML, CSS styles sheets, JavaScript, images, XML and JSON files) present on the local file system.
5. Refresh the application by reentering the URL with the RAD/JAD parameters (see Step 3.)
6. To exit RAD/JAD mode, set the parameter in the URL to `bpm_app_rad_enable=false`.

   The RAD/JAD exit URL is in the following format:

   `http://127.0.0.1:<port>/apps/<app name>/<landing page>?bpm_app_rad_enable=false`

Application Roles

Application roles are used to define the access control for applications. Users can access an application itself or a feature/component within that application, based on the application roles you assign them - for example: user, administrator, supervisor.

With application roles, you can ensure that only the relevant user interface is displayed to the appropriate users according to the tasks they are expected to perform within the application. For example, for an administrative application, the user interface displayed to an administrator (with the 'administrator' role) is different from the user interface displayed to a basic user (with the 'user' role). You can configure an application so that you assign multiple roles to users or groups.

Application roles are defined independently of the organisational model. Use of application roles is optional.

Application roles are not provided as part of the MyWorkApp application, but can be used by customer-developed applications.

Creating Application Roles

An application role configuration contains all the information necessary to appropriately configure an application for a particular user role. Application roles can control access to a single or multiple applications.

Application roles are available through the object API and so can be used directly by the UI application developer.

The Application role configuration must contain information such as:

- Role name
- Role description
- Locations of applications/components and their related configuration files

Roles are defined in the role configuration file (*roleName*`.rol.json`). For example, `admin.rol.json`:

```
{
  "name": "admin",
  "description": "administrator role",
  "applications": {
    "/apps/workViews": {
      "name": "workViews",
      "accessLevel": "admin"
    },
    "/apps/processViews": {
      "name": "processViews",
      "accessLevel": "user"
    }
  }
}
```

The *rolename*`.rol.json` file contains the following types of information.

Key	Description	Additional Notes
name	Role name	The role configuration file has the name *Role-Name*.rol.json.  We recommend naming the file to match the role.  If two different files have the same role, the filename can be different to the role. For example, you can have rolea.rol.json and roleb.rol.json (both containing the admin role). When this is processed, the second copy of admin is ignored and you will get a warning. When deleting, if the first admin role is deleted, the second admin role is processed and displayed, so it can appear that the role was not deleted.
description	Description of the role	
applications	List of applications	List of applications that the user belonging to this role can see.
/apps/appViews	Location of the application	Location of the application that is deployed into the Application Development platform.
"name": "workViews"	Name of the application	
"accessLevel": "admin"	Name of the configuration (customization) for the above mentioned application	Customized configuration file will be resolved using *app-name_accessLevel*.rac.json . In the above example: workViews_admin.rac.json.

The configuration of an application role consists of a list of qualifying organizational entities (groups and positions) that are maintained with the application role. If a user is a member of any of these organization entities then they qualify for the application role. Each user can be associated with a number of roles based on a mapping. See Mapping a Role to an Organization Entity and Mapping Roles to Applications.

Roles can be deployed into the Application Development platform using the standard zip archive (the name of the archive should have the suffix `.role`, for example, `adminrole.role.zip`.). See Uploading a Role.

The roles.map.json File

You can map a single role or multiple roles to an organizational entity (which exists in the organization model).

The `roles.map.json` file lists the GUID of an organizational entity and maps it to one or more roles.

Mappings are saved to this file but it is for internal use only and mappings should be updated using Mapping Roles to Applications and Components.

It is possible for organization entity mappings to get out of date, if organization entities are deleted as part of a major version upgrade of the organizational model. If this occurs and the mappings are not updated, then it can affect the roles that require any of the deleted organizational entities.

Mapping Roles to Applications and Components

In the *`Role-Name`*`.rol.json` XML configuration file, you can associate roles with multiple applications or components of applications.

In this example, the `user` role is associated with two applications, `workViews` and `componentApp`.

roles.map.json | user.rol.json

```
{
  "name": "user",
  "description": "user role",
  "applications": {
    "/apps/workViews" : {
      "name": "workViews",
      "accessLevel": "user"
    },
    "/apps/componentApp": {
      "name": "componentApp",
      "accessLevel": "user"
    }
  }
}
```

When an application has been associated with a role, only users with that role can access that application. So in this example, the user `user` can access `workViews` and `componentApp`.

This is not enforced, so the custom application needs to enforce it.

