



TEMENOS
The Banking Software Company

Hybrid Guide



Information in this document is subject to change without notice.

No part of this document may be reproduced or transmitted in any form or by any means, for any purpose, without the express written permission of TEMENOS HEADQUARTERS SA.

© 2010 Temenos Headquarters SA - all rights reserved.

Table of Contents

Document History	4
About this Guide	5
Online Help	5
Licensing and Technical Support	5
Hybrid Solutions	6
Build Process	6
Anatomy of a Hybrid Application	6
<i>Server Application</i>	6
<i>Shell Application</i>	6
<i>HTML Resources</i>	6
<i>Offline Pages</i>	6
<i>Error Phases</i>	6
Terminology	7
Build	8
Hybrid Licence	8
Add a Hybrid Project	8
Build a Hybrid Project	9
<i>Offline Pages</i>	9
<i>Error Phases</i>	10
<i>Presentation</i>	10
<i>Multi-language solutions</i>	10
<i>Language Selection Rule</i>	11
Native Device Features	12
<i>Using Cordova Hybrid Widgets</i>	13
<i>DEVICE_INFO</i>	14
<i>Creating your own Cordova Hybrid Widget</i>	15
<i>Cordova Plugins provided with edgeConnect</i>	15
<i>Using Third party Cordova plugins</i>	17
<i>Debugging a 3rd party plugin in EdgeConnect Hybrid</i>	20
<i>Progress Bar</i>	20
<i>Testing Native Device Features</i>	23
Cordova Push Notifications	25
<i>To Receive Push Notifications</i>	25
<i>Push Notification Message Flow</i>	26
Deploy	29
Deployment Set-up	29
Jenkins	29
<i>Starting Jenkins</i>	29
<i>Starting Android Slave</i>	30
<i>Starting iOS Slave</i>	31
<i>Starting Windows Phone 8 Slave</i>	31
<i>Starting Windows 8 Slave</i>	32
Hybrid Server Connection Details in edgeConnect Deployer	33
Add Deployment	33

Deployment Configuration	33
Profiles	34
<i>Common Settings tab</i>	35
<i>Certificates tab</i>	35
<i>Android Specific tab</i>	36
<i>iOS Specific tab</i>	36
<i>Windows Phone 8 Specific tab</i>	37
<i>Windows Specific tab</i>	37
Messages	38
Versioning	38
Deployment Wizard	38
<i>Deployment Results</i>	41
<i>Deployment Error Messages</i>	41
Publishing Hybrid Solutions	43
Android	43
<i>Publisher Account</i>	43
<i>Icons</i>	43
<i>Splashscreen</i>	43
<i>Assets needed for publishing</i>	43
<i>Signing</i>	44
<i>Deploying</i>	44
<i>Publishing</i>	45
iOS	45
<i>Publisher Account</i>	45
<i>Icons</i>	45
<i>Assets needed for publishing</i>	46
<i>Deploying</i>	46
<i>Publishing</i>	46
<i>Upload Binary</i>	47
Windows Phone 8	48
<i>Publisher Account</i>	48
<i>Icons</i>	48
<i>Splashscreen</i>	48
<i>Tile Images</i>	48
<i>Assets needed for publishing</i>	49
<i>Signing</i>	49
<i>Deploying</i>	49
<i>Publishing</i>	49
Windows	50
<i>Publisher Account</i>	50
<i>Customisable Resources</i>	50
<i>Assets needed for publishing</i>	50
<i>Associating the app with the Store</i>	51
<i>Verifying the app with the Windows App Certification Kit</i>	54
<i>Signing the appxupload file</i>	56
<i>Deployer IDE</i>	56
<i>Publishing in the Windows Store</i>	56



Author	Version	Date
Product	5.3	23/6/2015
	5.4	20/05/2015
	1394861 - new versioning interface	30/06/2015
	1395896- document relative locations for profiles	1/07/2015
	1416644 - override URLs and confirm save changes	22/07/2015
	1413277 - lang map updates	28/07/2015
	1313006 - run Jenkins as administrator	28/07/2015
	1497065 - Cordova plugins	13/10/2015
	1504803 - Cordova push notifications	20/10/2015

Document History


Comments:



About this Guide

This guide is designed for edgeConnect users who are familiar with the edgeConnect IDE and who have experience of producing simple solutions with the tool. This guide will explain how to develop and deploy hybrid solutions for mobile devices.

Online Help

The online help provides a full reference guide for all the features of the tool and can be used in conjunction with this guide. It is accessed via the Help item on the standard IDE toolbar or by clicking  on the standard button bar. The contents provide a list of all the major sections within the help. The index provides an alphabetical list of all the topics in the help, which is searchable. The search facility can be used to search for all topics related to a keyword you have entered.

Licensing and Technical Support

Prior to starting any development work described in this guide, it is assumed that the edgeConnect development tool has been set up correctly for your environment. Product installation, setup and registration are described in the Installation Guide.

For technical support, please contact Temenos Product Support Portal at:

<http://www.temenos.com/contact-us>

For more information about Temenos Customer support, please see:

<http://www.temenos.com/services/temenos-customer-support>.



Hybrid Solutions

A hybrid application is an edgeConnect application wrapped in a native mobile “shell” which can be run on a mobile platform. It is built like a standard non-hybrid application, but has some additional features which are required for the mobile environment. When the solution is deployed, the platform specific elements of the application are added. It can then be installed on a mobile platform.

Build Process

The diagram below illustrates the basic steps in the hybrid build process.



You will need to develop and test the project in the edgeConnectIDE prior to deployment.

Anatomy of a Hybrid Application

A hybrid application consists of a number of composite parts:

- Server application
- Shell application
- HTML Resources
- Offline Pages

Server Application

This is the edgeConnect application developed and tested using the edgeConnect IDE. This is also accessible from a browser.

Shell Application

This is the native part of a hybrid application and is therefore specific to the mobile platform the hybrid solution is installed on. It consists of the native Android or iOS wrapper which calls the edgeConnect server application. The shell uses a WebView, which is an embedded browser, to display the edgeConnect application.

HTML Resources

These are

- css
- js
- html
- img (png, gif, jpg, etc)

files that are needed by the edgeConnect application. The shell application also stores these files on the phone. It serves them up instead of downloading them from the server in order to reduce bandwidth and speed up the application. The server application maintains a version number so that the shell application can stay in sync with the server by downloading html resources that have subsequently changed.

Offline Pages

Offline pages are content which is displayed to the user when there is no internet connection. They must not be reliant on any connection to the edgeConnect server application but should enable the user to navigate back to the online application if required.

Error Phases

Specific error phases need to be created for a hybrid application relating to the availability and validity of the internet connection.

- Offline Error Phase (mandatory) - displayed when the phone has no internet connection. This page should allow offline navigation to other offline pages, as well a button or more that should allow navigating back to the online application.
- SSL Error Phase (optional) - displayed when the server's SSL certificate is not valid.



Terminology

The following shorthand terms will be used in the document.

- **%PROJECT_HOME%** the home folder of the project
- **%APPLICATION_NAME%** the application name as selected inside a Profile
- **%DEPLOYMENT_LOCATION%** destination folder as selected inside a Profile
- **\$INSTALL\$** the directory where edgeConnect is installed.
e.g. c:\Program Files\Temenos\edgeConnect



Build

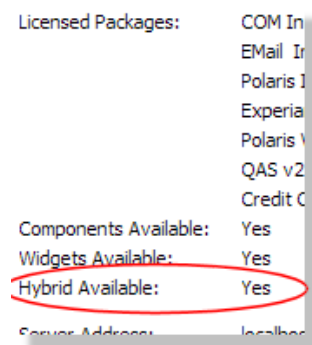
Hybrid Licence

In order to develop a hybrid project, you need to have set up an edgeConnect environment (either Standard or Enterprise) and have a valid licence which allows hybrid development.

To Check for a Valid Licence

- 1 Open the edgeConnect IDE
- 2 Click **Help** on the IDE toolbar
- 3 Click **About** from the dropdown menu

In the About dialog box, you will see **Hybrid Available:Yes** if you have a valid licence installed.



If this is not shown, you will need to obtain a licence. Please contact **24by7@temenos.com** to obtain a valid licence.

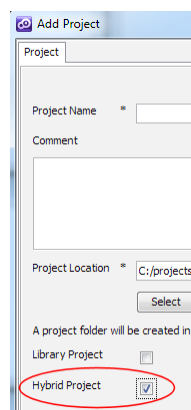
For further information about installing the edgeConnect development environment, please see the Installation Guide.

Add a Hybrid Project

A hybrid project is built in the same way as any other edgeConnect solution, but it must be marked as a hybrid project when it is created. This creates an additional hybrid folder, inside %PROJECT_HOME% which contains default icons, images, and certificates.. You can change an existing non-hybrid project to a hybrid project at a later date.

To Create a Hybrid Project

- 1 Click **File** on the IDE toolbar
 - 2 Click **New** from the dropdown menu
- The **Add Project** dialog appears.
- 3 Enter the project name and an option comment
 - 4 Check the **Hybrid Project** checkbox





This will mark your project as a hybrid project. This option will only appear if you have a hybrid licence. To check if you have a hybrid licence, display the Help > About box and it will be listed as Hybrid Available.

- 5 Click **OK** to create the project

Successfully Saved Project should appear in the status bar at the bottom of the IDE window.

A new folder, "hybrid", should be created inside %PROJECT_HOME%; it should contain default icons, images, and certificates. If this does not appear, then run the project once from the Presentation Editor.

Build a Hybrid Project

Building a hybrid project is similar to a non-hybrid project, but certain features must be included in order for the project to function correctly in the mobile environment. These are described below.

