TIBCO Spotfire® Deployment Kit for Apple iOS

Software Release 2.10
June 2018

Two-Second Advantage®

TIBCO®

Contents

Important information

TIBCO Spotfire Mobile 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

The following documents for this product can be found in the TIBCO Documentation Library:

- *TIBCO Spotfire® Deployment Kit for Apple iOS* (HTML version)
- *TIBCO Spotfire® Deployment Kit for Apple iOS* (PDF version)
- *TIBCO Spotfire® Deployment Kit for Apple iOS Release Notes*

Documentation for the app can be found at TIBCO Spotfire® for Apple iOS.

System Requirements for Spotfire Products

For information about the system requirements for Spotfire products, visit http://spotfi.re/sr.

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.

For quick access to TIBCO Spotfire content, see https://community.tibco.com/products/spotfire.

TIBCO Spotfire Deployment Kit for Apple iOS

The TIBCO Spotfire® Deployment Kit for Apple iOS consists of a set of app source code files that an enterprise developer can use to customize, build, and deploy a special version of Spotfire for Apple iOS . Your customized app can feature enterprise branding and an enterprise-controlled user experience. Use this guide to learn the process of customizing and building your own enterprise version of the app.

Deployment to users is usually achieved by distribution through an enterprise iTunes store, or in another means you might already have in place. App distribution is out of scope for this guide. You should talk to your systems administrators for more information about the procedure for distributing iOS apps in your enterprise.

Customized Spotfire for iOS app features and limitations

This deployment kit contains many features to provide a customized user experience. It does not provide support for certain advanced development aspects.

Use the deployment kit to customize the following resources:

- The app name (as it appears in the home screen and in **Settings** on the device).
- The app icon.
- The iTunes app icon (as it appears in the iTunes store).
- The app launch screen.
- Text displayed in the application, and text describing the app in place of the default settings values in the Settings app.
- The language of the text displayed in the application by providing a `Localizable.strings` file in the folder specifying the language.
- Any images displayed in the application replacing default images.
- The default settings for Spotfire renamed, removed, or made to be static and unchangeable by users.

Use the deployment kit to customize the user experience:

- Control the app behavior at start up:
  - Preconfigure server connections in the custom app.
  - Specify the length of time the launch screen is displayed.
  - Customize the login information for single-sign-on and assign a custom full or partial URL for a custom sign-on page across all connections.
  - Load a specific Spotfire analysis file from a Spotfire Server and display it in the Analysis view.
  - Browse a specific Spotfire library URL.
  - Provide a custom URL to an enterprise web page.
- Disable users' ability to add or edit Spotfire libraries.
- Disable users' ability to add or edit analyses in the Favorites view in the app.
- Hide the Favorites view and Recently viewed in the app.
- Display custom menu items with custom icons in the app navigation sidebar menu that open custom enterprise web pages.
- Provide customized Help, available from the app navigation sidebar menu.
- Hide the **Help** button from the app navigation sidebar menu.

- Display a custom set of analyses displayed in the Examples view in the app.
- Capture usage data using Google Analytics.

You cannot use the deployment kit to do the following:

- Write custom Objective C code to be executed when the app runs.
- Add new string or image resources used by the app. Only existing resources can be modified.
- Add new settings values to be used by the app. Only existing ones can be relabeled, localized, made read-only to the user, or hidden.
- Disable the Annotation view for adding and sharing annotations on an analysis.
- Disable or change the behavior of commands in the action menus displayed in the Analysis view toolbar in the app.

Prerequisites

Before you begin customizing the Spotfire Deployment Kit for Apple iOS app, review the following tools recommendations.

We tested and built Spotfire Deployment Kit for Apple iOS using the following tools. We recommend them for customizing and deploying your custom version of TIBCO Spotfire® for Apple iOS .

- An Apple computer running OS X 10.13 or higher.
- Xcode 9.4 or higher installed (available for free on the Apple® App Store).
- A graphics editor, such as Adobe Photoshop®, to edit image files and icons.
- An iPad® or iPhone® device running iOS 10 or higher to test on (optionally you can use the iOS Simulator that comes with Xcode for initial testing).
- An Apple Developer account, with a developer certificate or an enterprise developer certificate, which is required to install and test on an iOS device.

This certificate is not required for testing the app in the iOS Simulator.

Spotfire Deployment Kit for Apple iOS contents

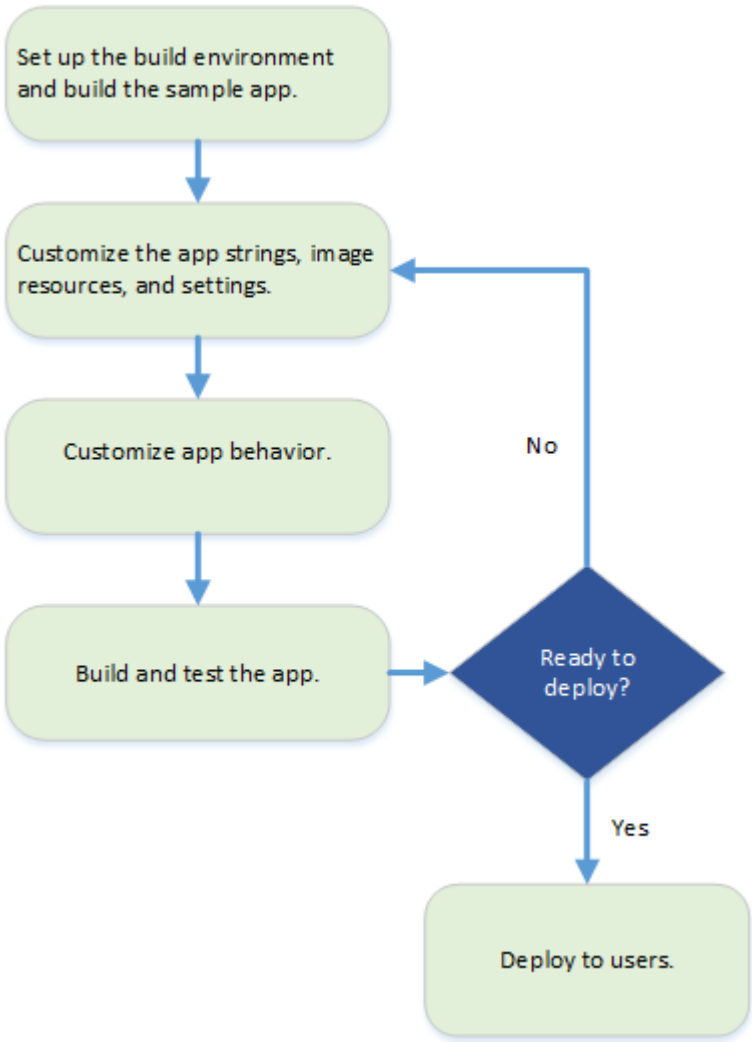
The Spotfire Deployment Kit for Apple iOS contains the components to develop a customized iOS app for Spotfire.

Component	Filename	Description
Spotfire mobile app framework file	`SpotfireFramework.embeddedframework`	The Spotfire mobile app framework file encapsulates the entire Spotfire for Apple iOS program, along with all the default resources and behaviors you see in the Spotfire version of the app.
Xcode project	`DeploymentKitApp.xcodeproj`	Using the Xcode project, change the provided string and image resources and the settings values. Modify values in the provided configuration file to implement a customized version of Spotfire for Apple iOS .

Customization workflow

Before you create a customized app using Spotfire Deployment Kit for Apple iOS, you should understand the workflow for taking the app from set up to deployment.

The following diagram illustrates the workflow for building a customized app.



Setting up the build environment

The first step to customize the TIBCO Spotfire® for Apple iOS app is to ensure you can build the sample app provided in the Deployment kit.
Perform this task in the copy of the DeploymentKitApp in Xcode on your computer.

Prerequisites

Download the `DeploymentKitApp` project file.

Procedure

1. In Xcode, open the project file `DeploymentKitApp`.
2. Click **Product** > **Build** (or press Command + B) to build the project with no changes.
   The project should build with no errors or warnings.

3. Set the active scheme to any device simulator for iOS 10 or higher, and then, in Xcode, click **Run**. The app should launch in the device simulator on your computer. The app contains all the features found in the Spotfire for Apple iOS app on the iTunes store.
4. Examine the app features in its default state before you customize it.
5. Click **Stop** to quit the app and return to Xcode.

Customize the app string, image resources, and settings

Begin the actual app customization by changing its most visible identifiers.

Before you customize the app, make a copy of the DeploymentKitApp and keep it in a safe place.

You can perform any of the following tasks to customize the app appearance, working in a copy of the DeploymentKitApp in Xcode on your computer.

Sign the app for distribution

If you are targeting iOS 10 devices or the simulator, you must sign the application with your team signing identity. You might also need to add the entitlement for Keychain sharing to your application in Xcode.

For applications running on iOS 10 to access the device or simulator keychain (which the app uses to store credentials for libraries you might connect to), they must be signed with a team distribution signing identity. If you use a development signing identity, such as your Xcode personal team identity (which is provided by Xcode), you must also enable the keychain sharing entitlement under **Capabilities** for the project in Xcode.

If you forget to change these settings in the project, when you run the app on an iOS 10 or higher device or in the simulator, the app is unable to save credentials for libraries in the keychain. The side effect of this problem is you that are prompted multiple times for username and password.

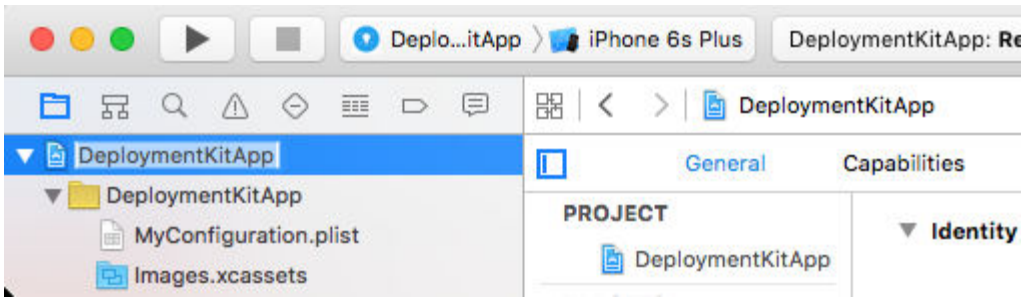
The following three tasks, in order, describe preparing the app for distribution.