In addition to this mapping each application has additional mappings stored in application-specific `.json` files. Each application can have its own settings and also a list of components which make up the application. Each component defined also has a `.json` file which can store settings for the component itself.

Application Roles Configuration Files

A number of configuration files are used to control how application roles are used.

The names of the files must be as follows:

File	Description
RoleName`.rol.json`	Role descriptor file.  We recommend naming the file to match the role.  If two different files have the same role, the filename can be different to the role. For example, you can have `rolea.rol.json` and `roleb.rol.json` (both containing the admin role). When this is processed the second copy of admin is ignored and you will get a warning. When deleting, if the first admin role is deleted, the second admin role is processed and displayed, so it can appear that the role was not deleted.
ApplicationName_accessLevel`.rac.json`	Application custom configuration file, where *applicationName* is the name of the application and *accessLevel* is the access level required to access this application.
ApplicationName`.rcd.json`	Application default configuration file.
ComponentName`.rcd.json`	Component default configuration file.
ComponentName_accessLevel`.rcc.json`	Component custom configuration file, where *componentName* is the name of the component and *accessLevel* is the access level required to access this component.

For example:

```
{
"name": "admin",
"description": "administrator role",ex
"applications": {
"/apps/rolesShowcase":
{ "name": "rolesShowcase", "accessLevel": "admin" }
}
}
```

The structure of the directory of configuration files is important. There is a separate `.json` configuration file for each application or component.

The root folder contains the roles (`.rol.json`) and mapping file. Under this folder there are applications and components folders.

- These need to match the path specified for the application or component in the role. So if the role refers to an application with the path `/applications/myWorkApp`, the configuration files for that application need to be in the `applications` folder and named as `myWorkApp.`*xx* (*xx* could be `.rcd.json` or `_admin.rac.json` for example.).
- The same applies for components. If the path is `/components/component1` the files need to be located under the `components` directory.
- You can create extra levels of structure, so if you want `:/components/core` and `/components/extra` you could create those folders and then specify the appropriate path in the role. **The path must match the folder structure.**

Application authors should provide a default configuration (`.rcd.json`) along with the application. Customised configurations are provided by deploying a new configuration file associated with a particular application/component.

For example `myWorkApp` can have a default configuration of `myWorkApp_rcd.json`.

```
{
"settingsMap": {
    "pageSize": {
      "defaultValue": "10"
    },
    "showWorkViews": {
      "defaultValue": "true"
    },
    "showFavourites": {
      "defaultValue": "true"
    }
  }
}
```

To provide a customization of the `myWorkApp` deploy a customized configuration with the name matching to the *ApplicationName_accesslevel* `_rac.json`.

```
{
  "settingsMap": {
    "pageSize": {
        "value": "30"
    },
    "showWorkViews": {
        "value": "false"
    },
    "showFavourites": {
        "value": "true"
    }
  }
}
```

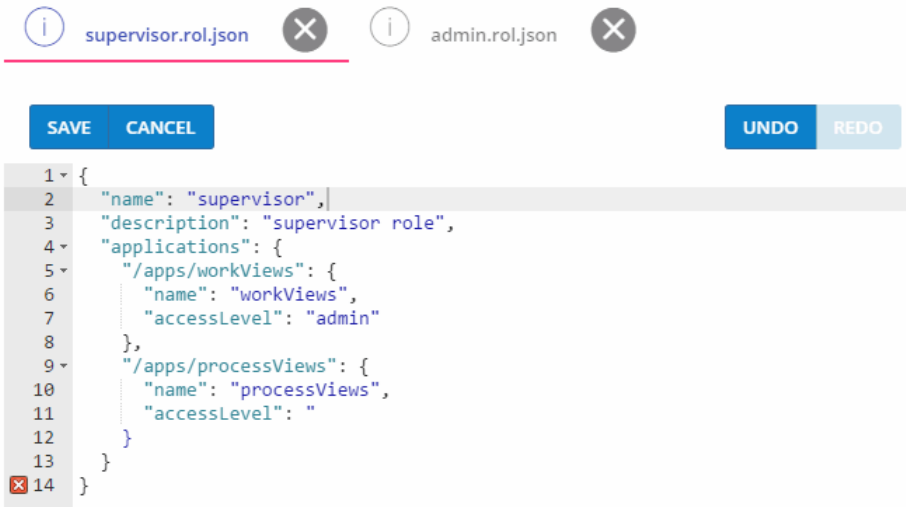
Viewing and Editing Application Roles Configuration Files

You can view and edit application roles configuration files.