Offline Pages

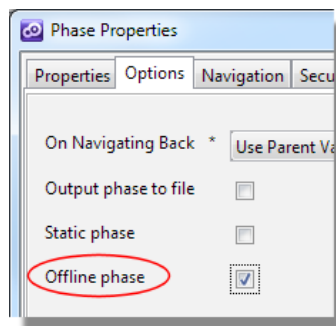
A hybrid application differs from a non-hybrid one in that it needs to contain "offline pages". These are required to be shown when no internet connection is available. They must contain navigation which allows the user to return to the online application. They may also enable navigation to other offline pages. They must not be reliant on anything from the edgeConnect server as there will be no connection available to it.

To Mark Phases as Offline

- 1 Double click phase you wish to mark as offline in Process Editor

The Properties dialog is displayed.

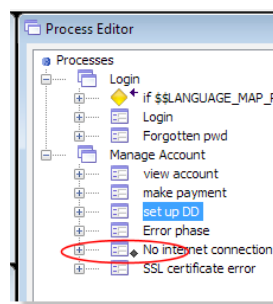
- 2 Select **Options** tab
- 3 Check **Offline Phase**



Note that this option is only available for Hybrid projects.

- 4 Click **OK**

The offline Phase is added to the Process. An additional icon is used to mark offline Phases in the Process.



Note that if a hybrid project has no offline phases, a warning will be displayed in the Verifier.



Error Phases

You will need to create a specific error phase:

- when the mobile device is not connected to the internet

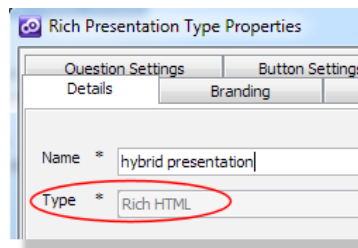
and an optional error phase:

- when the server's SSL certificate is not valid

These pages should be simple pages, without complex rules, because they will be used while the phone is not connected to the internet.

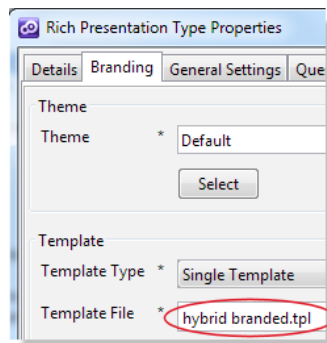
Presentation

Only use the **Rich HTML Presentation** type for hybrid projects. Do not use any other presentation type.



Template File

You may wish to modify the presentation's template file to suit the hybrid scenario, for example, preventing zooming or tap highlighting.



%PROJECT_HOME%/templates/hybrid branded.tpl template contains some useful declarations for hybrid which are found by searching for “hybrid” in the html.

index-hybrid.html

The first page that gets loaded when the hybrid application starts is %PROJECT_HOME%/index-hybrid.html. This page is stored on the phone. From here, the hybrid shell:

- populates DEVICE_INFO variable
- populates BROWSER_FEATURES variable
- submits information to the server.

This page is similar in functionality to index.html in a non-hybrid application. Its default appearance is a blank page, therefore you may want to add a background so that the user sees something while waiting for the first actual page of the application to display. If you have more than one process or presentation in your edgeConnect project, you may also want to change the values for the hidden input fields “PRODUCT” and “PRESENTATION_TYPE” in order to reflect the process and presentation that you want to use for the hybrid application.

Multi-language solutions

If you are creating multi-language solutions, using language maps, you will need to:

- load **all** language maps when starting the solution (use Auto-load on start up option).



- include a language selection rule in **all** the processes

The Language Editor (DeveloperIDE > Window > Language Editor) handles all language maps used by the hybrid solution. It will only list the loaded language maps, so only those will appear usable in Deployer.

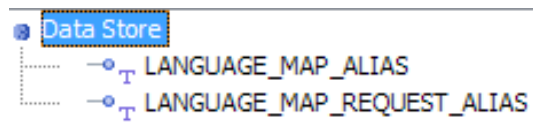
Language Selection Rule

The language selection rule must be set up as a pre-phase, process rule which is run before any other pre-phase rules which may be included in the phase.

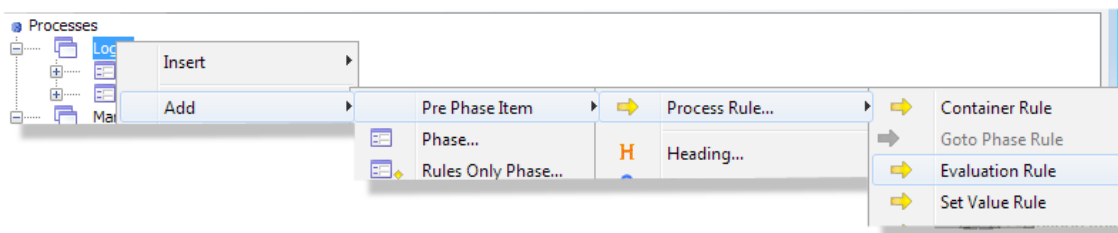
Create a “language selection” rule must be included in all your processes as a Pre-Phase rule. Please use the rule defined in the following fragment file; it shouldn’t be changed unless for customizing error messages/ phases, etc.

To Set Up Language Selection Rule

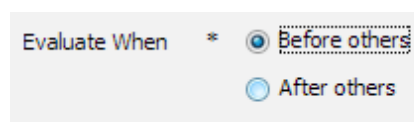
- Set up two data items, one for the current language map, the other for the requested language map



- Right click the Process in the Process Editor to add the rule
- Select **Add -> Pre-Phase Item -> Process Rule -> Evaluation Rule**

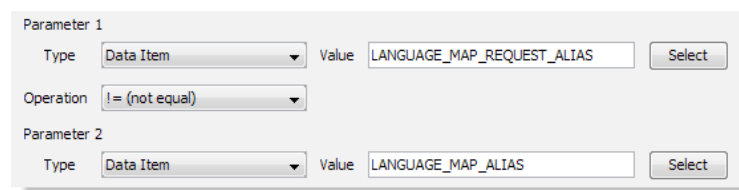


- Select **Definition** tab
- Evaluate When:** select **Before Others**



This will ensure that the language selection rule is evaluated before any other pre-phase rules.

- Select **Expression** tab
- Add the evaluation condition shown below



Then add a true rule to test if the requested language map value is not spaces

- Right click the rule you have just added, select **Add -> True Rule -> Evaluation Rule**
- Add the evaluation condition shown below



Parameter 1
Type: Data Item Value: LANGUAGE_MAP_REQUEST_ALIAS Select
Operation: != (not equal)
Parameter 2
Type: Value Value: Select

Then add a true rule to set the language map to the requested value

- 10 Right click the rule you have just added, select **Add -> True Rule -> Other**
- 11 **Rule Class:** select **Set Language Map Rule**
- 12 Click **Details** tab and set up rule as shown below

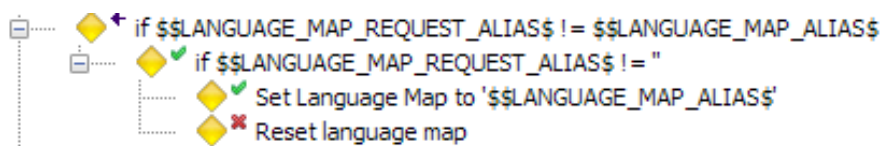
Add 'Set Language Map Rule'
Rule Name * Set Language Map
Definition Error Handling Details
Action * ☐ Reset the language map
☒ Set a language map
☐ Search for language map by ID
Language Map Alias * \$\$LANGUAGE_MAP_ALIAS\$
Not Found Action * Generate error

- 13 Click **OK**

Then add a false rule to the “evaluate for spaces” rule to reset the language map if the value is spaces.

- 14 Right click the previous space evaluation rule, select **Add -> False Rule -> Other**
- 15 **Rule Class:** select **Set Language Map Rule**
- 16 Click **Details** tab, select **Reset the Language Map**
- 17 Click **OK**

The completed rule should appear as shown below.



The structure of this rule should not be changed except to change error messages or phases.

Native Device Features

When you are creating a hybrid solution, you may wish to use some features which are provided by the device itself, for example device orientation, camera, contacts list. The following Cordova Hybrid widgets are distributed with the edgeConnect product to enable mobile device features to be used in your solution:

- **Back button** - allows the back button on the device to be used to navigate the solution
- **Device storage** - allows data to be stored permanently or temporarily on the mobile device
- **Notifications** - allows visual, audible, and tactile device notifications
- **Accelerometer** - allows you to capture device motion in the x, y, and z direction

This can be used for implementing logic for shaking or facedown. For example, we might log out the user



when he shakes the phone or when the phone is positioned horizontally with the face down.

- **Camera** - provides access to the device's default camera application This allows you to take pictures and choose images from the system's image library.
- **Geolocation** - provides access to location data based on the device's GPS sensor or inferred from network signals

When edgeConnect is installed the widget files are contained in:

\$INSTALL\$/IDE\widgets

and when you include one in your project the appropriate files are copied to:

%PROJECT_HOME%\templates\widgets

The widget folder contains all the necessary files as well as an explanation of how to use the specific widget and some sample code. Some examples of using these widgets are given below, but for more information on using widgets, please see the Widget Developers Guide.

Using Cordova Hybrid Widgets

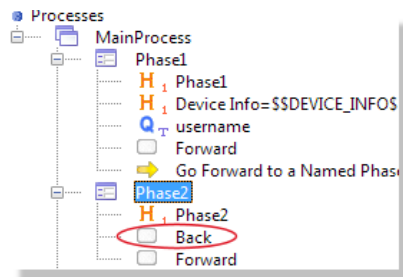
Native device features are triggered using either question, button or display button elements in the edgeConnect solution. When you link the native device behaviour to a button, you can choose to generate the body of the button in the solution or not. If you do not, the behaviour will exist on the mobile device, but you will not see anything in the solution.