Modifying the app identifier

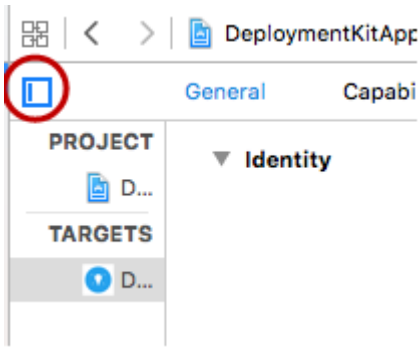
The app identifier (also known as a bundle identifier) is a combination of the reverse domain name of your company and the app name. Change this string to uniquely identify your version of the app from that of TIBCO or any other enterprise version.
Perform this task in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

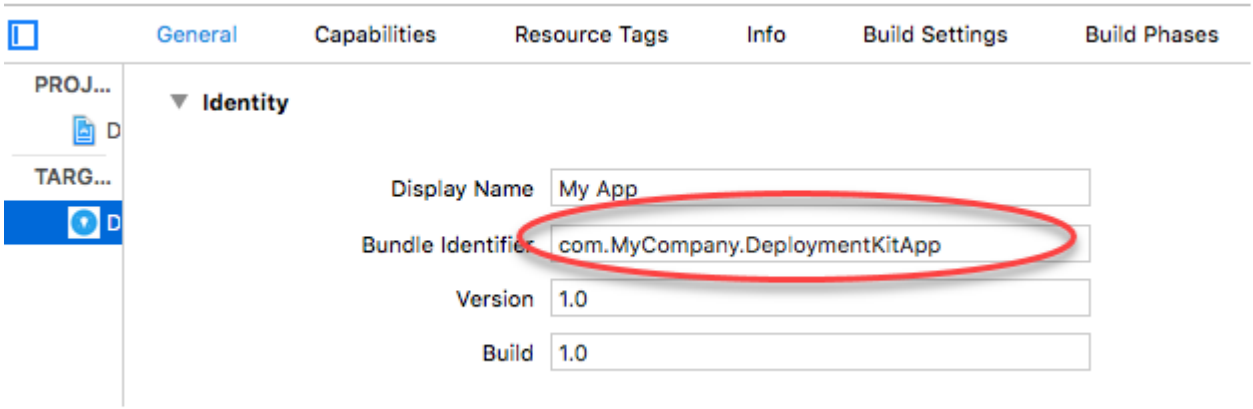
1. From the top of the Xcode Navigator, select the DeploymentKitApp.



2. Click the icon for showing project and targets list.



3. Click the existing name in the **Project and Targets** list, and then modify it.
4. Select the **General** tab of the project view, and then modify the editable portion of the **Bundle Identifier** text box, shown with your enterprise reverse domain name.



Setting team and developer account options

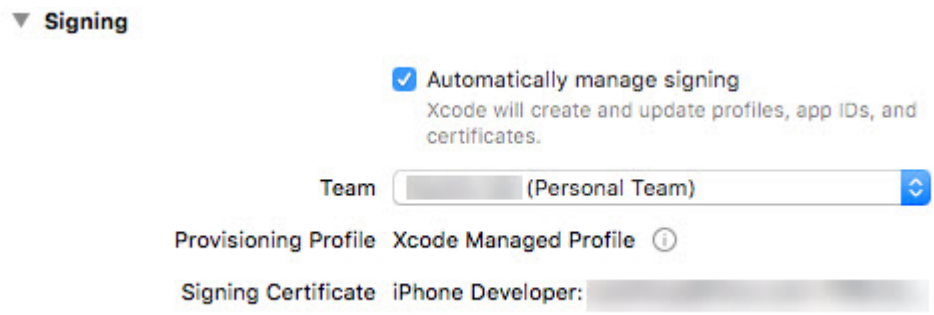
You can associate your Apple Developer account with Xcode and set the **Team** box in the **General** tab of the project settings to your Apple Developer Team account.
Perform this task in the copy of the DeploymentKitApp in Xcode on your computer. While you are customizing the TIBCO Spotfire® for Apple iOS app, set these account identifier options.

Prerequisites

To set these options, you must have the Xcode application and an Apple Developer account. See Prequisites for more information.

Procedure

1. Click **Xcode** > **Preferences** > **Accounts**.
2. In the resulting dialog, click the plus sign at the bottom of the **Apple IDs** list box.
3. In the resulting dialog, provide the team or personal account Apple ID and password.
4. Close **Preferences**, and then change the **Team** list box entry from `None` to the team associated with the Apple ID.



The option to have Xcode manage the signing automatically is selected by default.

Setting the capability for keychain sharing

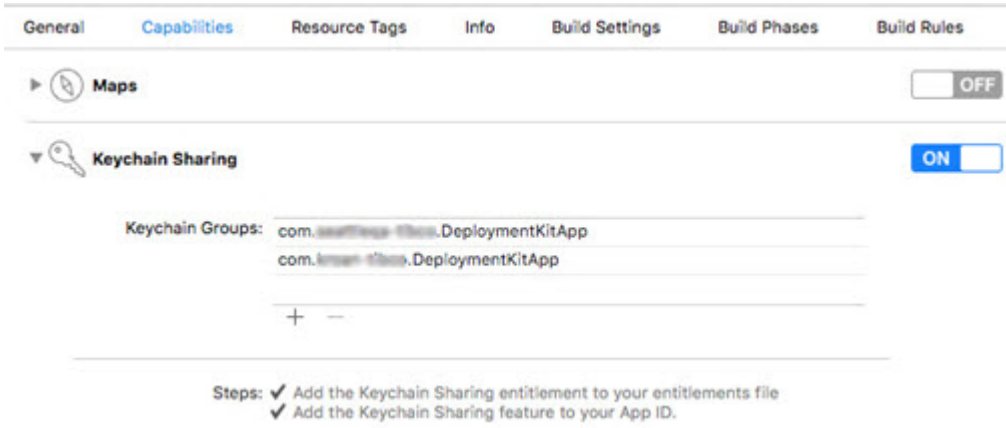
If you signed your app for distribution, and you are using a personal team identifier, you must set the capability for keychain sharing in Xcode.
Perform this task in the copy of the DeploymentKitApp in Xcode on your computer.

Prerequisites

You must have a personal team identifier. See Setting team and developer account options and Prequisites for more information.

Procedure

1. In Xcode, click the **Capabilities** tab.
2. Locate and enable the **Keychain Sharing** capability.
3. For **Keychain Group**, specify an appropriate keychain group identifier, such as the `com.`*`CompanyName`* portion of your product bundle identifier.



4. Click the **General** tab and check that you see no warning indicators below the **Team** entry.
   If you see warnings, follow the advice in those warnings to resolve further issues.

Modifying the app icon

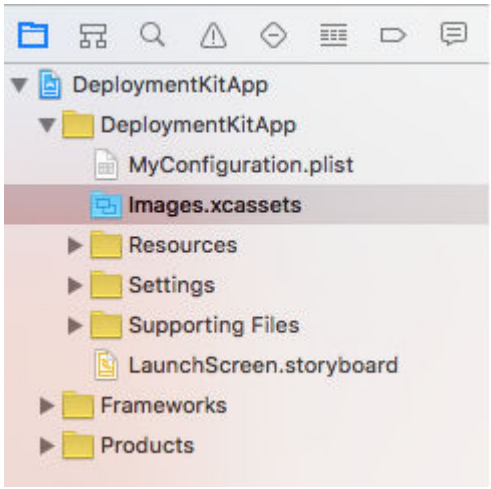
The Spotfire Deployment Kit for Apple iOS app uses image asset catalogs for the app icon. You can change the app icon that appears in the home screen on the iOS device.
Perform this task in the copy of the DeploymentKitApp in Xcode on your computer.

Prerequisites

You must have the images for the new app icon that are the same point size as the corresponding default images.

Procedure

1. In the Xcode Navigator, open `Images.xcassets`.



2. In the editor window, in the left pane list, click **AppIcon**.
The **AppIcon** pane appears, displaying the default icons.

3. Drag and drop the icon image files (`.png` format files) for the appropriate size matching that of the placeholder area for each 1x and 2x region.
The point size given is for both the width and height of the 1x icon image, so the 2x version is double the size listed.

   - 1x specifies non-retina display.
   - 2x specifies retina display.

Editing the launch screen storyboard

The Spotfire Deployment Kit for Apple iOS app uses a storyboard resource file for the startup (launch screen) image. You can change the launch screen that displays when the app opens. The `LaunchScreen.storyboard` is an interface builder storyboard file that uses auto-layout and some basic constraints on the controls to adjust the display for all the supported devices.

Perform this task in the copy of the DeploymentKitApp in Xcode on your computer. By default, the launch screen shows on the device for just a moment when the app launches.

To display an image on the launch screen using the LaunchScreen storyboard, you can create a new image set in the assets catalog named `Images.xcassets` included with the DeploymentKitApp project and drag and drop image files into the placeholders in this set.

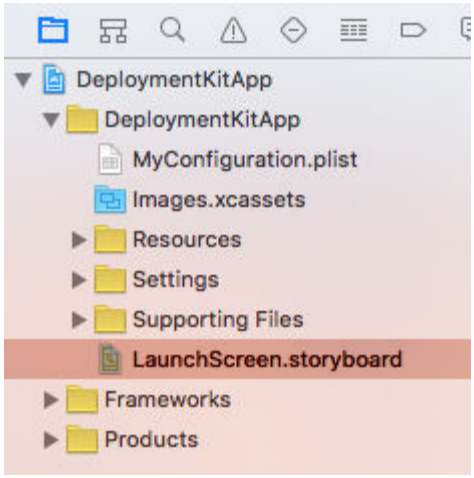
Optionally, you can just change the default text that already appears in the provided storyboard and drag and drop an image into the View Controller within this storyboard file.

Prerequisites

To edit the LaunchScreen storyboard file, you should have experience working with Xcode, editing storyboards, and using layout features such as Constraints and auto-layout. For more information about working with storyboard layouts, see Xcode help.

Procedure

1. In the Xcode Navigator, open the folder `LaunchScreen.storyboard`.



   The View Controller Scene is opened displaying the default text provided with the DeploymentKitApp. In addition to default text, the View includes Constraints for centering the title and placing the copyright information to the bottom of the screen.

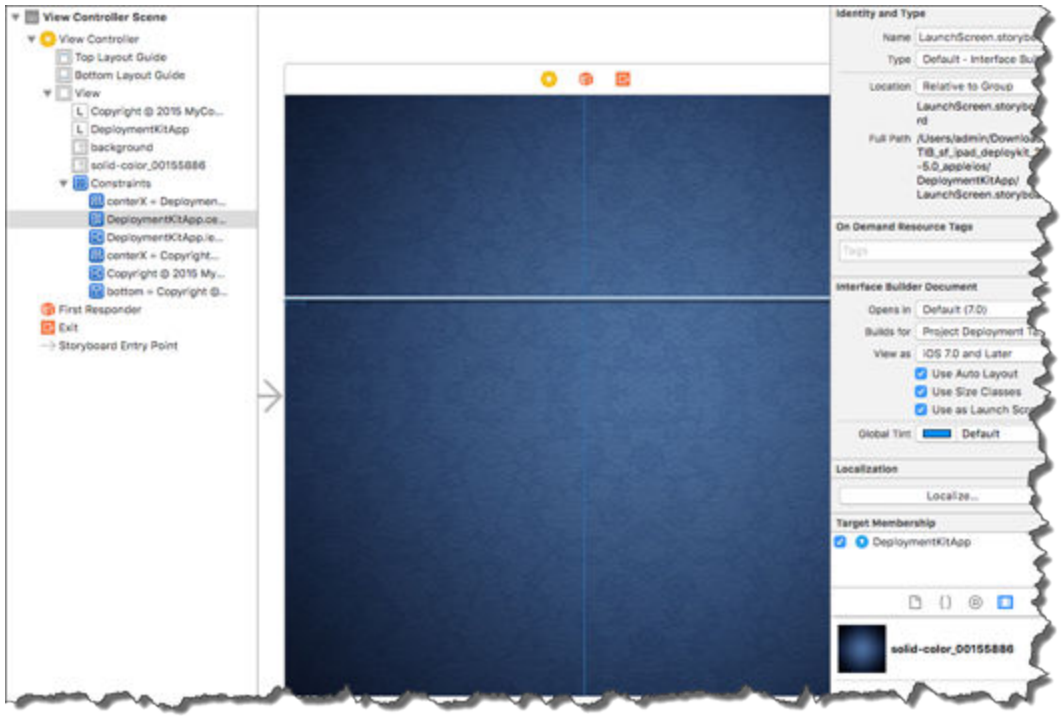
2. Remove or edit any of the default text.
3. Drag and drop an image file to the placeholder area in the editor window..