Procedure

1. Select **Roles**.

2. Click the icon to display the editing options.
3. Click the application role you want to select.
4. Double-click the selected role to reveal the list of configuration files.
5. Double-click the filename you want to view or edit.
6. View or edit the contents of the file. You can **SAVE**, **CANCEL**, or **REDO** or **UNDO** any changes.



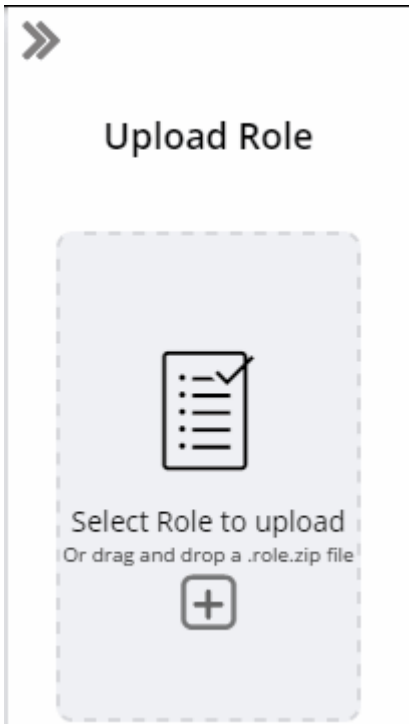
Uploading Roles

You deploy the set of application roles configuration files as a zip file.
You can import `.json` files and edit them to meet your requirements.

If you upload a zip file containing roles, the latest roles you upload overwrite previous copies (for example, if you uploaded a user role, and then uploaded a zip file containing a user role, the user role in the zip would overwrite the previously-uploaded user role. So in this case the upload will act as a merge.

Procedure

1. Zip up your application role configuration files (which must have a specific name ending in `.role.zip`). See Application Configuration Files.
2. From the Roles page, open **Upload Role** on the right hand side of the screen and click on the plus sign.



You can also drag and drop to import a `role.zip` file.

Alternatively, you can upload the roles files using **UPLOAD APPLICATION** at the top of the screen.

3. Navigate to the *Name*`.role.zip` file you want to import and click **Open**.
   The application roles you have uploaded are now available for you to view and edit.

Deleting Roles

You can delete roles using Application Development.

When you delete roles you cannot subsequently undelete them, so you must be sure that you no longer require them.

Procedure

1. Select **Roles**.
2. Click  to display the menu options.
3. Select the role/s you want to delete.
4. Click **DELETE**.
5. You are asked to confirm that you want to delete the selected role/s. Click **YES**.
6. You are informed that the deletion was successful. Click **OK**. The role/s you have deleted are no longer visible in the list of roles.

Mapping Organization Model Positions to Application Roles

You can create, edit, view and delete mappings of organization model positions or groups to application roles from **Roles**.

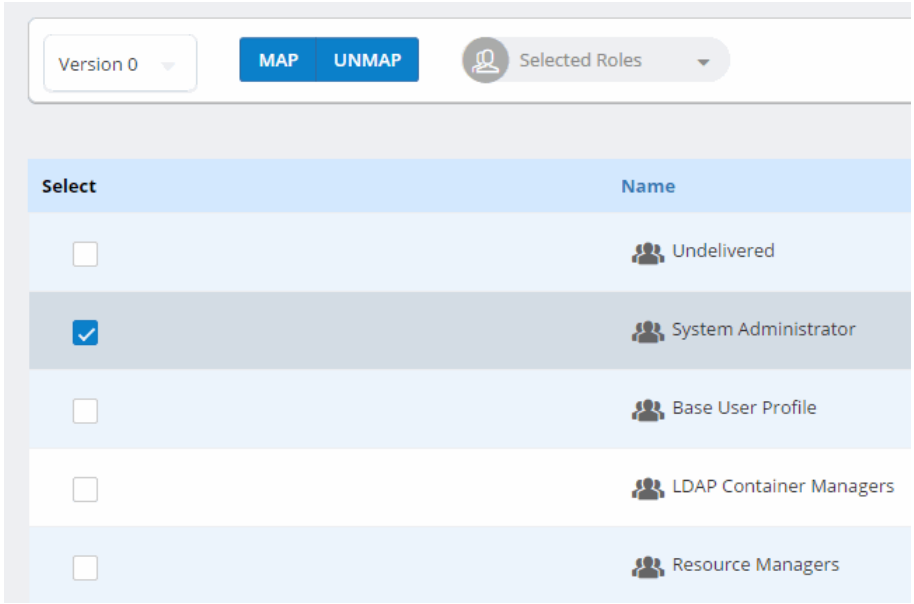
If you have a number of roles defined, use the Search Roles box to find roles you want to manage.