You can link the device behaviour to a display button when there is no need to record the action in the data store.

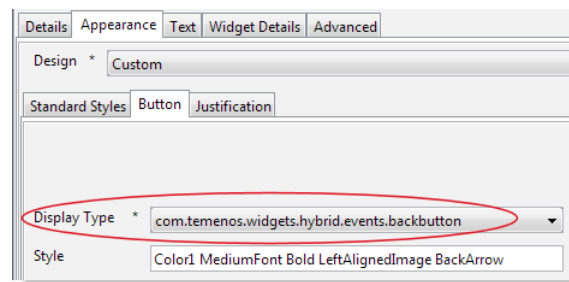
To Use Native Device Features - Button Example

- 1 Add a button/question/display button to your solution to trigger the native device behaviour.

The example will enable the back button on the mobile device to be used to navigate backwards in the solution.



- 2 In the **Presentation Editor**, select **Custom** for the Design.
- 3 Click the **Button** tab, select the required widget from the drop down list of **Display Types**

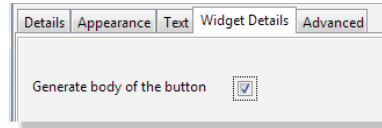


- 4 Configure the widget in the **Widget Details** tab

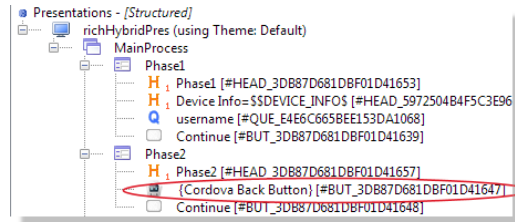
In this case, check the Generate body of the button checkbox if you wish to see a back button displayed in the solution. If you do not wish to see a back button, leave this unchecked. You will still be able to use



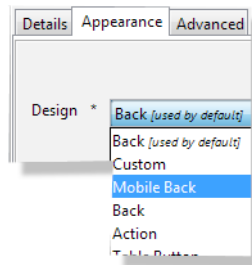
the back button on the mobile device to navigate backwards in the solution.



5 Click OK

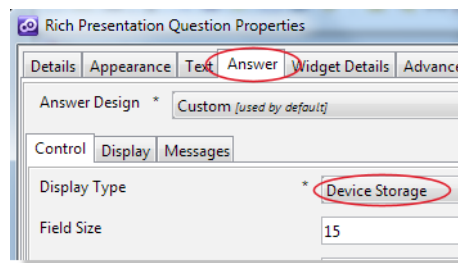


Note that if you want multiple back buttons to behave in this way, you can create a design for this widget and apply the design to any back buttons you require to show this behaviour.

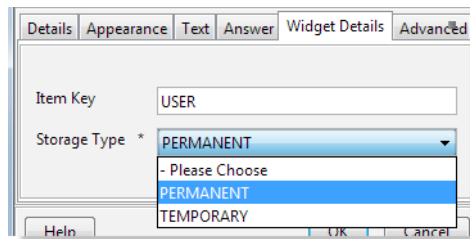


To Use Native Device Features - Question Example

When you link a device feature to a question, you follow the same steps as for a button. You need to select the mobile widget as the display type for the answer. For example you can use native device storage to store the value of a question answer.



In the widget details tab, you need to specify an item key i.e. a name for the value and what type of storage to use; either permanent or temporary.



When the application is loaded, if a value is no provided by the user and it has been stored permanently on the device, the field will be populated with the value from the device. For example, you may use this to store the user name for a personal banking application.

DEVICE_INFO

When you are using a mobile widget, at application startup device information is returned to the server. This can



be accessed using the System word `DEVICE_INFO`.
It contains the following properties:

- `model` – the model of the mobile device e.g. Samsung Galaxy S III will return “GT-I9300”
- `platform` – the OS of the device e.g. Samsung Galaxy S III will return “Android”
- `version` – OS version e.g. “4.3”
- `cordova` – the Apache Cordova version running on the device
- `isHybrid` – will always contain the value Y when a mobile device is used

The image below shows the device info displayed when the solution is executed on a mobile phone emulator.



```
Device Info:
isHybrid:Y
platform:Android
model:Custom Phone - 4.4.4 - API 19 - 768x1280
cordova:3.5.1
uuid:f9b828086f668d64
version:4.4.4
```

Creating your own Cordova Hybrid Widget

The Cordova Hybrid widgets supplied with edgeConnect use Apache Cordova Plugins that allow a mobile application developer to access native device functions from JavaScript. A plugin is a bit of add-on code that provides a JavaScript interface to native components. If you require functionality which is not provided by widgets distributed with the product, you can create your own widgets that use the Cordova Plugins distributed with edgeConnect.

As a starting point for creating your own widget, you can use the Cordova Hybrid widgets distributed with the product as an example. For more information on widgets, please refer to the Widget Developers Guide.

Cordova Plugins provided with edgeConnect

All the main Cordova API features are implemented as plugins and are described briefly below. These plugins are included in the edgeConnect install. When edgeConnect is installed, the javascript files for Cordova are contained in:

\$INSTALL\$IDE\HybridTemplate\html\js\cordova.

When you create a new edgeConnect hybrid solution, they are copied to the following location in your project:

%PROJECT_HOME%\html\js\cordova

At deploy time, these files are also copied into the hybrid shell so that the shell application can use the files stored on the device instead of downloading them from the server. They are treated as html resources, so can be download as new versions become available on the server.

If you are not using the standard template, you will need to include **cordova_loader.js** in the head section of the template header for your solution. This will load the javascript libraries required to invoke native device functions. The standard template will automatically include the cordova loader if it is a hybrid project.

In the Cordova documentation you will see that all the examples load the `cordova.js` file. Note that edgeConnect hybrid uses a modified loading mechanism, so you only need to use `cordova_loader.js` instead of `cordova.js`. Once cordova is included, you can consult the API of the plugin required and call the native functionality.

Camera

This plugin provides an API for taking pictures and for choosing images from the system's image library.

See <https://github.com/apache/cordova-plugin-camera> for more information.

SplashScreen

The SplashScreen Cordova plugin is used at application startup.

See <https://github.com/apache/cordova-plugin-splashscreen> for more information.



Device

The Device Cordova plugin is also used at application startup to transmit the device information to the server. This is used to populate the properties of the system word `DEVICE_INFO` as follows:

- **model**: the model of the mobile device accessing the application e.g. Samsung Galaxy S III will return "GT-I9300"
- **platform**: the OS of the device e.g. Samsung Galaxy S III will return "Android"
- **version**: the OS version e.g. Samsung Galaxy S III will return "4.3"
- **cordova**: the cordova version running on the device
- **isHybrid**: this will have the value Y when a mobile device is used

Accelerometer

This plugin provides access to the device's accelerometer. This is a motion sensor that detects the change in movement relative to the current device orientation, in three dimensions along the x, y, and z axis. This can be used for implementing logic for shake or facedown. For example, you might want to log out the user when he shakes the phone or when it is positioned horizontally with the face down.

See <https://github.com/apache/cordova-plugin-device-motion> for more information.

File and File Transfer

The File plugin implements a File API allowing read/write access to files residing on the device.

See <https://github.com/apache/cordova-plugin-file> for more information.

The File Transfer plugin allows you to upload and download files.

See <https://github.com/apache/cordova-plugin-file-transfer> for more information.

Notification

This plugin provides access to some native dialog UI elements. This is not part of the push notification functionality.

See <https://github.com/apache/cordova-plugin-dialogs> for more information.

Vibration

This plugin provides a way to vibrate the device.

See <https://github.com/apache/cordova-plugin-vibration> for more information.

Network Information

This plugin provides information about the device's cellular and wifi connection, and whether the device has an internet connection. It also allows you to register event handlers for the online and offline events i.e. when the phone goes online or when the phone goes offline.

See <https://github.com/apache/cordova-plugin-network-information> for more information.

Geolocation

This plugin provides information about the device's location, such as latitude and longitude. Common sources of location information include Global Positioning System (GPS) and location inferred from network signals such as IP address, RFID, WiFi and Bluetooth MAC addresses, and GSM/CDMA cell IDs.

See <https://github.com/apache/cordova-plugin-geolocation> for more information.

Media

This plugin provides the ability to record and play back audio files on a device.

See <https://github.com/apache/cordova-plugin-media> for more information.

Contacts

This plugin provides access to the device contacts database. It allows you to search or create/edit/delete contacts.

See <https://github.com/apache/cordova-plugin-contacts> for more information.



InAppBrowser

This plugin provides a web browser view that displays when calling `window.open()`. Note the window that gets created is a standard browser, it does not benefit from the cordova APIs.

See <https://github.com/apache/cordova-plugin-inappbrowser> for more information.

Whitelist

This plugin is a Cordova internal plugin that allows to control what pages, external apps and external resources can be loaded by native code or by the WebView.

By default, all external links and applications can be opened. If you want to restrict this, you can provide a customised `config.xml` file in `%PROJECT_HOME%/hybrid/android/res/xml/config.xml`. A starting point is the existing `config.xml` file on the android slave found in `$INSTALL$/Jenkins/home/workspace/shell/platforms/android/res/xml/config.xml`.

You can also customise what network requests are allowed to be made by the WebView. This is done by changing the meta `http-equiv="Content-Security-Policy"` declaration in your presentation's template file. See <https://github.com/apache/cordova-plugin-whitelist> for more information.

Using Third party Cordova plugins

Depending on your requirements, you may need functionality which is not provided by the Cordova plugins included in edgeConnect. This functionality may already exist in other Cordova plugins not readily available in edgeConnect Hybrid or you may need to create your own plugin.