   Only static content, such as images, work in the launch screen. No interactive controls, such as buttons, work in this storyboard.

4. Optional: Adjust the constraints to control any text placement.

Result

Here is an example displaying a color for the launch screen.



What to do next

If you want to change the duration that the launch screen is displayed, see Setting the splash launch display duration.

Adding optional content to the Settings view footer area

If you want to provide an image or other content in the Settings view, you can provide the HTML file in the app bundle.
Perform this task in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

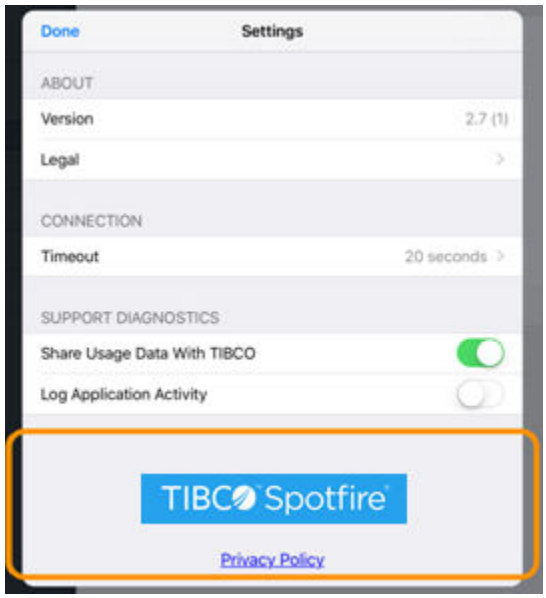
1. Create an HTML document containing any image, link, or text you want to display.
   The available content area is 180 points high and as wide as the Settings view displays, which depends on the device and orientation used.
2. Save the HTML document with the name `SettingsFooter.html`.
3. Place the file `SettingsFooter.html` in the project.

   Place `SettingsFooter.html` in the `Resources` folder, where other project resources are kept.

Result

When the project is built and tested, the contents are displayed in the footer of the Settings view.

The footer view remains hidden until the HTML file and all resources referenced from it have finished loading. Consider using only local resources.



Changing the app title

You can change the name of the app as it appears on the Home screen and in the Settings app.
Perform this task in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. In the Xcode Navigator, open the file `DeploymentKitApp/Supporting Files/DeploymentKitApp-Info.plist`.
2. In the editor window, locate the key labelled Bundle display name.

Key	Type	Value
Bundle display name	String	`My App`

In the Source Code view, this section appears as follows.

```
<key>CFBundleDisplayName</key>
    <string>My App</string>
```

3. Modify this value to change the displayed title for the app.

Key	Type	Value
Bundle display name	String	`My Fantastic App`

Modifying the app iTunes store picture

When your app appears in the iTunes store, the image that it uses is obtained from your app package with an image file called `iTunesArtwork`. You can customize this image file.
Perform this task in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. in the folder `DeploymentKitApp`, Expand the folder `Resources`.
2. Right-click one of the files `iTunesArtwork` or (for retina display), `iTunesArtwork@2x`, and then click **Show in Finder** to open a Finder window for these files.
   These files are just PNG format image files with the `.png` extension removed.
3. Either edit the existing files or replace them with another PNG file.

   Do not change the file name. Do not include a file extension.

   `iTunesArtwork` is a PNG format image file with the resolution 512 x 512 pixels. The file `iTunesArtwork@2x` has a resolution double that: 1024 x 1024. The artwork for the two files should be the same, and it is usually the same as your app icon.

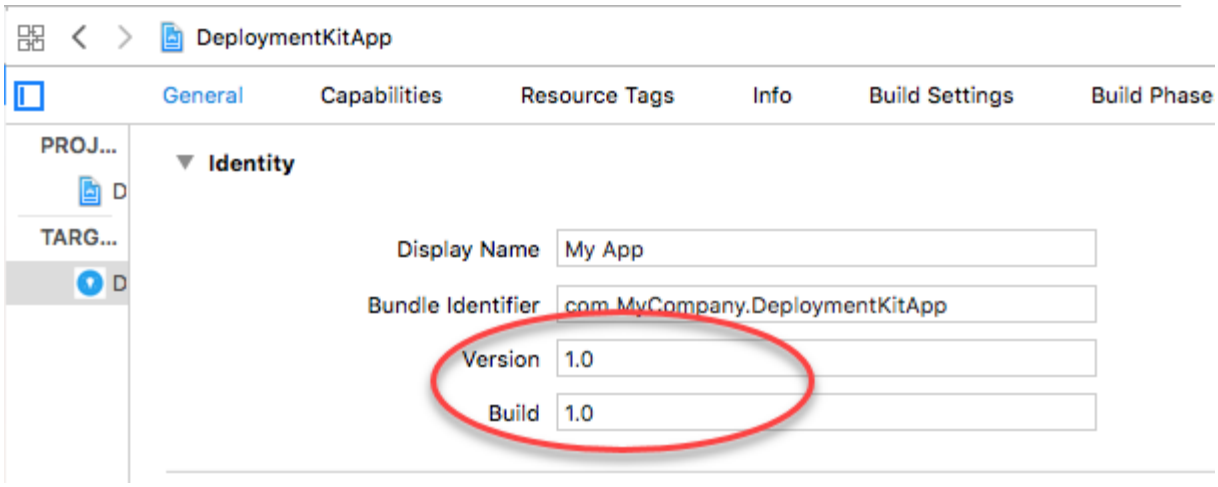
Modifying the app version number

By default, the version number of the app is set to 1.0 and the build number is also set to 1.0. You can change these numbers. You should increment the app version number with each release of your app. Incrementing the app version number with each new release can provide control over configuration. For example, if you provide a custom server connection in `MyConfiguration.plist`, the specified connection replaces any server connections from previous app versions. To make sure your custom server connection is updated correctly, you must update the app version number for each new version. For more information on custom server configuration, see Configurations for a custom login page for single sign on.

Perform this task in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. In the **Project and Targets** view, select the target DeploymentKitApp.
2. Click the **General** tab.
3. Under **Identity**, provide the **Version** number and **Build** number.



Result

The app version number, followed by the build number in parentheses, is displayed as the first entry in **Settings** for the app, as well as in the **Settings** sidebar menu item view in the app.

Modifying the app end user license agreement

The Settings view includes an end user license agreement (EULA). You can customize this EULA. Perform this task in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. In the project files, browse to the folder `Resources`.
2. Open the file `legal-en.txt` in your favorite text editor.
   This folder contains four sample text files for localization purposes. By default all text is in English. You can localize each of these files to is specified language.

`legal-de.txt`	Specifies German
`legal-es.txt`	Specifies Spanish
`legal-fr.txt`	Specifies French
`legal-en.txt`	Specifies English

   To provide legal text in another language, save the file with the convention `legal-`*`langcode`*`.txt`, where *`langcode`* specifies the two-letter international language code.
3. Replace the contents of the sample text with your license text.
4. Save the file.

Result

The edited license text appears in a simple scrolling text view in the app, displayed when the user taps the **Legal** item in the Settings view.

The app strings

The app strings contained in the file `Localizable.strings` are used by the app to display user interface elements and information to the user during an operation. You can have multiple copies of this file—one for each language you translate for—each in a folder specifying a target language.

Localized versions for Spanish, French, and German are included with the project. For information about localizing, see Localizing the app strings.

The strings contained in `Localizable.strings` constitute the set of all the strings you can customize and their default values, as defined in the SpotfireFramework.embeddedframework linked to this app. If you remove any string from this list, the app uses the default value defined in the `SpotfireFramework.embeddedframework`.

The strings are organized into sections that relate to views or functionality in the app. The comments describe under what circumstances the string is presented, and from which view it is visible.

A common workflow for customizing strings is to identify the strings to customize, and then remove the rest, leaving just the customized strings.

Before you modify the file, make a backup copy of the original file so you can identify other strings you might want to customize in the future.

See Modifying the app strings for more information.

If you add strings with keynames to this file, they are not used by the app, unless you change the customization configuration file `MyConfiguration.plist` to refer to them. See Customize app behavior for more information.

Modifying the app strings

You can modify, customize, and translate nearly all of the strings that are displayed for the user. Perform this task in the copy of the DeploymentKitApp in Xcode on your computer. See The app strings for more information.

Before you modify the file containing the strings, make a backup copy of the original file so you can identify other strings you might want to customize in the future.

Procedure

1. In the Xcode Navigator, expand the `Resources` folder.
2. Select the file `Localizable.strings`.
   The editor view changes to edit this file.
3. Customize the strings, following any special instructions in the comment immediately following the string.

   Do not modify the keyname of the string, which is specified by the quoted string on the left of the equal (=) sign on each line.

   Leave spaces on either side of the equal sign, and put a semi-colon at the end of the line.

String Format

```
"[keyname]" = "[value]";
```

Localizing the app strings

All strings displayed in the app are already localized to English, French, German, and Spanish. These localizations appear as child entries in the Xcode project hierarchy under `Resources/Localizable.strings`. If you need to add your own localization of the app strings, follow this procedure.
In the DeploymentKitApp project folder, the following project files appear in `Resources/Localizable.strings`.

- `en.lproj` for English.
- `es.lproj` for Spanish.

- `fr.lproj` for French.
- `de.lproj` for German.

Procedure

1. Duplicate the folder `en.lproj`.
2. Rename the duplicate folder *`langcode`*`.lproj`, where *`langcode`* is the two letter international language code for your language.
3. Localize the strings in the `Localizable.strings` file in its corresponding language folder.
4. In the Xcode project, in the Navigator, right-click the `Resources` folder to display the menu.
5. Click **Add files to project**, and then select the `Localizable.strings` file from your new language folder.

Result

The entry in the Xcode project hierarchy now contains a new entry under the item `Localizable.strings` with your new localization file.

What to do next

Localize the app settings.

Modifying the app images

Nearly all of the images used in the user interface of the app can be customized. Override the default image with one you create that has the same dimensions, file name, and extension as the default. The project folder `DeploymentKitApp` has a subfolder `DefaultImages` containing copies of the default images. See the contents of this folder using Finder.

This folder also contains a file `_README.txt` that explains the purpose of each image file and where it is used in the app.

Procedure

1. Identify which image files you need to customize by using the reference in `DefaultImages/_README.txt`.
2. Copy the needed files to a new folder.