Procedure

1. Select **Roles**.
2. Click the icon to display the editing options.
3. Do one of the following:
   - Select **SELECT ALL** to select all the roles.
   - Click the checkbox on one or more individual roles to select them.

   Select **DE-SELECT ALL** to deselect roles.
4. Select **MAP OR UNMAP ROLES**.
5. Select which **Version** of the organization model you want to use.
6. Select each Name of a position or group you want to add to your role. For example, you can add System Administrator to the Supervisor role.

   You can only select positions or groups. Other organization model entries have a checkbox, but you get a no entry sign if you try to select it.



   Use the dropdown on **Selected Roles** to remind yourself which roles are selected.

   To help with selection from entities within the currently selected organization model version that match the characters you've entered, key characters into Search Org Models box to limit your list.
7. Select **MAP** or **UNMAP**.

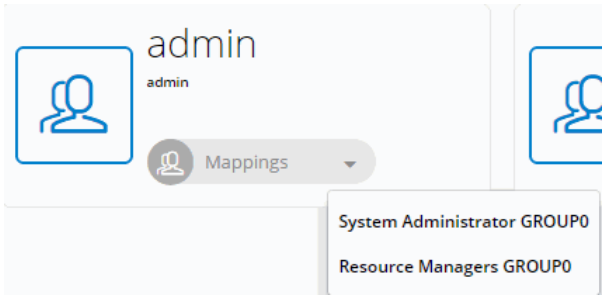
Using the Roles Example

A roles example called **rolesShowcase** is provided which you can use to demonstrate how roles work.

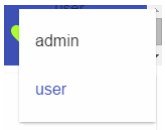
If you do not see what you expect at any stage, remember to refresh your screen using .

Procedure

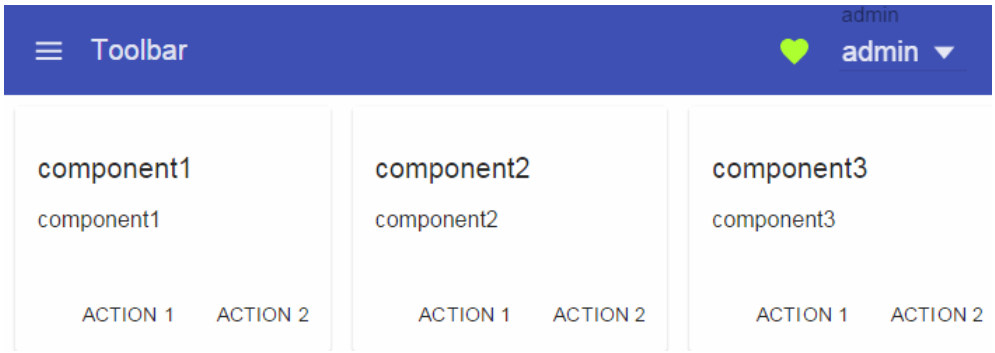
1. Download the following roles example files: rolesShowcase.role.zip and rolesShowcase.app.zip.
2. Upload the roles example files into Application Development in the following order: `rolesShowcase.role.zip` and `rolesShowcase.app.zip`.
3. From Applications, select **rolesShowcase** in the application list.
4. On Roles, map the admin and user roles to groups. See Mapping Organization Model Positions to Application Roles. For example, for admin the mappings can be as shown:

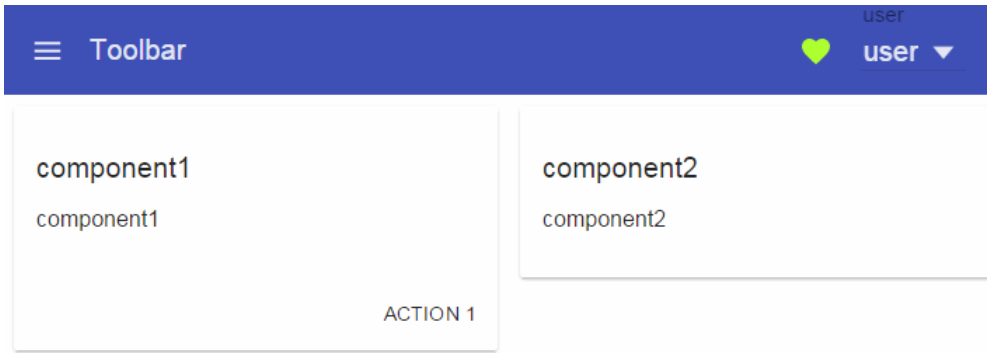


5. Map **tibco-admin** to the same groups using the organization model in Openspace.
6. You now see the roles in the sample, and can switch between views using the dropdown depending on whether you have an admin or user role.



For example, admin has more actions available to them than user.




Toolbar
user
user
component1
component1
ACTION 1
component2
component2

Localizing Applications

The localization service is used to provide one or more languages to applications that are developed using Application Development. You can quickly and easily add new locales by dragging and dropping a language pack to your application. You can dynamically customize and modify locale information for your application.

The applications must have been internationalized. In other words, your application developer has provided localization support by externalizing the necessary keys in the application.

The following terms are used in the localization service:

- **Locales** - identify a specific language spoken in a geographic region.
- **Language Packs** - consist of a collection of message properties files for locales. Language packs are used for distribution. For example, you may have an Asian language pack and a European language pack.
- **Message Property Files -** consist of key/value phrases.
  - **keys** - the localizable terms specified in the application by your application developer.
  - **value** - the strings used with the keys for a locale.

Using the Application Development Platform localization service, you can:

- define your own language packs for your applications.
- modify the existing localization data for an application. For example, you may want to add a new locale, or define a new fallback chain or add new key/value phrases.

A video called Localizing Applications is available that shows how to add and customize locale information for you application.

Application Localization

You can use message properties files to enable you to localize your applications.

Applications are made up of components. Components are reusable program building blocks that can be combined with other components to form an application. Depending on how you have developed your application, your application could have:

- no message properties files.
- a default message properties file for the whole application.
- a message properties file for each component that makes up your application.

It is best practice, when creating an application that may require localization, to create at least one default message property file for the application. When you want to localize the application, you can use the default message property file as a template, and then create a new message property file for each language you want the application to use. If you also want the application to be available in different locales within that language, create a message property file for each locale.

Depending on the size of your application, you may want to create a message properties file for each component in your application. Therefore, for an individual component in an application, you may have:

- *component*Messages.properties - default language.
- *component*Messages_fr.properties - French language but no specific locale.
- *component*Messages_fr_CA.properties - French language/Canadian locale.

For example, you may want a component called AboutMessages to be available in the French language. A French language pack could contain 6 locales. In this case, the AboutMessages component could contain:

- `AboutMessages_fr.properties`
- `AboutMessages_fr_BE.properties`
- `AboutMessages_fr_CA.properties`
- `AboutMessages_fr_FR.properties`
- `AboutMessages_fr_LU.properties`
- `AboutMessages_fr_MC.properties`
- `AboutMessages_fr_CH.properties`

Creating Language Packs

You can create your own localization resources by creating the message property files that represent your components and translating the key/value phrases into the language you require.

Procedure

1. Copy the default message property file that contains the translatable key/value phrases in the default language.
2. Rename the message property file so that the file name includes the new language key and optionally, the locale key.

   How you rename your message properties file depends on whether you are creating a message property file for a language or for a locale within a language. If you are creating a message property file for:

   - a language, the name of the file must be in the format of *componentName*Messages_*ll*.properties. For example, if you are translating from American English to Spanish, then `LoggerMessages.properties` file must now be called `LoggerMessages_es.properties`.
   - a locale within a language, the name of the file must be in the format of *componentName*Messages_*ll_CC*. For example, if you are translating from American English to Mexican Spanish, then `LoggerMessages.properties` file must now be called `LoggerMessages_es_MX.properties`.

   The definition of *ll_CC* is

   - *ll* is a two-letter ISO 639-1 code which identifies the language.
   - *CC* is a two-letter ISO-3166-1 code that identifies the locale.

   If you want the Application Development Platform to use your new message properties file, you must at least specify a language code.

   For a list of all codes, go to https://www.iso.org and search for **iso 3166 codes**.
3. Translate the strings used with the keys.
4. Save the files as a zip file.