Creating a new plugin is beyond the scope of this document, however, the Cordova site contains a Plugin development guide that contains all the information needed to develop a new plugin. A good strategy is to study the plugins from the Cordova plugin registry. You can find an existing plugin that resembles what you need and adapt from there.

The steps required to use a third party plugin are detailed below.

Install Cordova

First, you will need to install Cordova (see <http://cordova.apache.org/docs/en/5.1.1/guide/cli/index.html>):

- Download and install node.js
- Open a console and install cordova-cli. You should install the exact version of Cordova that is used by edgeConnect hybrid. Note that it is not advisable to install a different version of Cordova. In order to install cordova-cli run the command:
 - Windows: `npm install -g cordova@5.1.1`
 - Mac OS and linux: `sudo npm install -g cordova@5.1.1`

Create a new Cordova project for development and testing

It is recommended that you first develop and test your plugin in a new project and only integrate in an edgeConnect hybrid project after the plugin is working as required. The main steps for this process are described below.

To Create a New Cordova Project

- 1 Run the command: `cordova create hello com.example.hello HelloWorld`
In the command above, "hello" is the name of the directory that will be created for the project, "com.example.hello" provides your project with a reverse domain-style identifier, and "HelloWorld" is the name of the project.
- 2 Add a platform to the project.
This explained on the Cordova site. Note that your environment must contain all the necessary information for locating the platform sdk tools.
- 3 You can reuse elements from the configuration you have already done for starting the Jenkins android, windows and iOS slaves.

For example, for android on windows the batch file used to start the android Jenkins slave, can be adapted by



changing only the last line:

```
set JAVA_HOME=d:\java\jdk\jdk1.6.0_31
set ANT_HOME=d:\java\tools\apache-ant-1.9.2
set PATH=%JAVA_HOME%\bin;%ANT_HOME%\bin;%PATH%
set ANDROID_HOME=d:\java\ide\androidEdge\sdk
cordova platform add android
```

A similar approach is recommended for Windows Phone and iOS.

In this example, the newly created android project is located in hello/platforms/android. You can import the project in your favourite editor for the respective platform.

Add a Cordova plugin to your new project

A cordova plugin is a directory containing:

- plugin.xml: a plugin descriptor file
- the javascript source files
- the native source files (iOS, android, windows phone, etc).

The plugin can reside on the internet or be a local directory.

To Add a Cordova plugin from a Remote directory

- 1 Locate the plugin in the Cordova plugin registry
In our example this corresponds to the “hello” directory
- 2 From the plugin directory, run the command
cordova plugin add <plugin package>
The plugin package is the name of the plugin in the plugin registry.

To Add a Cordova plugin from a Local directory

- 1 From the project’s directory run, the command
cordova plugins add <path to directory containing the plugin>,
Put quotes round the path name if it contains spaces.

Note that if %ANDROID_HOME%\tools is not in %PATH%, you will receive a warning that “android is not recognized”. You can safely ignore that warning.

Integrating a Cordova plugin in EdgeConnect hybrid solution

Including Cordova Plugins in an edgeConnect project adds some extra requirements as the Cordova source files included in edgeConnect are slightly modified.

- For iOS, Cordova plugins can only be added on a Mac OS machine
- For WP8, Cordova plugins can only be added on a Windows 8 machine
- Some platform specific Cordova configuration files have been moved and renamed in edgeConnect
- the plugins.js files used by the web server are located in %PROJECT_HOME%\html\js\cordova\

The restrictions mentioned above mean that the command “cordova plugin add <plugin>” must run on specific operating systems, but the js files must appear both in the platform specific packages and in the mentioned solution folders.

To meet these requirements, it is necessary to configure additional plugins from the Developer IDE and make the slaves (already running on specific OSs) execute the Cordova commands. Therefore you will need to:

- configure the plugins in the Developer IDE
- run a deployment using at least one hybrid platform (WAR file is not required)

When the slaves perform their specific packaging they also add the extra plugins and create the js files needed for the hybrid app and also for the web server. When the build is finished, the necessary js files are automatically copied to the solution.



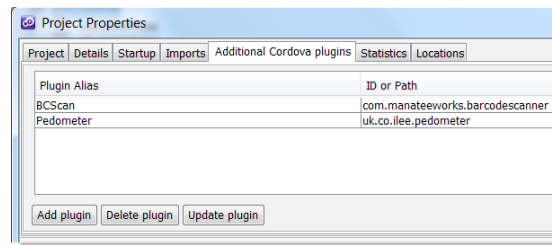
It is important to note that:

- The IDE source files are not modified. Master and slaves also remain unchanged and can be used for other projects.
- Even if the plugins are configured in the IDE, they won't be actually added until a build is performed. Although this may seem inconvenient, it has the following advantages:
 - normally you need a hybrid app deployed anyway in order to test the plugin properly;
 - using the slaves for this operation releases the user from the task of setting a multi OS environment just to add Cordova plugins.
- If you use Enterprise edgeConnect you will have to check in the javascript files generated in this process.

Note that Cordova must be installed on each slave machine. However, it is not required on the developer or master machine.

Configuring the additional plugins

In the Developer IDE menu click **Project->Properties** and select the **Additional Cordova plugins** tab. The list is empty by default.



You can add new plugins, or remove or modify a previously added plugin. Each plugin definition consists of the following descriptors:

- **Alias:** a custom name to easily identify the plugin
In the case of locally downloaded or developed plugins, this will be the name given to the plugin in the application
- **ID or Path:** this can either be an ID of a plugin from Cordova plugins repository or the path to a local folder where the plugin resides (either previously downloaded or developed as described in previous chapters)

After configuring the plugins, the project must be saved and then a hybrid deployment must be made. Any changes will not take effect until that is completed.

Making a deployment

You must perform at least one hybrid build in order to obtain the plugin files and all hybrid builds must have the project configured properly for each platform. Building a WAR file is not necessary.

Running the web server

Running a web server either on a dedicated server or started from the Developer IDE can only contain the added plugins after performing the configuration and deployment steps described above. Consequently, the order of the deployment is:

- generate offline pages
- build hybrid
- build WAR.

To run the server with the new plugins from Developer IDE might require reloading the project so the IDE indexes the new files. You will need to accept all the modified javascript files.

Calling the plugin

Now the plugin is integrated you can make use of it. Depending on what the plugin does, the usual way to call the plugin is to create an EdgeConnect widget that calls the javascript API of the plugin. This is described in the previous section and for more information on widgets, please see the Widget Developers Guide.



Debugging a 3rd party plugin in EdgeConnect Hybrid

Once the plugin is added, you will have to test the new functionality. If you have integrated and tested the plugin with a blank Cordova project, the plugin integration in edgeConnect hybrid should work smoothly. However, if you have deployed your project and the web application, but the hybrid app is not working as expected, the best way to debug it is to get the sources of the native project (android, iOS, Windows Phone), load them in your IDE and debug. You can find the sources on the machine that has Jenkins Slave for the platform. For example, for android, the sources would be in:

```
$INSTALL$/jenkins/.jenkins_android/workspace/build_android/shell/platforms/android.
```

- For Android, you will need Android Studio or Eclipse ADT in order to import the project.
- For Windows Phone, you will need Visual Studio 2013 Professional.
- For iOS, you will need Xcode 7.

Progress Bar

It is recommended the progress bar feature is enabled. This allows a progress bar to be shown when a solution submits or makes an AJAX call. Having a visual indicator that some processing is occurring makes a better user experience.

Customising the Progress Bar

The default option is to show a progress bar with no text. The script to enable this feature is in the hybrid template file %PROJECT_HOME%/templates/hybrid branded.tpl, contained in the <script> block that comes before the closing of the <body> tag. If you want to display some text with the progress bar (for example: Loading...) you will need to change some of this script.

There are 2 options:

- The text to display with the progress bar is internationalized and comes from the hybrid messages editor, under the key "loading_wait". You need to change the call navigator.spinnerDialog.show(false, null), in the function showProgress(), to pass "true" as the first argument:

```
navigator.spinnerDialog.show(true, null);
```

The text you have specified for the message will be displayed with the progress bar.

- If you want to display custom text, you need to change the call to navigator.spinnerDialog.show(false, null). Pass "true" as the first argument and your custom text as the second argument:

```
navigator.spinnerDialog.show(true, "custom loading text");
```

It is also recommended that you enable the progress bar in the file %PROJECT_HOME%/index-hybrid.html. This page is stored on the phone, and it submits an initialization form that launches the EdgeConnect application. The progress bar is not enabled by default in this file. In order to enable it, uncomment the function showProgress() by deleting the starting /* and the closing */. The same customizations as above can be applied when calling navigator.spinnerDialog.show().

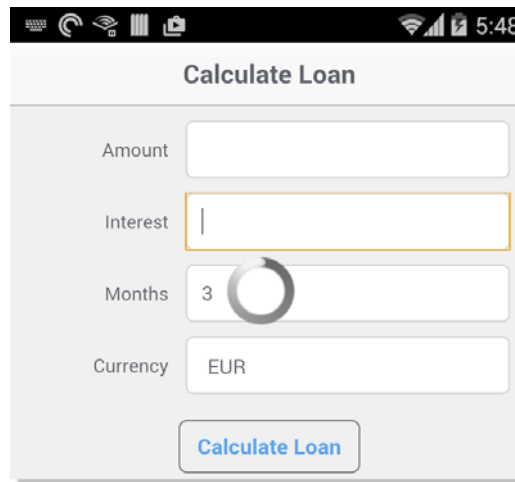
Appearance

Although the native progress bar should be sufficient for most mobile applications, you may wish to customize the look and feel of the progress bar to complement that of the application.

Android

The layout for the progress bar in Android is the file %PROJECT_HOME%/hybrid/android/res/layout/custom_progress_layout.xml.