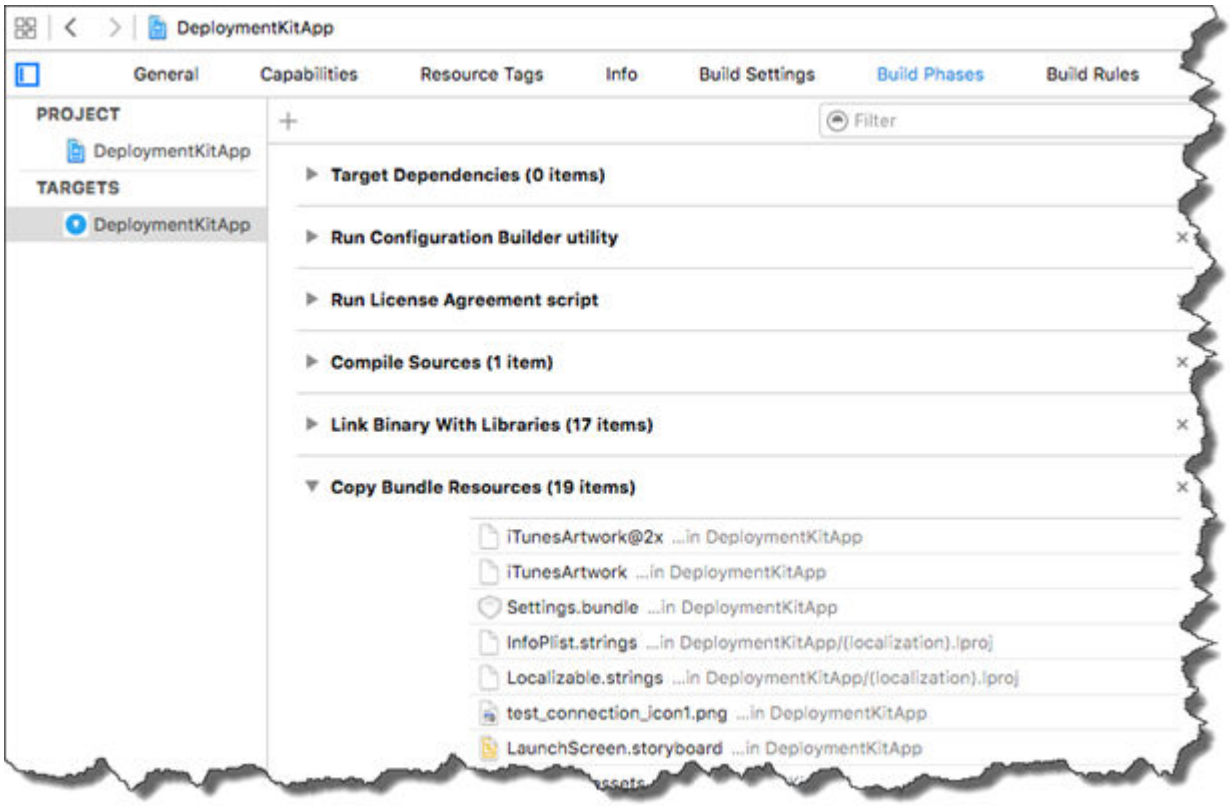
   You must retain the same file names as the originals.
3. Customize these image files using the graphic editor of your choice.

   You must retain the same resolution (that is, the height and width in pixels), pixel density, and file format as the originals.
4. Add the customized image files.

   You can do this either by right-clicking the `Resources` folder to display the context menu, and then clicking **Add files to DeploymentKitApp**, or by dragging and dropping the files from Finder onto the `Resources` folder.
5. Ensure your images are at the top of the compiled resources list.

   This step is required so the app uses your customized images rather than the defaults.

   a) In the Xcode Navigator, select the `DeploymentKitApp` project.
   b) In the editor window, under **Project** > **Targets**, select **DeploymentKitApp** (or the target name for the app if you changed it) from the projects and targets list.
   c) Click the **Build Phases** tab, and then expand **Copy Bundle Resources**.



d) In the list of resources, make sure the images you added to the project are at the top, listed above all others.

These resources are specified by "… in DeploymentKitApp" or the name of your target app, after the name. You might have to find them in the list, select them, then drag them to the top of the list.

The app settings

The DeploymentKitApp comes with a Settings bundle that is a duplicate of the default Settings bundle in the Spotfire for Apple iOS app.

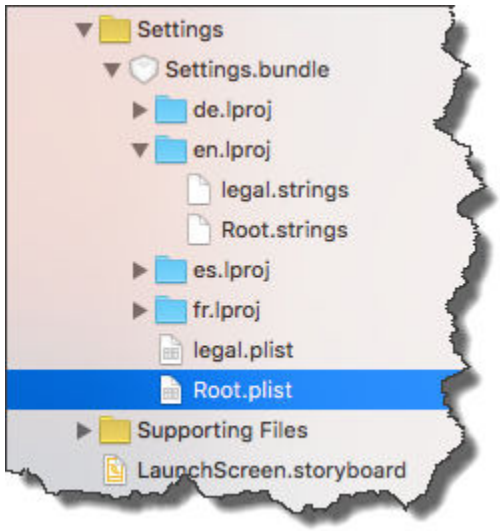
Changing the app settings should be done only with a clear understanding of XCode and app behavior. Changing settings without understanding XCode can result in unexpected behavior.

Allowed	Restricted
You can customize the app behavior using the file `MyConfiguration`. See Customize app behavior for more information.	You cannot define settings the app uses because the collection of app settings is defined and controlled by the app code.
You can add new settings that you refer to in the customization of the app behavior.	You cannot change the app code.
You can modify the titles, default values, and possible values of the settings as they appear in the Settings app.	Do not change the setting control default type, except for changing the title to read-only.
You can make settings read-only for users of the app.	You cannot add new settings or values.

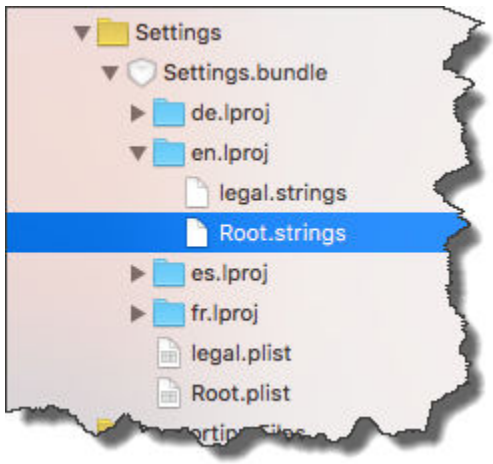
Allowed	Restricted
You can remove settings.	

Much like string and image resources, settings have unique key names that you can use in the app to override the default values of the same setting key name defined in `SpotfireFramework.embeddedframework`.

The app project hierarchy contains the folder `Settings`. This folder contains localized settings bundles. In a settings bundle, use the file `Root.plist` to change the title, the default value, and possible choices for an application setting. You can also use `Root.plist` to make a setting read-only so users cannot change it.



The settings bundle also contains the file `Root.strings`, which you can use to customize the settings titles for values in settings that appear in the iOS Settings app.



Changing the titles of values in Settings

The file `Root.plist` specifies the settings for the app; however, you can override the values in the localized settings bundle file `Root.strings`.
For this task, work in the Xcode Navigator.

Procedure

1. Open the file `DeploymentKitApp/Settings/Settings.bundle/langcode.lproj/Root.strings` (where *langcode* is the two-letter language specifier for the folder).
2. In the file `Root.strings`, locate the setting value to change.
3. Set the key to the existing title, and set the value to the new title.
   Customized value titles must follow the format `"[existing title]" = "[newtitle]";`

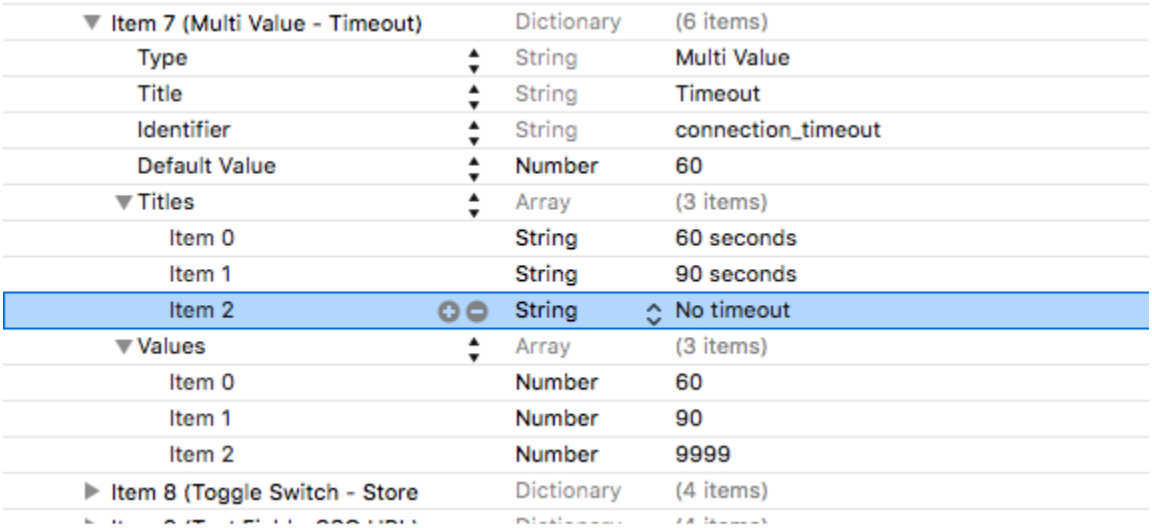
   

   Leave spaces on either side of the equal sign, and put a semi-colon at the end of the line.
4. Rebuild and redeploy the app to the device to test. The Settings app displays the new value setting.

Example: changed title in Settings

In the following example, suppose you want to change the title defined in the file `Root.plist` in the `Titles` array from `"No Timeout"` to `"Takes forever"`.

`Root.plist` contains the following setting:



To override the definition, open the file `Root.strings`, and then add the following line:

```
"No Timeout" = "Takes forever";
```

When the app is rebuilt and redeployed on the device, the Settings app displays the title **Takes forever** as the last choice in the **Timeout** setting.

Making settings read-only

You can make a setting visible but read-only, so users cannot change it.
Perform this task in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. In the **Navigator**, open the file `DeploymentKitApp/Settings/Settings.bundle/Root.plist`.
2. In the Property List view, find the setting to change to read only and expand its view to see its properties.

3. For the setting's `Type` property, change the string value to `Title`.

   If you edit this file in the Source Code view, change the value to `PSTitleValueSpecifier`.

4. Save and close the file.

Result

The user can see the setting but not change it.

Example: Title is read-only

By default, the `Title - Version` property in the file `Root.plist` is set to read only, because its Type key is set to `Title`.

Key	Type	Value
Item 1 (Title - Version)	Dictionary	(4 items)
Type	String	`Title`
Default Value	String	
Title	String	`Version`
Identifier	String	`app_version`

In the Source Code view, this section appears as follows.

```
<dict>
   <key>Type</key>
   <string>PSTitleValueSpecifier</string>
   <key>Title</key>
   <string>About</string>
```

Remove settings

You can remove app settings without causing problems with the app.