   - The zip file must be named *name.locale*.zip.
   - The file must be in a `/l10n` folder in the zip.

Result

You can now import the language pack to your application. See Importing Language Packs.

Importing Language Packs

For a language pack to be available for use with an application, you must import the language pack into the application.

When you import a language pack, an *applicationname*`.locale.zip` file is merged with the current application. The localization files should be in the root folder. However, once the localization files are merged, they are stored in an l10n folder in the application.

Prerequisites

Make sure your language packs are available. See Creating Language Packs.

Procedure

1. From the Application List view, select the application into which you want to import the language pack.
2. Click **Language**.
3. Click  to display the menu options.
4. Click **Import** and browse to the *name.locale*.zip file you require.
5. Click **OK** to save your changes.
   Once you have saved your changes to an application, the application is saved to memory and the new language is immediately available to the application.

Result

You are now ready to test and publish your application.

Exporting Language Packs

You can export the languages and localization resources used by an application. You may want to do this if you want to import it for use with another application. For example, your company may have opened a new office in another country and you want to create a new language pack for the application for the new location.

After exporting, a *applicationname*`.locale.zip` file is created where *applicationname* is the name of the application. The file will have all the localization files in its root folder.

Procedure

1. From the Application List view, select the application whose language packs you want to export.
2. Click **Language**.
3. Click  to display the menu options.
4. Click **Export**. All the available locales in the application are exported to the zip file. The file name is the name of the application, in the format *applicationname.locale*.zip.

Editing Phrases in a Locale

You can edit phrases for a locale directly in the application. You can either amend a single key or multiple keys, depending on your requirements. This is useful because it means you can edit phrases without having to export and re-import language packs.

Procedure

1. From the Application List view, select the application whose phrases you want to edit.
2. Click **Language**.
3. Click  to display the menu options.
4. Select the locale whose phrases you want to edit.
5. Click **VIEW PHRASES**.
6. Click the phrase of the key you want to edit and make your changes.
7. (Optional) You can add/edit a comment for the phrase. Click the comment to edit it.
8. Once you have made your changes, select **SAVE** to save your changes.
9. Click  to go back to the Language view.
   You can now test and publish your application, depending on your requirements. See Managing Applications.

Configuring a Fallback Chain for a Locale

You can configure a fallback chain for a locale for each of your applications. A fallback phrase is used if a specific key/value phrase in a given language is not available. In this case, a fallback chain is specified to show an available phrase from another language.

When configuring fallback chains, usually the fallback languages are based on similarities between languages. The last language in the chain falls back to en_US. For example, a fallback chain for the French language might be, fr_FR > fr_CH > fr_BE > en_US.

There can be any number of locales in the chain but it must not cause either:

- a circular dependency, for example, fr_FR > fr_CH > fr_BE > fr_FR, or
- a co-dependency, for example,
  - fr_FR > fr_CH
  - fr_CH > fr_FR

Procedure

1. From the Application List view, select the application whose fallback chain you want to edit.
2. Click **Language**.
3. Click  to display the menu options.
4. Select the locale for which you want to set the fallback locale.
5. Click **SET FALLBACK**.
6. Click the fallback locale that you want to use for the selected locale.

7. Once you have made your changes, click **SAVE** to save your changes.

8. Click  to go back to the Language view.
   You can now test and publish your application, depending on your requirements. See Managing Applications.

Deleting Locales

You can delete a locale from an application. However, you cannot delete the default locale, en_US, which is always the last locale in the fallback chain. Secondly, you cannot delete a locale that is referenced in fallback chain as this breaks the chain.

If you select a locale that is used in a fallback chain, the **Delete** button is not available. To remove a locale in a fallback chain, you must remove all the locales in the chain first. For example, to remove the FR locale in the following chain, fr_FR > fr_CA > fr_BE > fr_CH, you must remove the locales in the following order; fr_CH, fr_BE, fr_CA, fr_FR.

Procedure

1. From the Application List view, select the application whose locale you want to delete.
2. Click **Language**.
3. Click  to display the menu options.
4. Select the locale you want to delete. You can select as many locales as you require.
5. Click **Delete**.
   A **Delete selected locales from the app?** message is displayed.
6. Select **Yes** to delete the locale or **No** to return to Application Development.

Customizing the Workapp Application

The **Workapp** application demonstrates how to use typical BPM-related user interfaces and functions. You can customize the user interfaces and functions, depending on your requirements. For example, you may want to change the **Workapp** application to use your own company logo and theme.