In Android hybrid the progress bar consists of a rotating circle, its exact appearance is dependent on the android version.



You can change:

- positioning on the screen (currently the progress bar is positioned in the middle of the screen)
- style of the progress bar
- color or size of the text (if you choose to display text with the progressbar).

Android provides several styles for the progress bar. You can use them by adding a style attribute to the ProgressBar element:

style="@android:style/Widget.ProgressBar.Large "

Possible values for the style are:

- Widget.ProgressBar.Horizontal
- Widget.ProgressBar.Small
- Widget.ProgressBar.Large
- Widget.ProgressBar.Inverse
- Widget.ProgressBar.Small.Inverse
- Widget.ProgressBar.Large.Inverse

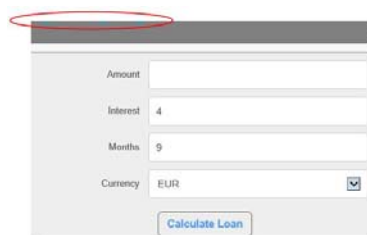
See the Android reference for the progress bar.

For a custom progress bar rotating icon, you can start with this article: <http://www.hrpin.com/2011/09/how-to-make-custom-indeterminate-progressbar-in-android-or-how-to-change-progressbar-style-or-color>, or google "android custom progress indicator".

Windows Phone

The layout for the progress bar in Windows Phone is the file %PROJECT_HOME%/hybrid/wp8/res/ProgressIndicator.xaml.

In Windows Phone hybrid the progress bar consists of a series of similar and equidistant colored blocks; they appear from left to right as the operation reaches completion.





You can change:

- positioning on the screen
- background
- text color and font size (if you choose to display text with the progressbar).

Windows

The layout for the progress bar in Windows is the file %PROJECT_HOME%/hybrid/windows/res/VisualElements/ProgressIndicator.xaml

In Windows hybrid the progress bar is the same as that for Windows Phone shown above.

You can change the positioning on the screen, the background, the text color and font size (if you choose to display text with the progressbar).

iOS

The layout for the progress bar in iOS is the file %PROJECT_HOME%/hybrid/ios/res/spinnerLayout.plist.

In the iOS hybrid shell the progress view is an iOS specific activity spinner placed in a square in the middle of the screen. A short explanatory text can be added below the spinner.



The user can customize the spinner appearance by editing the layout properties file mentioned above. Spinner properties which can be changed are:

- the rectangle background color
`<key>backgroundColor</key>`
`<string>#000000</string>`
- the transparency of the rectangle
`<key>alphaTransparency</key>`
`<real>0.5</real>`
- the height of the rectangle
`<key>height</key>`
`<integer>150</integer>`
- the width of the rectangle
`<key>width</key>`
`<integer>150</integer>`
- the spinner style – possible values: 0 = white large, 1 = white, 2 = grey
`<real>0.5</real>`
- the height of the rectangle
`<key>height</key>`
`<integer>150</integer>`
- the width of the rectangle
`<key>width</key>`
`<integer>150</integer>`



- the spinner style – possible values: 0 = white large, 1 = white, 2 = grey
`<key>spinnerType</key>`
`<string>0</string>`
- the text color
`<key>textColor</key>`
`<string>#FFFFFF</string>`
- the text font size
`<key>textFontSize</key>`
`<integer>12</integer>`

Testing Native Device Features

Once you have completed your hybrid project, you will need to test the solution either using a mobile device or a mobile device emulator.

To Test Native Device Features

- 1 Deploy the hybrid project
- 2 Run the hybrid project
- 3 Open the mobile device emulator or connect the mobile device
- 4 Install your deployed solution on the mobile device

You can now test the native mobile device features you are using in your project.

Using an Android Mobile Phone

You will need:

- to deploy the hybrid application for Android and obtain the .apk file
- a developer-enabled android phone that runs at least android KitKat (version 4.4)
- to keep the phone connected to the computer through USB.
- to install the .apk file on the android device.
- a Chrome browser for debugging.

Follow the steps from <https://developer.chrome.com/devtools/docs/remote-debugging>

It might be helpful to see all the log messages of the shell application. For this, install the Eclipse ADT (or Android Studio) and watch the LogCat output. Useful log messages to watch out for are the messages from the application and the messages having the tags "CordovaLog" or "CordovaActivity".

How to test the Windows app

After deploying a Windows profile, the app binaries are generated in

`%DEPLOYMENT_LOCATION%/.zip`

The zip contains the following:

- The file WindowsHybrid_AnyCPU_bundle.appxupload. This is for publishing the app in the Windows Appstore.
- The directory WindowsHybrid_Test. In this directory there is a file called WindowsHybrid_AnyCPU.appxbundle. This file can be deployed on a Windows 8.1 machine (desktop, laptop, Surface Pro 3 tablet, etc).



In order to deploy locally, on the target machine extract the directory `WindowsHybrid_Test` from the zip. If this is the first time that you are installing the app on your machine, you must first trust the application self-signed certificate. The self-signed certificate is the file `WindowsHybrid_Test\WindowsHybrid_AnyCPU.cer`. The procedure for trusting the certificate is as follows:

- Double-tap the certificate file in the folder and then tap **Install Certificate**. This displays the **Certificate Import Wizard**.
- In the **Store Location** group, tap the radio button to change the selected option to **Local Machine**.
- Click **Next**. Tap **OK** to confirm the UAC dialog, if this dialog is displayed.
- In the next screen of the **Certificate Import Wizard**, change the selected option to **Place all certificates in the following store**.
- Tap the **Browse** button. In the **Select Certificate Store** pop-up window, scroll down and select **Trusted People**, and then tap **OK**.
- Tap the **Next** button; a new screen appears. Tap the **Finish** button.
- A confirmation dialog should appear; if so, click **OK**. (If a different dialog indicates that there is some problem with the certificate, you may need to do some certificate troubleshooting. However, describing what to do in that case is beyond the scope of this topic.)

Note that trusting the certificate is a one-time operation; you do not need to repeat this for every deployment.

Another required operation needed is to get a developer license on the machine. The developer license is valid for 6 months, so you might need to do this again in the future. First, you need to have a Microsoft account. If you do not have one, you can register free at <https://signup.live.com/>. For getting a developer license, please refer to one of the two articles below: * <http://www.howtogeek.com/129535/how-to-sideload-modern-apps-on-windows-8/> * <https://msdn.microsoft.com/en-us/library/windows/apps/hh974578.aspx> (read the paragraph "Getting a developer license at a command prompt")

For installing the app, right-click on the file `WindowsHybrid_Test\Add-AppDevPackage.ps1` and select "Run with PowerShell" from the contextual menu. The PowerShell window may display a text asking for confirmation, type "yes" and press enter.

If the install was successful, go to the Windows 8 Menu and test the app. Note that the application can be uninstalled only from the Windows 8 Menu, thus you will have to manually uninstall before installing a new version.

Note that the Windows app requires an https URL for the webapp, and you'll have to deploy the EdgeConnect webapp to a server that has a valid SSL certificate, a self-signed certificate will not work. Also, if the app doesn't work (you see a white screen), you might try one of the following:

- Open Internet Explorer on the target machine and change the IE settings to bypass the proxy if your connection uses a proxy. (Internet Options->Connections->Lan Settings).
- Open Internet Explorer on the target machine and add the https URL for the webapp in the list of trusted sites. (Internet Options->Security->Trusted sites).

The Windows app does not support changing the URL of the webapp and for the check for updates servlet. Even if you enable "Development mode" in the Windows profile, you will not be able to change the URLs from the application's settings (they are shown, but they are read-only).

Note that the Windows app does not allow access to the native features of the device through cordova plugins, like contacts, camera, accelerometer, because cordova is not available for windows hybrid.

How to set access rights for windows 8.1

If you cannot start Jenkins master or slave because of access rights you must modify the access rights of the folder where the Jenkin is installed.



Start a command prompt with Administrator rights and execute these lines:

```
attrib -r +s C:\WorkingDir attrib -r +s C:\WorkingDir\*
```

where the WorkingDir is the path to the Jenkins installation folder.

appear from left to right as the operation reaches completion.

Cordova Push Notifications

Push, or server push, describes a style of Internet-based communication where the request for a given transaction is initiated by the publisher or central server. It is contrasted with pull/get, where the request for the transmission of information is initiated by the receiver or client.¹

Push notifications refer to all types of messages sent from a server to a mobile phone app regardless of platform. A similar mechanism is used for iOS, Android and WP8 devices, however a different broadcasting service is used for each platform.

In order to receive push notifications, the devices must be registered with the appropriate broadcasting service:

- The Amazon Fire OS implementation uses Amazon's ADM(Amazon Device Messaging) service.
- The Android implementation uses Google's GCM (Google Cloud Messaging) service.
- The iOS version is based on Apple APNS Notifications.
- The WP8 implementation is based on MPNS.aspx.
- Windows8 uses Microsoft WNS Notifications.