Removing any setting in the app causes the app to automatically use the default value of the setting as defined in `SpotfireFramework.embeddedframework`. The removed setting no longer appears in the iOS Settings app for the DeploymentKitApp.

Localizing the app settings

All settings displayed in the app are already localized to English, French, German, and Spanish. These localizations appear as child entries in the DeploymentKitApp Xcode project hierarchy under `Settings/Settings.bundle`, with folder names consisting of a two character international language code followed by the extension `lproj`.

- `en.lproj` for English.
- `es.lproj` for Spanish.
- `fr.lproj` for French.
- `de.lproj` for German.

Procedure

1. In the DeploymentKitApp, right-click the bundle named `Settings.bundle` and click **Show in Finder**.
2. In Finder, right click this bundle, and from the menu click **Show Package Contents**.
3. Select the the folder `en.lproj` and copy it to your Desktop.
4. Rename this folder copy to *`langcode`*`.lproj`, where *`langcode`* is the two letter international language code for your language.
5. In the language folder, in the file `Root.strings`, localize the strings, following the same procedure as outlined in Changing the titles of values in settings.
6. Copy the localized folder.
7. Repeat Step 1 and Step 2.
8. Paste the localized `lproj` folder to the package `Settings.bundle`.
9. In Xcode, in the Navigator, and then open the `Settings` folder.

Result

The `Settings.bundle` entry in the Xcode project hierarchy now contains a new folder with your new localization of Settings for the app.

Customizing the list of settings that appear in the app Settings view

You can change the list of Settings items that appear in the app Settings view by modifying a list of settings key names
Perform this task in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. In the Xcode Navigator, open `Resources/Localizable.strings`. Select the English version of `Localizable.strings`.
2. In the editor, locate the string `"SFSettingsListViewController_Settings_DisplayInAppKeys"`.
   This is a resource consisting of the names of settings keys, which are delimited by semi-colons and are used to control the settings displayed in the Settings view in the app. By default, we provide the following settings keys.
   - `app_version`
   - `app_legal`
   - `connection_timeout`
   - `do_logging`
   - `email_support`
3. Add or delete settings keys from this list.

   Be careful not to add spaces between entries, and be sure you preserve the semi-colons between items.

Example: remove the settings options logging and e-mail support

Original String Resource	`"SFSettingsListViewController_Settings_DisplayInAppKeys" = "app_version;app_legal;connection_timeout;do_logging;email_support";`
Modified String Resource	`"SFSettingsListViewController_Settings_DisplayInAppKeys" = "app_version;app_legal;connection_timeout";`

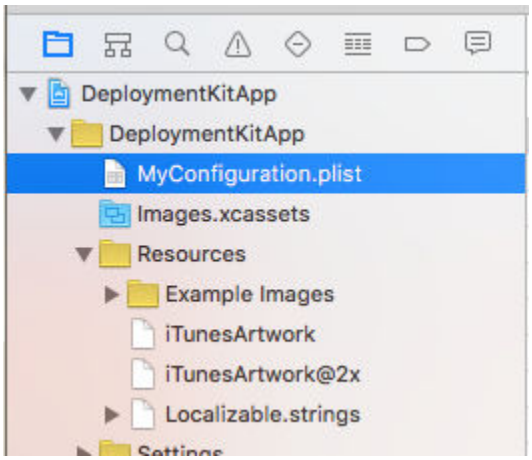
The entries `do_logging` and `email_support` were removed from the list to accomplish this task.

You can find the settings key names used in this string in the file `Settings/Settings.bundle/Root.plist`.

Customize app behavior

In addition to customizing strings, images, and settings, you can modify certain app behaviors to change the user experience.

The configuration file that controls app behavior is `MyConfiguration.plist`. It is located in the DeploymentKitApp project hierarchy, as displayed in the Xcode Navigator.



You can control the following behavior using this file.

- At startup:
  - Authenticate through a Spotfire Business Author login page rather than the login service.
  - Preconfigure a Spotfire Business Author server connection. See Configurations for a custom login page for single sign on for more information.
  - Open a specific analysis document in Analysis view.
  - Browse a Spotfire library at a specified URL.
  - Open a web page at a specified URL.
- Disable adding or editing Spotfire libraries.
- Disable adding and removing analyses in the Favorites view.
- Add custom menu items (with text and custom icons) below the **Recently Viewed** item in the sidebar menu. This is useful for displaying custom web pages to present a selection of analyses to load.

- Hide the Favorites view.
- Hide the Examples view.
- Hide the Recently viewed view.
- Specify a URL to download a list of analyses document URLs on a Spotfire server for display in the Examples view.

MyConfiguration.plist file

The file `MyConfiguration.plist`, provided in the DeploymentKitApp project, is a standard Apple XML `plist` file containing keynames and their allowable values. `MyConfiguration.plist` also refers to example string resources, settings, and images, which are contained in the DeploymentKitApp project. You can customize the`MyConfiguration.plist` with values that are appropriate to your customized app.

You can customize most settings in `MyConfiguration.plist` by editing key and value pairs in the Property List view in XCode. Optionally, you can edit the XML directly in the Source Code view. To work in the XML view, right click the file name, and then click **Open As** > **Source Code**.

Key	Type	Value
▼ Root	Dictionary	(21 items)
AppStartupAction	String	custom: https://docs.tibco.com/products/
DisableAddEditLibraries	Boolean	NO
DisableAddEditFavorites	Boolean	NO
HideFavorites	Boolean	NO
HideExamples	Boolean	NO
HideRecentlyViewed	Boolean	NO
HideCustomViewCloseButton	Boolean	NO
DisableLoginPrompting	Boolean	NO
SplashScreenDuration	Number	0
HideAnalysisInfo	Boolean	NO
PopulateExamplesFromUrl	String	
GoogleAnalyticsID	String	
GoogleTagManagerContainerID	String	
HideHelp	Boolean	NO
HelpUrl	String	http://spotfi.re/iOS-users-guide
CustomViewCloseAlertTitle	String	Custom View Close Alert Title
CustomViewCloseAlertMessage	String	Custom View Close Alert Message
AnalysisViewCloseAlertTitle	String	Analysis Close Alert Title
AnalysisViewCloseAlertMessage	String	Analysis Close Alert Message
▶ CustomMenuItems	Array	(2 items)
▶ CustomConnections	Array	(2 items)

Key	Type	Possible Values

`AppStartupAction`	String	`analysis: url`
		`browse: url`
		`custom: url`

Key	Type	Possible Values
DisableAddEditLibraries	Boolean	YES, NO (default NO)
DisableAddEditFavorites	Boolean	YES, NO (default NO)
HideFavorites	Boolean	YES, NO (default NO)
HideExamples	Boolean	YES, NO (default NO)
HideRecentlyViewed	Boolean	YES, NO (default NO)
HideCustomViewCloseButton	Boolean	YES, NO (default NO)
DisableLoginPrompting	Boolean	YES, NO (default NO)
SplashScreenDuration	Number	Any number of seconds or fraction of seconds. (default 0)
HideAnalyisisInfo	Boolean	YES, NO (default NO)
PopulateExamplesFromURL	String	*url*
GoogleAnalyticsID	String	Set to your unique Google Analytics Identifier assigned to your Google Analytics account for your app. (The default is empty. If left empty, no tracking is done in the app you create.)
GoogleTagManagerContainerID	String	Set to your unique Google Tag Manager Container ID for your app from your Google Tag Manager account. (The default is empty. If left empty, no tracking is done in the app you create.)
HideHelp	Boolean	YES, NO (default NO)
HelpURL	String	Set to your customized version of the browser-based app help. By default, this link opens the app help provided on docs.tibco.com.
CustomViewCloseAlertTitle	String	Add a title to the close alert message that is displayed when a user closes a custom web view. If you provide a message, the title is optional.
CustomViewCloseAlertMessage	String	Add a close alert message that is displayed when a user closes a custom web view.
AnalysisViewCloseAlertTitle	String	Add a title to the close alert message that is displayed when a user closes the Analysis view. If you provide a message, the title is optional.

Key	Type	Possible Values
`AnalysisViewCloseAlertMessage`	String	Add a close alert message that is displayed when a user closes the Analysis view.
`CustomMenuItems`	Array of Dictionaries	See CustomMenuItems
`CustomConnections`	Array of Dictionaries	See CustomConnections

CustomMenuItems

The example `MyConfiguration.plist` contains an array for `CustomMenuItems`. Each dictionary entry in the `CustomMenuItems` array can have a set of key-value pairs.

Key	Type	Possible Values
icon	String	*`image file name + ext`* (Specifies the menu item is not currently selected.)
iconSelected	String	*`image file name + ext`* (Specifies the menu item is currently selected.)
title	String	*`title`*
url	String	*`url`*

For more information about adding custom menu items, see Adding custom menu items to the sidebar.

CustomConnections

The example `MyConfiguration.plist` contains an array for `CustomConnections`. Each dictionary entry in the `CustomConnections` array can have a set of key-value pairs.

Key	Type	Possible Values
icon	String	*`image file name + ext`* (Specifies the connection is not currently selected.)
iconSelected	String	*`image file name + ext`* (Specifies the connection is currently selected.)
title	String	*`Connection 1`*
url	String	*`http://www.myserver1.example.com`*
ssourl	String	*/myssologinpageurl*
loginSupport	String	*`Message shown in the dialog asking the user to authenticate.`* (This message can contain markdown links. See Configurations for a custom login page for more information.)

For more information about setting the `CustomConnections` options, see

- Configurations for a custom login page
- Adding preconfigured server connections to the sidebar

for more information.

Defining keywords for dynamic configuration values

For all string type values in the `MyConfiguration.plist` configuration file, you can specify special keywords in the value string to cause the app to find the value in string resources or in settings. This task demonstrates specifying the title and icon for a custom menu item in the app's sidebar menu. Perform this task in the copy of the DeploymentKitApp in Xcode on your computer.

You can customize most settings in `MyConfiguration.plist` by editing key and value pairs in the Property List view in XCode. Optionally, you can edit the XML directly in the Source Code view. To work in the XML view, right click the file name, and then click **Open As** > **Source Code**.

Procedure

1. Add your custom string resource entry to the appropriate `Localization.strings` file.
   See The app strings for more information.
2. Add your image file to the `Resources` folder in the DeploymentKitApp project.
   See Modifying the app images for more information.
3. In the file `MyConfiguration.plist`, specify the `string:` and `resource:` keywords to indicate that those configuration values should be loaded from resources. See the example.

Likewise, you can specify the keyword `setting:`, as shown in the configuration example, and then specify the key name of an app setting value to use instead. Then the app looks up the setting value and substitutes it in place of the keyword when it evaluates the configuration value.

Example: defining a custom menu item

The following example demonstrates deining a custom menu item.

1. In the app file `Localizable.strings`, we defined these string resources.
```
"my_menuitem_icon1_filename" = "test_menuitem_icon1.png";
"my_menuitem_icon1selected_filename" = "test_menuitem_icon1_active.png";
```
2. In the file `MyConfiguration.plist`, in the Property List view, find and expand the key labelled CustomMenuItems.
3. Add the following entries.