In summary, to do this, clone the **Workapp** application, create a new application, and modify the custom templates.

Cloning the Workapp Application

You cannot customize the **Workapp** application, directly in Application Development. You must clone the **Workapp** application, create a new application and then customize your new application.

Procedure

- From Application Development, clone, the **Workapp** application.
  See Cloning Applications.

Localizing Your Customized Application

If you want to localize your new application, you must configure it to use the correct localization files. To do this, unregister **Workapp** and register your new application.

Procedure

1. Edit `MyWorkCtrl.js` to configure your new application to use it's translation files.
2. Uncomment the following code:
   ```
   // app.run(['BPMTranslationServiceConfig', function(BPMTranslationServiceConfig)
   {
   //     BPMTranslationServiceConfig.registerApp("MyApp_name_here", null);
   // }]);
   ```
3. Unregister the **Workapp** application. For example:
   ```
   BPMTranslationServiceConfig.unregisterApp(workapp,null);
   ```
4. Register your new application. For example:
   Change:
   ```
   BPMTranslationServiceConfig.registerApp(workapp,null);
   ```
   to:
   ```
   BPMTranslationServiceConfig.registerApp("applicationName", null);
   ```
   where *applicationName* is the name of your new application.

Customizing the Workapp Application Custom Templates

Change the custom templates as required. For example, if you want to add your company name to a Work View:

Procedure

1. Open your application.
2. Double-click the file `customTemplate\bpm_work_list\bpm_work_list.html` to open it.
3. Find the following code:
   ```
   <div layout="row" class="work-list-header" ng-show="!showDetails &amp;&amp; !
   showForm">
   ```

```
   <div flex="20" flex-gt-md="20" class="work-list-title">
      <md-icon md-svg-icon="grocery">
      </md-icon>
      <span translate="{{$root.appConfig.appId}}.BpmWorkList.work_list_title"
translate-values="{title: '{{$root.currentMenuTitle}}'}" translate-default="Work
Items">{{$root.currentMenuTitle}}</span>
   </div>
</div>
```

4. Add your company name. For example,

```
<div layout="row" class="work-list-header" ng-show="!showDetails &amp;&amp; !
showForm">
   <div flex="20" flex-gt-md="20" class="work-list-title">
      <md-icon md-svg-icon="grocery">
      </md-icon>
      <span translate="{{$root.appConfig.appId}}.BpmWorkList.work_list_title"
translate-values="{title: '{{$root.currentMenuTitle}}'}" translate-default="Work
Items">{{$root.currentMenuTitle}}</span>
   </div>
XYZ Insurance Company
</div>
```

5. Save and close `bpm_work_list.html`.
6. Publish your application to make it available to your users. See Publishing Applications.

Application Development System Actions

Access to functions in Application Development is controlled by system actions. System actions are actions that need to be authorized, or need to be restricted to users with a certain level of authority. These actions might include, for example, editing roles and applications, or publishing applications.

This authorization is implemented by associating system actions with privileges within the organization model that is built in TIBCO Business Studio. See *TIBCO Business Studio Modelling Guide* for more information. Some Application Development functions are controlled by a single system action and some are controlled by multiple system actions.

The table below describes the system actions that are required for Application Development and what each system action allows you to do.

System Action	Allows you to ...
ViewRoles	View the roles.
EditRoles	Create/edit/delete roles and add/remove/update mappings of roles to organization entities.  If you are creating, editing or deleting roles, you must also have the EditApplication system action.
EditLocales	Edit resource bundles for a given application or to delete a resource bundle from a given application for a given locale. You cannot edit the resource bundles for locales for applications marked with the System permission.  You must also have the EditApplication system action.
EditApplication	• Deploy a new application. • Revert a deployed application to an earlier version. • Export a deployed application. • Clone a deployed application. • Delete a deployed application. • Update permissions of an application. • Executing an application in RAD/JAD mode (see Working Offline With Applications)
PublishApplication	Publish a deployed application.

If you are developing applications that call any of the TIBCO Client Application Framework Object APIs or that access the services of any of the Application Development component services, then there are other system actions required. For example, if you are developing an application for users that uses the startBizAction operation from the BPMBusinessActionsService, a user would need to have the executeBusinessService system action. See the *TIBCO ActiveMatrix® BPM Client Application Developer's Guide* for more information.