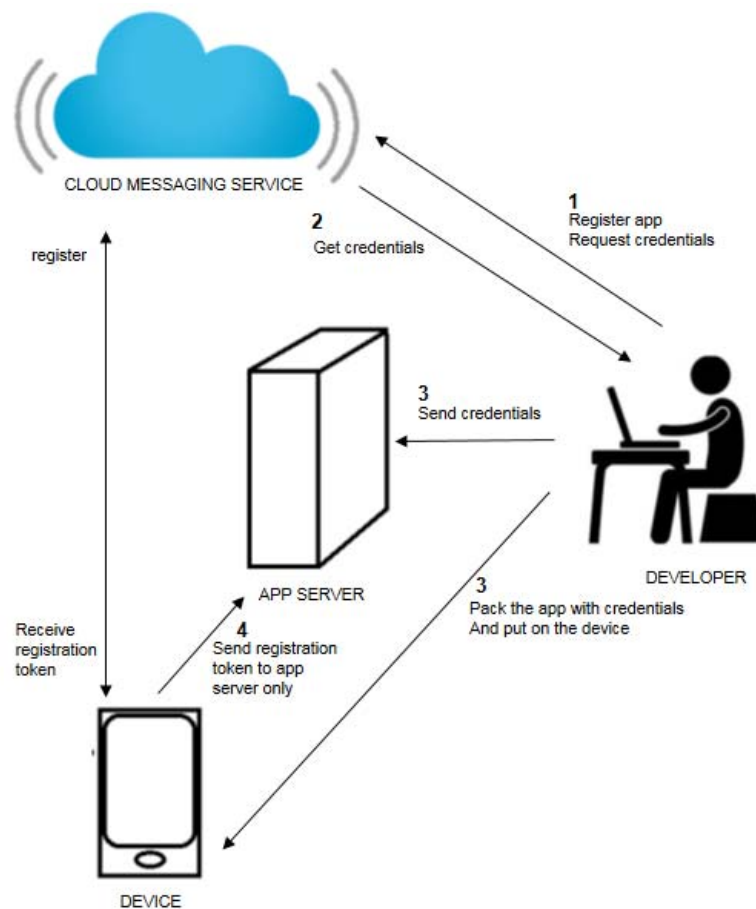
To Receive Push Notifications

The general steps you have to follow in order to have a mobile app that receives push notifications are shown below:

- Register your app with the appropriate message broadcasting service.
No details about the app itself are required, but the services will provide the required credentials that will be used only by your app and its registered devices.
- Pack into the hybrid (or native) app the credentials required by the device to register with the broadcasting services.
 - iOS: the certificate file is required (with push notification permission),
 - Android: a key, called sender ID, is required

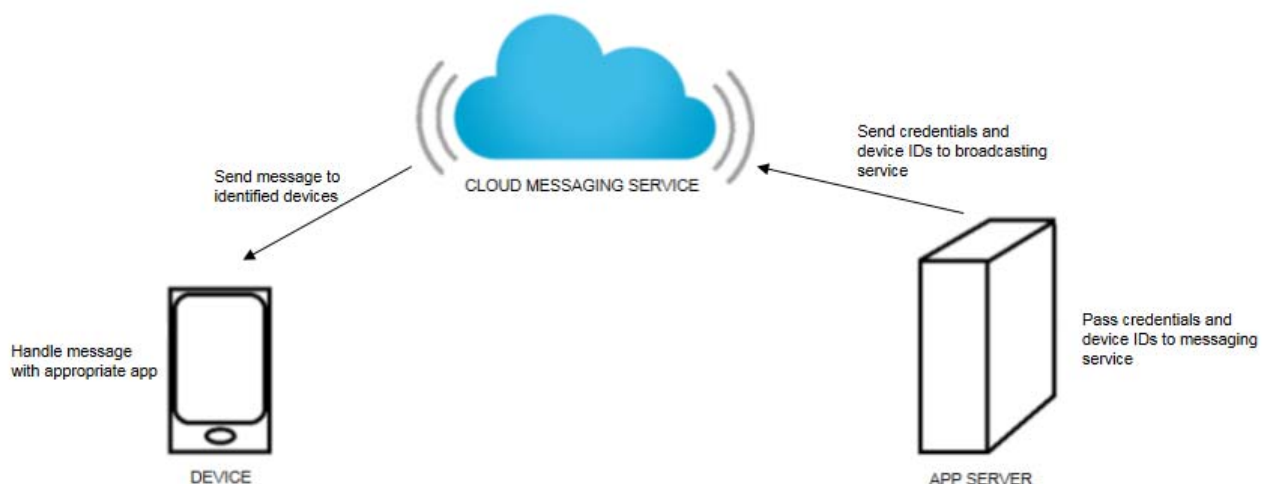
These credentials are also obtained when registering your app with the service.

- When the device first starts the app, it registers with the broadcasting service and obtains an identification token which is unique for the device and app. The device sends this token to the application server so the server can use it to send push notifications to the device.



Push Notification Message Flow

- Using the credentials obtained from the messaging service and the device registration tokens, send the message to the broadcasting service.
- The broadcasting service then analyses the credentials and the list of device IDs and send the message to all devices in the list
- The OS on the device receives the message from the broadcasting service and hands it to the native application which then shows it as a notification even if the application is turned off





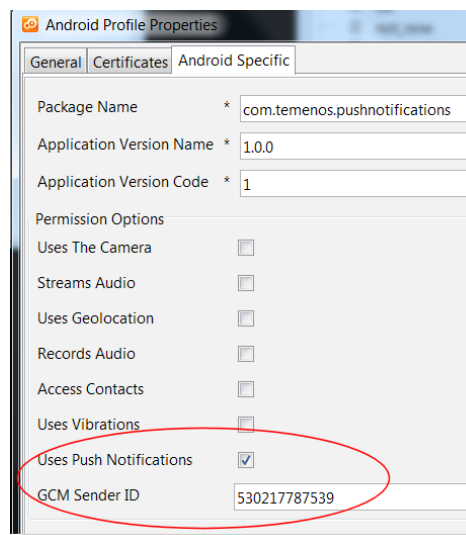
Although you can choose to allow push notifications or not, the Cordova Push Notification plugin is added by default to an edgeConnect hybrid project. The file **connect_hybrid_pushNotifications.js** exists in %PROJECT_HOME%\html\js.

Note that projects created using older versions of edgeConnect (pre-v5.4) will need to add the latest version of this file to their js library.

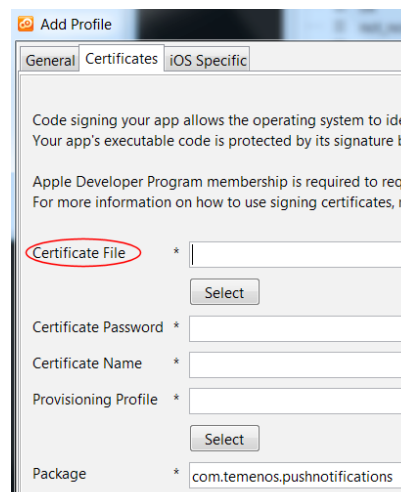
To Use the edgeConnect Cordova Push Notification Plugin

The following steps are required to complete the registration process described in Figure 1.

- 1 Access the appropriate cloud messaging services listed at the beginning of this document and register the application
You will receive the appropriate credentials either keys and/or certificate files.
 - iOS: certificate file
 - Android: API KEY for sender and SENDER ID for device
 - Windows Phone 8 has no authentication data; a unique URL is generated for each device-app pair when the device registers with the service
- 2 Pass the credentials to the devices through Deployer IDE using the profiles
Android Profile

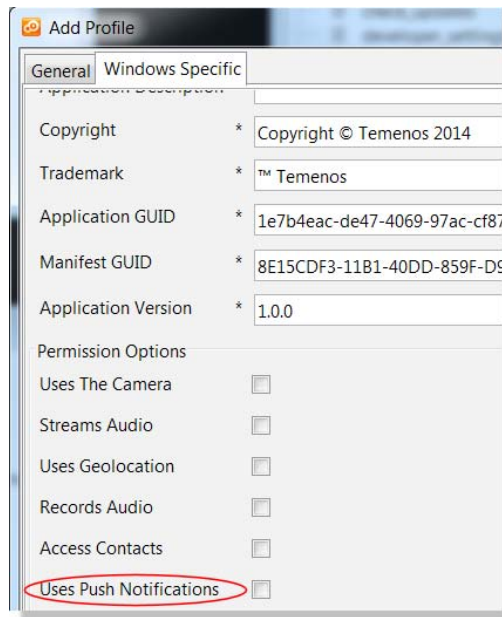


For iOS all credentials are passed through the certificate file (with push notifications allowed)





For Windows Phone 8 you only need to check Uses Push Notifications



It is important to note that edgeConnect has no support for push notifications in Windows 8 applications at this time.

- 3 Develop a message sender service as
 - part of the application
 - a separate application,
 - a third party application e.g. Microsoft Azure.

The app need to pass registration IDs from DeviceInfo (**DEVICE_INFO.cloudMessagingId**) to the message sender service.

You will also need to store the credentials for message sender authentication, for example API KEY for Google Cloud Messaging. Steps 1 and 2 in Figure 2 must be developed by the developer

- 4 In the deploy process “Use Push Notifications” must be checked for each profile.

For more information on how to use the plugin, see <http://plugins.cordova.io/#/package/com.phonegap.plugins.pushplugin>

Message Handling

In order to change the default message handling for a hybrid app you can change the Javascript code of the handlers. For this please see file **connect_hybrid_pushNotifications.js**



Deploy

After you have finished building and testing your application you will need to deploy your project to produce a file which is ready for release. The deployment process packages the edgeConnect solution with the appropriate native shell.

Deployment Set-up

You will deploy the project using the edgeConnect Deployer installed as part of the Standard IDE. However, before you can deploy your project, there are some additional set-up steps required which are required.

- Start Jenkins master and appropriate slave
- Set Server Connection details in edgeConnect Deployer

Note that, if edgeConnect is installed in Program Files on a local C: drive , you need Administrator rights in order to copy files within a folder from Program Files (required for Hybrid Master and Hybrid Slave). Therefore, you will need to start the servers with Administrator Rights (Right Click -> Run As Administrator).

Jenkins

Jenkins is an application that monitors executions of repeated jobs, such as building a software project. It consists of a Jenkins Master, which is the main Jenkins installation, and this coordinates the Jenkins slaves. The Jenkins slaves perform the specific builds for Android or IOS. Note that the the IOS hybrid app can only be built on an IOS system. The Android slave can be installed on any OS that allows installing the Android SDK. You will only need to start the specific slave for the mobile platform you are deploying to.