Key	Type	Value
CustomMenuItems	Array	(2 items)
Item 0	Dictionary	(4 items)
icon	String	`string: my_menuitem_icon1_filename`
iconSelected	String	`string: my_menuitem_icon1selected_filename`
title	String	`setting: my_test_menuitem1_title`
url	String	`http://spotfire.tibco.com`

In the Source Code view, this section appears as follows.

```
<dict>
   <key>icon</key>
   <string>string: my_menuitem_icon1_filename</string>
   <key>iconSelected</key>
   <string>string: my_menuitem_icon1selected_filename</string>
   <key>title</key>
   <string>setting: my_test_menuitem1_title</string>
   <key>url</key>
   <string>http://spotfire.tibco.com</string>
</dict>
```

4. In the Xcode project navigator, find and open the file `DeploymentKitApp/Settings/Settings.bundle/Root.plist`.
5. In the Property List view, find the setting Text Field - My Custom Menu Item Title.

Key	Type	Value
Item 19 (Text Field - My Custom Menu Item Title)	Dictionary	(4 items)
Default Value	String	`My Custom Menu Item 1`
Identifier	String	`my_test_menuitem1_title`
Title	String	`My Custom Menu Item Title`
Type	String	`Text Field`

In the Source Code view, this section appears as follows.

```
<dict>
   <key>DefaultValue</key>
   <string>My Custom Menu Item 1</string>
   <key>Key</key>
   <string>my_test_menuitem1_title</string>
   <key>Title</key>
   <string>My Custom Menu Item Title</string>
   <key>Type</key>
   <string>PSTextFieldSpecifier</string>
</dict>
```

Security considerations

Remember that when you build the app into a package for distribution with Xcode, string resources for the app are in clear text.

The Settings bundle file `Root.plist` and the file `Localizable.strings`, where string resources for the app are stored, are placed into the app package in clear text form.

These files have no encryption of any kind.

- Never store sensitive information such as user names, passwords, credential or session tokens, or Spotfire Business Author URLs for your enterprise in these files.
- Do not refer to them there using the `string:`, `resource:` or `setting:` keywords found in the file `MyConfiguration.plist`.

It is safe to store Spotfire Business Author URLs for your enterprise in the file `MyConfiguration.plist` because this file is not distributed in your compiled app package. Instead, this file is processed during the app build process, and the code produced is compiled into the binary app in your app package.

Do not store user names, passwords, or credential or session tokens in `MyConfiguration.plist`.

Configurations for a custom login page

If your enterprise uses a custom login page or other server that implements single sign on, or if it redirects users to authenticate, you can specify a full or partial url that points to that authentication server and displays a popover web page view in the app with the custom login page for users.

The Spotfire for Apple iOS app provides two facilities for providing single sign on authentication.

- You can use the Setting in the app settings called `SSO URL` (with the `Settings.bundle` key `sso_url`). You can set this option to either the full or partial string representing the URL for your custom login page, or to the single sign on authentication page. This setting supports regular expressions: you can use special syntax to match redirection URL addresses when they are encountered in the app. Use this setting to specify a URL applicable globally across all Spotfire Server connections in the app.
- You can preconfigure Spotfire Server connections in the `MyConfiguration.plist` file. You can specify a full or partial single sign on URL as part of these connections using the key `ssourl`. See CustomConnections for more information.
- You can customize the look of the connection in the list. Add two images to `MyConfiguration.plist` in the `CustomConnection` section : one for the unselected state for the key `icon`, and one for the selected state for the key `iconSelected`. (Default images are provided in the DeploymentKitApp.)
- You can provide advice to the user by adding a string to the `loginSupport` key in the `MyConfiguration.plist` file. This string can include either a mail or a website markdown link. See Setp 4 of Adding preconfigured server connections to the sidebar for more information.

If a single sign on URL is specified in a preconfigured connection, it takes precedence for this connection over the global app setting `SSO URL`.

Setting start-up action

You can customize the start-up action for your app.
Perform this task by editing the file `MyConfiguration.plist` in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. In the file `MyConfiguration.plist`, in the Property List view, locate the key labelled AppStartupAction.

Key	Type	Value
AppStartupAction	String	`custom: https://MyCustomWebPageURL`

2. Edit the entry, providing the specific key and value pairs for the start-up action you want to occur.

- To load a specific Spotfire analysis when the user starts the app, add key and value pair similar to the following example.

Key	Type	Value
AppStartupAction	String	`analysis:https://`*`MySpotfireWebplayerServer`*`/spotfire/wp/OpenAnalysis?file=<`*`analysis id`*`>`

In the Source Code view, this section appears as follows.

```
<key>AppStartupAction</key>
<string>analysis:https://MySpotfireWebplayerServer/spotfire/wp/OpenAnalysis?
file=<analysis id>
```

- To start the app and allow the user to browse a specific Spotfire library, add a key and value pair similar to the following example.

Key	Type	Value
AppStartupAction	String	`browse: https://MySpotfireWebPlayerServer/spotfire`

Ensure that the user's credentials are valid to log in to the specified server, because the user might be prompted to provide credentials if the server is authenticated.

- To start the app and load a web page in a web viewer, add a key and value pair similar to the following example.

Key	Type	Value
AppStartupAction	String	`custom: https://docs.tibco.com/products/tibco-spotfire-deployment-kit-for-apple-ios`

Specifying authentication through Spotfire Business Author

The file `MyConfiguration.plist` contains settings for specifying whether to use the login service. You can disable the login service for users who are logged in through a Spotfire Business Author.
If you are using a server that does not have a login service configured, or if you want users authenticated through a Spotfire Business Author login, set this option. Perform this task by editing the file `MyConfiguration.plist` in the copy of the DeploymentKitApp in Xcode on your computer.

An error reporting a user authentication failure is the most likely symptom indicating that this option should be set.

You can customize most settings in `MyConfiguration.plist` by editing key and value pairs in the Property List view in XCode. Optionally, you can edit the XML directly in the Source Code view. To work in the XML view, right click the file name, and then click **Open As** > **Source Code**.

Procedure

1. In the file `MyConfiguration.plist`, in the Property List view, find the key labelled DisableLoginPrompting.

Key	Type	Value
DisableLoginPrompting	Boolean	`NO`

2. Change the value from `NO` to `YES`.

In the Source Code view, this section appears as follows.

```
<key>DisableLoginPrompting</key>
    <true/>
```

Key	Type	Value
DisableLoginPrompting	Boolean	`YES`

Result

Users are not prompted for login from the app; rather they can now log in through a Spotfire Business Author authentication service.

Adding preconfigured server connections to the sidebar

You can preconfigure connections to a Spotfire Server Web Player service. You can add the connections as menu items in the sidebar below the existing default Favorites view, Examples view, and Recently viewed items.
Perform this task by editing the file `MyConfiguration.plist` in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. In the file `MyConfiguration.plist`, in the Property List view, find and expand the key labelled CustomConnections.

   The following example shows two connections.

Key	Type	Value
CustomConnections	Array	(2 items)
Item 0	Dictionary	(6 items)
icon	String	`test_connection_icon1.png`
iconSelected	String	`test_connection_icon1active.png`
title	String	`Connection 1`
url	String	`http://www.myserver1.example.com`
ssourl	String	`/myssologinpageurl`
loginSupport	String	`Forgot your credentials? [Contact your administrator](http://www.myserver1.example.com/support)`
Item 1	Dictionary	(3 items)
title	String	`Connection 2`
url	String	`http://www.myserver2.example.com`
ssourl	String	`/myssologin`

These entries specify the start of an array of custom server connections.

Within the array, add the custom connection keys and values, as shown in this example. Each connection is a Dictionary (within `<dict>` and `</dict>` markers), and each key and value is delimited by marker pairs for `<key>` and `<string>`. You can add more of these blocks, one for each of the preconfigured server connections you want to add to the sidebar.

In the Source Code view, this section appears as follows.

```
<key>CustomConnections</key>
   <array>
     <dict>
       <key>icon</key>
         <string>test_connection_icon1.png</string>
       <key>iconSelected</key>
         <string>test_connection_icon1active.png</string>
       <key>title</key>
         <string>Connection 1</string>
       <key>url</key>
         <string>http://www.myserver1.example.com</string>
       <key>ssourl</key>
         <string>/myssologinpageurl</string>
       <key>loginSupport</key>
         <string>Forgot your credentials? [Contact your administrator](http://
www.myserver1.example.com/support)</string>
     </dict>
     <dict>
       <key>title</key>
         <string>Connection 2</string>
       <key>url</key>
         <string>http://www.myserver2.example.com</string>
       <key>ssourl</key>
         <string>/myssologin</string>
```

```
    </dict>
  </array>
```

2. Specify the title, the URL for the Web Player service connection, and if necessary, the full or partial URL for the single sign on login page to redirect users to if they are not authenticated for this connection.

Key	Type	Value
title	String	`Connection 1`
url	String	`http://www.myserver1.example.com`
ssourl	String	`/myssologinpageurl`

If you specify a value for the key `ssourl` in a pre-configured server connection, that value overrides the app-wide setting in the app Settings for key `sso_url`. See Configurations for a custom login page for single sign on for more information.

3. Optionally, specify the filenames of images to indicate to the user whether the device is connected to the server.

Key	Type	Value
icon	String	`test_connection_icon1.png` indicates the connection is not currently selected.
iconSelected	String	`test_connection_icon1active.png` indicates that the connection is currently selected.

Sample images are provided in the `Resources/Example Images` folder of the DeploymentKitApp. If no other images are provided, the app uses the default sample images.

These image filenames can be localized in the `Localizable.strings` resource to provide different images for users in different locales.

4. Optionally, specify a string for `loginSupport` to be displayed to users who provide incorrect credentials.

Key	Type	Value
loginSupport	String	`Forgot your credentials? [Contact your administrator] (http://www.myserver1.example.com/support)`

If you do not provide a customized string, the following default string is used:

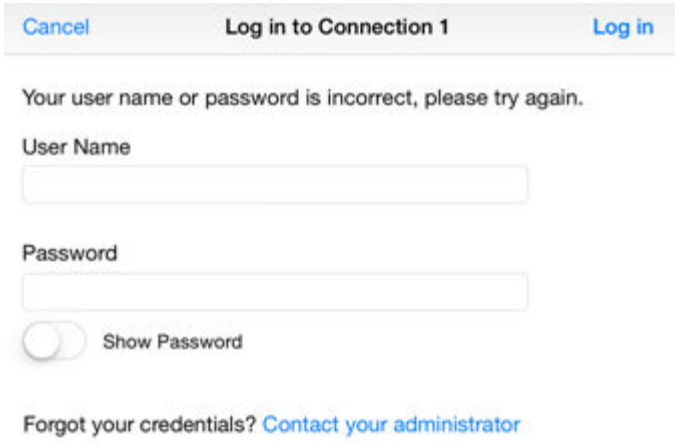
```
Forgot your credentials? Contact IT for help
```

The custom string can include an `http:` or a `mailto:` link. Place the string that you want to appear as the link in square brackets `[ ]`, followed by the link in parentheses.

Examples

```
Forgot your credentials? Contact your administrator.
Forgot your credentials? [Contact your administrator](mailto:admin@example.com)
Forgot your credentials? [Contact your administrator](http://www.example.com/
support)
```

The link examples display in the login page as follows, and the link opens the web browser or the mail application, specifying the target in the key string.