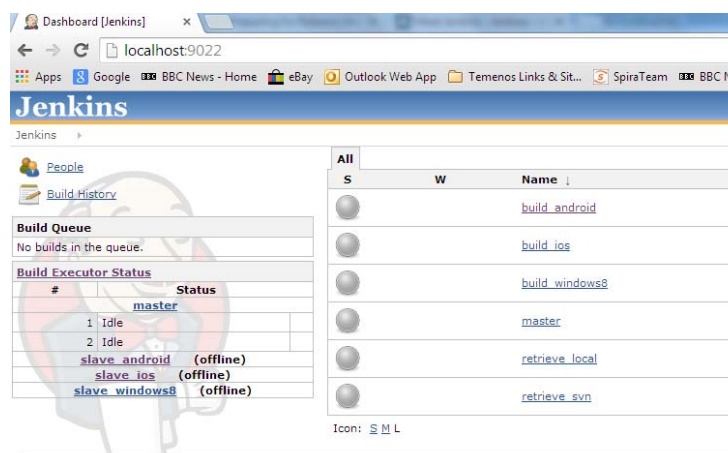
Starting Jenkins

The Jenkins Master requires:

- JDK (1.5+)
- Apache Ant (1.8.4+)
- %JAVA_HOME%/bin included in PATH
- %ANT_HOME%/bin included in PATH.

The Jenkins master is located on the install directory at \$INSTALL\$/jenkins/HybridMaster.exe. Start Jenkins by right clicking on the .exe file and clicking Run. If edgeconnect is installed within C:/Programs Files/InstallationFolder , you will need administrator rights to copy files in Program Files, which is required for Jenkins to run correctly. In this case click Run as Administrator.

Open the Jenkins dashboard in a browser, localhost:9022.



Alternatively, if you are using Windows, you can create a .bat file that sets the environment and then calls the HybridMaster.exe, as in the example below:



```
set JAVA_HOME=d:\java\jdk\jdk1.6.0_31
set ANT_HOME=d:\java\tools\apache-ant-1.9.2
set PATH=%JAVA_HOME%\bin;%ANT_HOME%\bin;%PATH%
start HybridMaster.exe
```

Again, you will need to run the .bat file with administrator rights if it is installed in C:/Programs Files/InstallationFolder. If this does not work, run the following from a command line started with administrator rights:

```
attrib -r + C:/Programs Files/InstallationFolder
attrib -r +s C:/Programs Files/InstallationFolder /*
```

and then run the bat file again with administrator rights.

Port Setting

If port 9022 is in use, you will need to change the port that the Jenkins master uses. This is in the settings file which is found in \$INSTALL\$/jenkins/HybridMaster.lax. Change this to a port number which is not in use.

```
# LAX.COMMAND.LINE.ARGS
# -----
# what will be passed to the main method -- be sure to quote arguments with spaces in them
lax.command.line.args=-f master.xml launch -Dport=9022
```

Starting Android Slave

The Android Jenkins Slave requires:

- JDK (1.7+)
- Apache Ant (1.8.4+)
- %JAVA_HOME%\bin included in PATH
- %ANT_HOME%\bin included in PATH.

In the newly installed Ant 1.8.4, please copy the file \$INSTALL\$/lib/ant-contrib-1.0b3.jar.

Android SDK and ANDROID_HOME variable

If you have not installed the Android SDK, you will need to do so. Please refer to the document \$INSTALL\$/installAndroidSDK.pdf for instructions on how to install the Android SDK. Once this has been done, set the environment variable ANDROID_HOME so that it points to the location of Android SDK.

The Jenkins Slave exe is located in \$INSTALL\$/jenkins/HybridAndroidSlave.exe and the settings file is \$INSTALL\$/jenkins/HybridAndroidSlave.lax. The slave needs to know where the Jenkins master installed. This is set in the settings file.

```
# LAX.COMMAND.LINE.ARGS
# -----
# what will be passed to the main method -- be sure to quote arguments with spaces in them
lax.command.line.args=-f slave_android.xml -DmasterServer=localhost -DmasterPort=9022
```

, the master server is located on the same machine as the slave, so localhost is used, otherwise you should use the IP address of the Jenkins Master.

Start the Android Jenkins slave by launching the .exe file. Alternatively, if you are using Windows, create a .bat file that sets the environment and then calls HybridAndroidSlave.exe, like in the example below:

```
set JAVA_HOME=d:\java\jdk\jdk1.6.0_31
set ANT_HOME=d:\java\tools\apache-ant-1.9.2
set PATH=%JAVA_HOME%\bin;%ANT_HOME%\bin;%PATH%
set ANDROID_HOME=d:\java\ide\androidEdge\sdk
start HybridAndroidSlave.exe
```

If it is installed in C:/Programs Files/InstallationFolder then you will need to run with administrator rights, as per the Jenkins Master, described above.



Starting iOS Slave

The iOS Jenkins Slave requires:

- XCode - refer to the \$INSTALL\$/Readme.txt for instructions on how to install XCode.
- Apache Ant (1.8.4+)

To install Apache Ant:

- download ant (version 1.8.4 or above), unzip, copy to /usr/local/apache-ant
- export PATH=/usr/local/apache-ant/bin:"\$PATH"
- echo 'export PATH=/usr/local/apache-ant/bin:"\$PATH"' >> ~/.profile

For iOS, the Jenkins Slave command is located in \$INSTALL\$/jenkins/Hybrid_iOS_Slave.command. The settings for the iOS Slave are in the file \$INSTALL\$/jenkins/Hybrid_iOS_Slave.command.lax.

The slave needs to know where the Jenkins master is, so modify the line in the lax file:

```
lax.command.line.args=-f slave_android.xml -DmasterServer=10.140.2.156 -DmasterPort=9022
```

In the example above, the master server is located on the machine with the IP 10.140.2.156, on port 9022.

Starting Windows Phone 8 Slave

The Windows Phone 8 slave requires the following:

- Windows 8 machine. The minimum OS version for this is Windows 8.
- Windows Phone 8 SDK. This is downloadable (free) from <http://dev.windows.com/en-us/develop/download-phone-sdk>. Make sure you install "Windows Phone SDK 8.0" only. Do not install a newer or older version. This can be installed on Windows 8 and higher, but **only on 64-bit (x64) platforms**.
- JAVA JDK. Available on the Oracle site <http://www.oracle.com/technetwork/java/javase/downloads/index.html>. We recommend version 7.
- ANT 1.8+. This is supplied with the edgeConnect installation or can be downloaded separately from <http://ant.apache.org>.

For Windows Phone 8, the Jenkins Slave exe is located in \$INSTALL\$/jenkins/HybridWP8Slave.exe. The settings for the WP8 Jenkins Slave are in the file \$INSTALL\$/jenkins/HybridWP8Slave.lax. The slave needs to know where the Jenkins master is, so modify the line in the lax file:

```
lax.command.line.args=-f slave_wp8.xml -DmasterServer=localhost -DmasterPort=9022
```

to set the masterServer and masterPort. In the example above, the master server is located on the same machine, so localhost is used, otherwise use the IP address of the Jenkins Master.

The Jenkins Slave needs JDK (1.7+) and Ant (1.8.4+), and that the %JAVA_HOME%/bin and %ANT_HOME%/bin directories are included in PATH. Manually install a JDK 1.7. An Apache Ant 1.9.3 is included in the edgeConnect installation. If you choose to install Ant yourself, in the newly installed Ant please copy the file \$INSTALL\$/linb/ant-contrib-1.0b3.jar.

Windows Phone 8 SDK and MSBUILD_HOME variable

If you have not installed the Windows Phone 8 SDK, you will need to do so. Please refer to the document \$INSTALL\$/instalWindowsPhone8SDK.pdf for instructions on how to install the Windows Phone 8 SDK. Once this has been done, set the environment variable MSBUILD_HOME to point to the location of the **32 bit** msbuild.exe.

In order to locate msbuild.exe, open a command prompt and execute the command:

```
reg.exe query "HKLM\SOFTWARE\Microsoft\MSBuild\ToolsVersions\4.0" /v MSBuildToolsPath
```




The output should be something like:

```
MSBuildToolsPath REG_SZ C:\Windows\Microsoft.NET\Framework64\v4.0.30319\
```

If there is “**Framework64**” in the output, eliminate the “64”, to give the correct path to use for Msbuild.exe, i.e. **C:\Windows\Microsoft.NET\Framework\v4.0.30319**

Start the Windows Phone 8 slave by launching the .exe file. Alternatively, if you are using Windows, create a .bat file that sets the environment and then calls HybridWP8Slave.exe, like in the example below:

```
set JAVA_HOME=D:\java\jdk\jdk1.6.0_31
set ANT_HOME=D:\java\tools\apache-ant-1.8.3
set MSBUILD_HOME=c:\Windows\Microsoft.NET\Framework\v4.0.30319
set PATH=%MSBUILD_HOME%;%JAVA_HOME%\bin;%ANT_HOME%\bin;%PATH%
start HybridWP8Slave.exe
```

Starting Windows 8 Slave

The Windows 8 slave requires the following:

- A Windows machine on which we can install the Visual Studio Express 2013 for Windows. The minimum OS version for this is Windows 8.1.
- Visual Studio Express 2013 for Windows. If you don't already have a licence for this, it is downloadable free from <https://www.visualstudio.com/en-us/downloads#d-express-windows-8>.
- JAVA JDK, available on the Oracle site <http://www.oracle.com/technetwork/java/javase/downloads/index.html>. We recommend version 7.
- ANT 1.8+. This is supplied with the edgeConnect installation or can be downloaded separately from <http://ant.apache.org>.