What to do next

To make sure that users do not accidentally delete a preconfigured custom server configuration, set the configuration option `DisableAddEditLibraries` to `YES`. See Preventing users from adding or editing libraries for more information.

Setting the launch screen display duration

You can change the duration, in seconds or fractions of seconds, that the app launch screen is displayed. Perform this task by editing the file `MyConfiguration.plist` in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. In the file `MyConfiguration.plist`, in the Property List view, locate the key labelled SplashScreenDuration.

Key	Type	Value
SplashScreenDuration	Number	0

In the Source Code view, this section appears as follows.

```
<key>SplashScreenDuration</key>
<integer>0</integer>
```

The default of `0` displays the launch screen for a fraction of a second.

2. Change the value from `0` to a number of seconds or fractions of seconds.

By default, the data type is set to `<integer>`. If you change the value to a fraction in the Source Code view, change the data type to `<real>`. (If you set the value to a fraction in the property list view, the data type is changed automatically.)

Result

The app launch screen displays for the specified duration.

What to do next

If you want to set the launch screen to a different image, see Editing the launch screen storyboard.

Disable user actions and views

You can disable the appearance of several views in the app, and you can control whether the user can add or edit libraries and Favorites view entries.

Consider the outcome for disabling actions and views. For example, if you choose to hide both the Favorites view, Examples view, or the Recently viewed view in the app, then by default, the app starts with an empty view. If you set these configuration options to `YES`, you should specify an AppStartupAction so that the user can interact with a working view at app start-up time.

You can customize most settings in `MyConfiguration.plist` by editing key and value pairs in the Property List view in XCode.

Preventing users from adding or editing libraries

You can configure the app so that the sidebar menu options **Add** and **Edit** for libraries are disabled.

Perform this task by editing the file `MyConfiguration.plist` in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. In the file `MyConfiguration.plist`, in the Property List view, find the key labelled DisableAddEditLibraries.

Key	Type	Value
DisableAddEditLibraries	Boolean	`NO`

You must set the DisableAddEditLibraries key to `true` when you have preconfigured for Spotfire Business Author or Spotfire Cloud connections in `MyConfiguration.plist`. Doing so prevents users from editing or deleting these connections in the app. See CustomConnections for more information.

2. Change the value from `NO` to `YES`.

In the Source Code view, this section appears as follows.

```
<key>DisableAddEditLibraries</key>
<true/>
```

Key	Type	Value
DisableAddEditLibraries	Boolean	`YES`

Result

The app sidebar menu options **Add** and **Edit** for libraries are not present if this key is set to `<true/>`. Setting the value to `<false/>` is equivalent to not having this key in the configuration file.

Preventing users from editing Favorites

You can prevent a user from adding to, rearranging, or removing analyses from the Favorites view.

Perform this task by editing the file `MyConfiguration.plist` in the copy of the DeploymentKitApp in Xcode on your computer.

You can customize most settings in `MyConfiguration.plist` by editing key and value pairs in the Property List view in XCode. Optionally, you can edit the XML directly in the Source Code view. To work in the XML view, right click the file name, and then click **Open As** > **Source Code**.

Procedure

1. In the file `MyConfiguration.plist`, in the Property List view, locate the key labelled DisableAddEditFavorites.

Key	Type	Value
DisableAddEditFavorites	Boolean	`NO`

2. Change the value from `NO` to `YES`.

In the Source Code view, this section appears as follows.

```
<key>DisableAddEditFavorites</key>
<true/>
```

Key	Type	Value
DisableAddEditFavorites	Boolean	`YES`

Result

The features that allow users to add, rearrange, or remove analyses from the Favorites view of the app are not present if this key is set to `YES`. Setting the value to `NO` is equivalent to not having this key in the configuration file.

Hiding the Favorites view

You can configure the app to hide the Favorites view.
Perform this task by editing the file `MyConfiguration.plist` in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. In the file `MyConfiguration.plist`, in the Property List view, find the key labelled HideFavorites.

Key	Type	Value
HideFavorites	Boolean	`NO`

2. Change the value from `NO` to `YES`.

In the Source Code view, this section appears as follows.

```
<key>HideFavorites</key>
<true/>
```

Key	Type	Value
HideFavorites	Boolean	YES

Result

The sidebar menu item to open the Favorites view and the default Favorites view presentation are not present if this key is set to YES. Setting the value to NO is equivalent to not having this key in the configuration file.

Hiding the Examples view

You can configure the app to hide the Examples view.
Perform this task by editing the file MyConfiguration.plist in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. In the file MyConfiguration.plist, in the Property List view, find the key labelled HideExamples.

Key	Type	Value
HideExamples	Boolean	NO

2. Change the value from NO to YES.

In the Source Code view, this section appears as follows.

```
<key>HideExamples</key>
<true/>
```

Key	Type	Value
HideExamples	Boolean	YES

Result

The sidebar menu item to open the Examples view and the default Examples view presentation are not present if this key is set to YES. Setting the value to NO is equivalent to not having this key in the configuration file.

Hiding the app help

You can configure the app to hide the **Help** button.
Perform this task by editing the file MyConfiguration.plist in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. In the file MyConfiguration.plist, in the Property List view, find the key labelled HideHelp.

Key	Type	Value
HideHelp	Boolean	NO

2. Change the value from NO to YES.

In the Source Code view, this section appears as follows.

```
<key>HideHelp</key>
<true/>
```

Key	Type	Value
HideHelp	Boolean	YES

Result

The sidebar menu item to open the app Help URL is not present if this key is set to YES.

Hiding the Recently-Viewed view

You can configure the app to hide the Recently viewed view.
Perform this task by editing the file `MyConfiguration.plist` in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. In the file `MyConfiguration.plist`, in the Property List view, find the key labelled HideRecentlyViewed.

Key	Type	Value
HideRecentlyViewed	Boolean	NO

2. Change the value from NO to YES.

In the Source Code view, this section appears as follows.

```
<key>HideRecentlyViewed</key>
<true/>
```

Key	Type	Value
HideRecentlyViewed	Boolean	YES

Result

The sidebar menu item to open the Recently viewed view is not present if this key is set to `<true/>`.
Setting the value to `<false/>` is equivalent to not having this key in the configuration file.

Hiding the Info button in the Analysis view toolbar

You can configure the app to hide the **Info** button in the Analysis view, which prevents the display of the analysis metadata, including the analysis URL.
Perform this task by editing the file `MyConfiguration.plist` in the copy of the DeploymentKitApp in Xcode on your computer. Set this option to ensure the user cannot view or change the analysis metadata, including the analysis URL.

Procedure

1. In the file `MyConfiguration.plist`, in the Property List view, locate the key labelled HideAnalysisInfo.

Key	Type	Value
HideAnalysisInfo	Boolean	`NO`

2. Change the value from `NO` to `YES`.

In the Source Code view, this section appears as follows.

```
<key>HideAnalysisInfo</key>
<true/>
```

Key	Type	Value
HideAnalysisInfo	Boolean	`YES`

Result

The **Info** button is not present if this key is set to `YES`. Setting the value to `NO` is equivalent to not having this key in the configuration file.

Hiding the Close button in the custom web view

You can configure the app to hide the **Close** button in a custom web view.
Perform this task by editing the file `MyConfiguration.plist` in the copy of the DeploymentKitApp in Xcode on your computer. Set this option to ensure the user cannot close a custom view and display only a blank screen.

Procedure

1. In the file `MyConfiguration.plist`, in the Property List view, locate the key labelled HideCustomViewCloseButton.

Key	Type	Value
HideCustomViewCloseButton	Boolean	`NO`

2. Change the value from `NO` to `YES`.

In the Source Code view, this section appears as follows.

```
<key>HideCustomViewCloseButton</key>
<true/>
```

Key	Type	Value
HideCustomViewCloseButton	Boolean	YES

Result

The **Close** button is not present if this key is set to YES. Setting the value to NO is equivalent to not having this key in the configuration file.

Providing custom app help

You can provide custom help content for the customized Spotfire app. By default, the URL is set to open content provided on docs.tibco.com for the Spotfire app.
Perform this task by editing the file MyConfiguration.plist in the copy of the DeploymentKitApp in Xcode on your computer.

Prerequisites

You must have a URL for the custom help content that you want to display when the app user taps the **Help** link in the sidebar.

You can hide the help by setting the value for HideHelp to True. See Hiding the app help for more information.

Procedure

1. In the file MyConfiguration.plist, in the Property List view, find the key labelled HelpURL.

Key	Type	Value
HelpURL	String	http://spotfi.re/iOS-users-guide-2-8

2. Edit the entry, providing the URL to the custom help.

Key	Type	Value
HelpURL	String	http://myCustomHelp/configure

In the Source Code view, this section appears as follows.

```
<key>HelpURL</key>
<string>http://myCustomHelp/configure</string>
```

Result

When the user taps **Help** in the sidebar, a browser opens and displays the page specified by the URL you provide.

Adding custom menu items to the sidebar

You can add custom menu items in the sidebar below the existing default Favorites view, Examples view, and Recently viewed items.

Perform this task by editing the file MyConfiguration.plist in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. In the file `MyConfiguration.plist`, in the Property List view, find and expand the key labelled CustomMenuItems.

   The following example shows two custom menu items.

Key	Type	Value
CustomMenuItems	Array	(2 items)
Item 0	Dictionary	(4 items)
icon	String	`string: my_menuitem_icon1_filename`
iconSelected	String	`string: my_menuitem_icon1selected_filename`
title	String	`setting: my_test_menuitem1_title`
url	String	`http://spotfire.tibco.com`
Item 1	Dictionary	(4 items)
icon	String	`test_menuitem_icon2.png`
iconSelected	String	`test_menuitem_icon2_active.png`
title	String	`My Test Menuitem 2`
url	String	`http://www.google.com`

   In the Source Code view, this section appears as follows.

```
<dict>
   <key>icon</key>
   <string>string: my_menuitem_icon1_filename</string>
   <key>iconSelected</key>
   <string>string: my_menuitem_icon1selected_filename</string>
   <key>title</key>
   <string>setting: my_test_menuitem1_title</string>
   <key>url</key>
   <string>http://spotfire.tibco.com</string>
</dict>
<dict>
   <key>icon</key>
   <string>test_menuitem_icon2.png</string>
   <key>iconSelected</key>
   <string>test_menuitem_icon2_active.pn</string>
   <key>title</key>
   <string>My Test Menuitem 2</string>
   <key>url</key>
   <string>http://www.google.com</string>
</dict>
```

   These entries specify the start of an array of custom menu items.

2. For each array, add custom menu item keys and values, as shown in the above sample.

   Each Dictionary block defines a custom menu item. You can add more of these blocks, one for each of the menu items you want to add to the sidebar, under the array.

Result

Your custom menu items display a custom icon and title, and when they are selected by the user, they can load the specified URL, as shown in the example.

Adding an alert for closing the Analysis view

You can add an alert to display when a user closes the Analysis view in the app.
Perform this task by editing the file `MyConfiguration.plist` in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. In the file `MyConfiguration.plist`, in the Property List view, locate the key labelled AnalysisViewCloseAlertMessage.

Key	Type	Value
AnalysisViewCloseAlertMessage	String	`Analysis Close Alert Message`

2. Edit the entry, providing the alert message to display.

   The alert is displayed only if the message setting contains one or more characters for the string value.

   In the Source Code view, this section appears as follows.

```
<key>AnalysisViewCloseAlertMessage</key>
<string>Analysis Close Alert Message</string>
```

Key	Type	Value
AnalysisViewCloseAlertMessage	String	*My new Analysis close message.*

3. Optional: Locate the key labelled AnalysisViewCloseAlertTitle.

Key	Type	Value
AnalysisViewCloseAlertTitle	String	`Analysis Close Alert Title`

4. Optional: Edit the entry, providing the title for the alert message.

   In the Source Code view, this section appears as follows.

```
<key>AnalysisViewCloseAlertTitle</key>
<string>Analysis Close Alert Title</string>
```

Key	Type	Value
AnalysisViewCloseAlertTitle	String	*My new Analysis close title.*

Result

The customized alert is displayed when the user closes the Analysis view.

Adding an alert for closing a custom view

You can add an alert to display when a user closes a custom view in the app.
Perform this task by editing the file `MyConfiguration.plist` in the copy of the DeploymentKitApp in Xcode on your computer.

Procedure

1. In the file `MyConfiguration.plist`, in the Property List view, locate the key labelled CustomViewCloseAlertMessage.

Key	Type	Value
CustomViewCloseAlertMessage	String	`Custom View Close Alert Message`

2. Edit the entry, providing the alert message to display.

   The alert is displayed only if the message setting contains one or more characters for the string value.

   In the Source Code view, this section appears as follows.

```
<key>CustomViewCloseAlertMessage</key>
<string>Custom View Close Alert Message</string>
```

Key	Type	Value
CustomViewCloseAlertMessage	String	*My new custom view close message.*

3. Optional: Locate the key labelled CustomViewCloseAlertTitle.

Key	Type	Value
CustomViewCloseAlertTitle	String	`Custom View Close Alert Title`

4. Optional: Edit the entry, providing the title for the alert message.

   In the Source Code view, this section appears as follows.

```
<key>CustomViewCloseAlertTitle</key>
<string>Custom View Close Alert Title</string>
```

Key	Type	Value
CustomViewCloseAlertTitle	String	*My new custom view close title.*

Result

The customized alert is displayed when the user closes the custom web view that is used for custom startup action and for customized menu items.

Configuring application usage tracking

You can control whether application usage data is tracked and how it is shared by setting the options found in Settings.bundle file `root.plist` and the DeploymentKitApp file `MyConfiguration.plist`. Perform this task in the copy of the DeploymentKitApp in Xcode on your computer.

Prerequisites

To configure the app to track usage, you must have a Google Analytics Identifier and a Google Tag Manager Container ID.

Procedure

1. In the Xcode project navigator, find and open the file `DeploymentKitApp/Settings/Settings.bundle/Root.plist`.

   You can customize most settings in `Root.plist` by editing key and value pairs in the Property List view in XCode.

   a) In the Property List view, find the setting `Share Usage Data with Developers`.

Key	Type	Value
Item 13 (Toggle Switch - Share Usage Data with Developers)	Dictionary	(4 items)
Type	String	`Toggle Switch`
Title	String	`Share Usage Data with Developers`
Identifier	String	`do_tracking`
Default Value	Boolean	`No`

   In the Source Code view, this section appears as follows.

```
<dict>
   <key>Type</key>
   <string>PSToggleSwitchSpecifier</string>
   <key>Title</key>
   <string>Share Usage Data With Developers</string>
   <key>Key</key>
   <string>do_tracking</string>
   <key>DefaultValue</key>
   <false/>
```

   b) Set the value from the default (`NO`) to `YES` (or in Source Code view, set it from `<false/>` to `<true/>`.

2. Open the file `MyConfiguration.plist`.
   a) Find the setting `GoogleAnalyticsID` and assign to it the string value for the Google Analytics Identifier for your app.

b) Find the setting `GoogleTagManagerContainerID` and assign to it the Google Tag Manager Container ID for your app.

The app uses Google Tag Manager to manage event tags for your app. See Google Analytics tags for more information.

3. Open a browser and log in to your Google Tag Manager account.
   a) Define the tags, rules, and macros for tracking within your app.

   A special macro, `gaProperty`, is defined in your Google Tag Manager account. This macro links tag events to your Google Analytics account. See the tag definition tables for appEvent, appTiming, and openScreen for more information about setting up your Google Tag Manager account.

You must configure this macro in Google Tag Manager as a data layer variable. The `GoogleAnalyticsID` you assigned in step 2 is sent automatically with each event to the Google Tag Manager.

Google Analytics tags

To support built-in usage tracking using Google Analytics, the app you customize with TIBCO Spotfire® Deployment Kit for Apple iOS must have tags, rules, and macros defined in your Google Tag Manager account for the app.

The reference topics provide information about the tags, rules, and macros established for tracking app usage.

Tags

Tag Name	Description
`appEvent`	Event tracking.
`appTiming`	Timing tracking.
`openScreen`	Screen name and app version.

Google Analytics tag: appEvent

To track your app events in Google Analytics, define the `appEvent` tag in your Google Tag Manager account. This tag requires that you define its properties as well as the macros and rules it relies on.

Tag Definition

Property	Value
Tag Name	`appEvent`
Tag Type	`Universal Analytics`
Tracking ID	`{{gaProperty}}`
Track Type	`Event`

Macros

Each of the following event tracking parameters has a macro defined for it, with the specified names.

Event tracking parameter	Macro name	Macro type	Data layer variable name
Event Category	{{appEventCategory}}	Data layer variable	{{appEventCategory}}
Event Action	{{appEventAction}}	Data layer variable	{{appEventAction}}
Event Label	{{appEventLabel}}	Data layer variable	{{appEventLabel}}
Event Value	{{appEventValue}}	Data layer variable	{{appEventValue}}
Application Version	{{appVersion}}	Data layer variable	{{appVersion}}
Tracking ID	{{gaProperty}}	Data layer variable	{{gaProperty}}

Rules

The following rule is associated with the tag `appEvent`.

Rule Name	Description
appEventRule	{{event}} equals appEvent

Google Analytics tag: appTiming

To track your app timing in Google Analytics, define the `appTiming` tag in your Google Tag Manager account. This tag requires that you define its properties as well as the macros and rules it relies on.

Tag Definition

Property	Value
Tag Name	appTiming
Tag Type	Universal Analytics
Tracking ID	{{gaProperty}}
Track Type	Timing

Macros

Each of the following timing tracking parameters has a macro defined for it, with the specified names.

Event tracking parameter	Macro name	Macro type	Data layer variable name
Timing Category	{{appTimingCategory}}	Data layer variable	{{appTimingCategory}}
Timing Variable	{{appTimingVar}}	Data layer variable	{{appTimingVar}}
Timing Value	{{appTimingValue}}	Data layer variable	{{appTimingValue}}

Event tracking parameter	Macro name	Macro type	Data layer variable name
Timing Label	{{appTimingLabel}}	Data layer variable	{{appTimingLabel}}
Application Version	{{appVersion}}	Data layer variable	{{appVersion}}
Tracking ID	{{gaProperty}}	Data layer variable	{{gaProperty}}

Rules

The following rule is associated with the tag appTiming.

Rule Name	Description
appTimingRule	{{event}} equals appTiming

Google Analytics tag: openScreen

To track screen access in Google Analytics, define the openScreen tag in your Google Tag Manager account. This tag requires that you define its properties as well as the macros and rules it relies on.

Tag Definition

Property	Value
Tag Name	openScreen
Tag Type	Universal Analytics
Tracking ID	{{gaProperty}}
Track Type	App view

Macros

Each of the following timing tracking parameters has a macro defined for it, with the specified names.

Event tracking parameter	Macro name	Macro type	Data layer variable name
Screen Name	{{screenName}}	Data layer variable	{{appTimingCategory}}
Timing Variable	{{appTimingVar}}	Data layer variable	{{appTimingVar}}
Timing Value	{{appTimingValue}}	Data layer variable	{{appTimingValue}}
Timing Label	{{appTimingLabel}}	Data layer variable	{{appTimingLabel}}
Application Version	{{appVersion}}	Data layer variable	{{appVersion}}
Tracking ID	{{gaProperty}}	Data layer variable	{{gaProperty}}

Rules

The following rule is associated with the tag `openScreen`.

Rule Name	Description
`openScreenRule`	`{{event}}` equals `openScreen`

Specify Examples view analyses

You can use the value for the key PopulateExamplesFromURL in the configuration file `MyConfiguration.plist` to specify a URL to connect to and then download a list of analyses to populate the Examples view when the app starts.

The list of analyses that you use to populate the Examples view when the app starts is provided by a service. You must implement the service, which delivers a JSON array of key-value pairs, similarly to the following example format.

```
[
{"Description":"Arkoma Well Analysis",
"ServerUrl":"http:\/\/spotfire-ondemand.tibco.com\/spotfire",
"ThumbnailUrl":"http:\/\/spotfire-ondemand.tibco.com\/spotfire\/
GetPreviewImage.ashx?analysisId=ffa2c946-23e5-47b7-870f-0b39f122ecc4",
"Title":"ArkomaWellAnalysis",
"Url":"http:\/\/spotfire-ondemand.tibco.com\/spotfire\/ViewAnalysis.aspx?file=\/
Production\/Examples\/ArkomaWellAnalysis"
}
…
]
```

The app connects to the specified URL that provides this JSON array using the user's credentials. The app prompts for the provided credentials when requesting this content, and then supplies the credentials to the server. This design allows content to be generated based on user identity, if necessary.

- See JSON Example Keys for descriptions of required or optional entries in the JSON array.
- See JSON Security Considerations for more information about using JSON.

JSON example keys

If you use JSON to populate the app Examples view from a URL, you need to know the optional and required keys used by JSON.

Keyname	Required	Description
`Description`	Yes	The description of the analysis that appears below the thumbnail (`ThumbnailUrl`) in the Examples view.
`ServerUrl`	Yes	The URL of the Spotfire Business Author or Spotfire Cloud on which the analysis is stored.
`ThumbnailUrl`	No	The URL of the thumbnail image file used to present a preview of the analysis in the Examples view. See Modifying the app Images for information about customizing the image to use. If not specified, the default blue background with the Spotfire logo in grey is displayed in the Examples view for this analysis.

Keyname	Required	Description
`Title`	Yes	The title of the analysis.
`Url`	Yes	The URL of the analysis, as specified when loading from a Spotfire library.

JSON security considerations

Because files are not encrypted when they are added to your distributed app package, you should take security precautions.

Do not store the URL to the server that provides the JSON array for this configuration option in the app file `Localizable.strings` or the Settings bundle and use the keywords `string:` or `resource:` or `setting:` to load it from `MyConfiguration.plist`.

These files are not encrypted when added to your distributed app package and are in clear text form.

Instead, you can safely store this URL directly in the file `MyConfiguration.plist`. The file `MyConfiguration.plist` is processed during the app build, and its contents are converted to code in the app binary file.

Index

J

K

L

O

P

R

S

T

U

V

W

X