For Windows, the Jenkins Slave exe is located in \$INSTALL\$/jenkins/HybridWindowsSlave.exe. The settings for the WP8 Jenkins Slave are in the file \$INSTALL\$/jenkins/HybridWP8Slave.lax. The slave needs to know where the Jenkins master is, so modify in the lax file the line

```
lax.command.line.args=-f slave_windows.xml -DmasterServer=localhost -DmasterPort=9022
```

to set the masterServer and masterPort. In the example above, the master server is located on the same machine, so I've used localhost, otherwise you should put the IP of the Jenkins Master.

The Windows Jenkins Slave needs JDK (1.7+) and Ant (1.8.4+), and that the %JAVA_HOME%/bin and %ANT_HOME%/bin directories included in PATH. Manually install a JDK 1.7. An Apache Ant 1.9.3 is embedded in the EdgeConnect installation. If you choose to install Ant yourself, in the newly installed Ant please copy the file \$INSTALL\$/linb/ant-contrib-1.0b3.jar.

You will also need to set the directory where the Msbuild 12 executable is located.

Msbuild comes with Visual Studio Express 2013 for Windows, or with the paid version of Visual Studio 2013. Please refer to the document \$INSTALL\$/instalVisualStudioExpress2013.pdf for instructions on how to install Visual Studio Express 2013 for Windows. After installing, set the environment variable MSBUILD_HOME so that it points to the location of the 32 bit msbuild.exe version 12. Msbuild v12 is installed in C:\Program Files (x86)\MSBuild\12.0\Bin, but if you customized the installation of Visual Studio, the location might be different.

Alternatively, create a bat file that sets the environment and then calls HybridWindowsSlave.exe, like in the example below:

```
set JAVA_HOME=D:\java\jdk\jdk1.6.0_31
set ANT_HOME=D:\java\tools\apache-ant-1.8.3
set MSBUILD_HOME=C:\Program Files (x86)\MSBuild\12.0\Bin
set PATH=%MSBUILD_HOME%;%JAVA_HOME%\bin;%ANT_HOME%\bin;%PATH%
start HybridWindowsSlave.exe
```

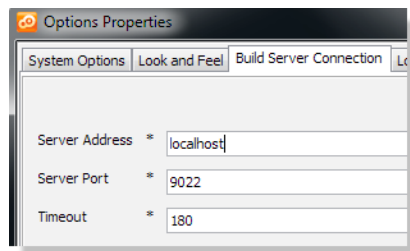



Hybrid Server Connection Details in edgeConnect Deployer

The Hybrid Server Connection details must be set in the edgeConnect Deployer. These should match the configuration of the Jenkins Master.

To Set Hybrid Server Connection Details

- 1 Open edgeConnect Deployer
- 2 Click **Tools** on the main menu, Select **Options**
- 3 Click **Build Server Connection** tab



- 4 Click **OK** to save settings

Add Deployment

A deployment setting file (.dsf) must be created for each project you wish to deploy. Once created, this file can be amended to accommodate any changes to the deployment which are required.

To Create a .dsf file

- 1 Open the **Deployer IDE**
- 2 Click **File** on the IDE standard toolbar
- 3 Click **New** from the dropdown menu
- 4 Click **Select**

The **Choose Project** dialog is displayed.

- 5 Select the project file you wish to deploy and click **Select**

The project file will have the .ifp extension.

The **Name** field will be populated with the name of the .ifp file. The **Name** field can also be completed manually.

- 6 Enter a description in the **Comment** field if required

This field should be used to enter a meaningful description of the destination of the deployment for future reference.

- 7 Click **OK**
A new deployment file is created.

Deployment Configuration

Once you have created the .dsf, you will need to configure the deployment by:

- Setting up Profiles
- Defining Messages



- Setting up Versioning Information

Profiles

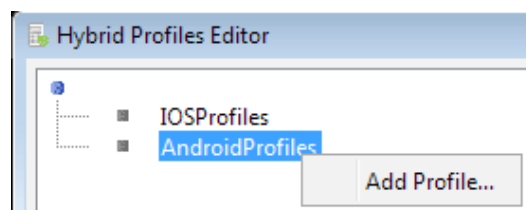
Profiles supply information about the location of the application, its certificate details and how it will operate on the target platform. Some of the profile information is platform specific. They can be selected from the Hybrid Build Wizard.

To Add a Profile

- 1 Open the **.dsf file** in edgeConnect Deployer
- 2 Click **Window** on the Deployer toolbar
- 3 Click **Hybrid Profiles Editor** from the dropdown menu

The Hybrid Profiles Editor is displayed. You can add four types of profile specific to the deployment platform:

- Android
 - iOS
 - Windows Phone 8
 - Windows
- 4 Right click the required root e.g. Android or iOS, click **Add Profile**



The Add Profile dialog is displayed. It contains three tabs:

- General
- Certificates
- platform specific e.g. Android/iOS specific

The latter two tabs are platform specific. The details contained on the tabs are described below.

- 5 Complete the appropriate fields
- 6 Click **OK** to save the profile



Common Settings tab

Name	Description
Profile Name	A meaningful name for the profile
Development Mode	Check to allow modification of edgeConnect Server URL, Update Servlet URL, HTML Resources Versions. Useful for testing and pre-sales.
Application Title	The title of the application as it will appear when published on Mac App store or Google Play
Application Orientation	Choose from dropdown. Recommended "unspecified" unless landscape or portrait is required. Unspecified allows the device to switch between landscape and portrait as its orientation changes.
OS Versions Supported	The minimum version of the supported target operating system. Currently read-only.
Jailbreak Detection	Check to detect if the device is jail-broken. Usage on this type of device can be perceived as a security risk - check to prevent functioning on such a device.
Resources Folder	Folder where the HTML Resources are located. They will be copied to the appropriate platform specific location. Allows customisation of icon and splash screen.
Destination Folder	Folder where the hybrid builds results, i.e. generated shell and build log, will be stored.
edgeConnect Application URL	The URL of the server application. This should contain /servletcontroller, otherwise the application will not function correctly. Format: http(s)://<ip>:<port>/<context>/<servlet controller name> e.g. http://localhost:8181/HybridApp/servletcontroller)
Update Servlet URL	The URL of the update servlet. This servlet controls the versioning of the HTML resources. Format: http(s)://<ip>:<port>/<context>/<mobile update link> \n e.g. http://localhost:8181/HybridApp/mobileUpdate)

Default Locations

The default locations are relative paths to the current project (resources folders) or installation (destination folders).

The Profile resources folder default value begins with `$$PROJECTHOME$`.

The Profile destination folder default value begins with `$$DEPLOY_HOME$`.

Both these values are replaced by the corresponding absolute path in subsequent selections and/or user inputs.

If a Profile destination folder points to a location inside a project (i.e. inside `$$PROJECTHOME$`), a non-blocking warning is displayed.

Certificates tab

The certificates tabs are slightly different for each platform and are prefilled with default values. Note that there is no certificates tab for Windows Phone 8.

Android certificate

The certificate is typically self-signed and is used for identifying the developer and for ensuring that the application has not been tampered with. It is fine to use the default certificate information supplied, but a new one is needed for publishing on Google Play.

For more information, see <http://developer.android.com/tools/publishing/app-signing.html>.



IOS Certificate

- **Certificate file:** this is a password protected .p12 file. Only Developer Program members can request signed certificates issued by Apple.
- **Certificate password:** the password of the .p12 certificate file
- **Certificate Name:** the certificate name as it appears in the Keychain Access
- **Provisioning profile:** created and downloaded from the Apple portal. It must match the certificate and the application id.
- **Package:** the bundle id of the application. It uniquely identifies the application on the AppStore and is represented by inverse DNS notation i.e. com.companyName.appName

Android Specific tab

Name	Mandatory	Description
Package Name	✓	A unique name for the package of the application. This should be changed to correspond to the organisation according to conventions found here: http://developer.android.com/tools/publishing/app-signing.html
Applicaition Version Name	✓	The name of the version of the application as it will be displayed in the Google Play store. This is in the format "1.0.0".
Application Version Code	✓	This is the actual version of the application. It is used by Google Play to notify the user that a new version is available. This should be an integer and should be incremented when deploying to the Google Play store.
Permission Options	✗	The permissions required by the application on the mobile device. These are presented to the user when installing the application. They are to be used in conjunction with cordova (phonegap) plugins.

iOS Specific tab

Name	Mandatory	Description
Build Type	✓	AdHoc: for internal distribution as a self-installing .ipa package Distribution: generates a zip package of the xcode archive that can be packed and uploaded to AppStore.
Application Version Name	✓	The App Store version of the application.
Device Configuration	✓	Specifies the types of devices that the app will run on e.g. phone, tablet, phone and tablet.
Run the App in Background	✗	If unchecked, the app will be killed when sent to background.
iOS Status Bar	✓	In iOS7, the screen overlaps the status bar and therefore it needs to be matched with the app background manually. Choose from: light(white), dark or hidden.
iTunes Link	✓	iTunes Store URL
Uses Push Notifications	✗	Push notifications will not be allowed if this box is not checked



Windows Phone 8 Specific tab

Name	Mandatory	Description
Company Name	✓	The name of the company distributing the application
Application Description	✓	A short description of the application.
Copyright	✓	The copyright information to include.
Trademark	✓	The trademark information to include.
Application GUID	✓	A new GUID can be generated by clicking on "Generate new GUID" button. The GUID uniquely identifies an installed application. If you need 2 versions of the same app installed in parallel, then you must generate a new GUID.
Manifest GUID	✓	A new one can be generated by clicking on "Generate new GUID" button. The manifest GUID uniquely identifies an installed application in the windows appstore. If you need 2 versions of the same app published in parallel, then you must generate a new GUID.
Application Version	✓	Format <major>.<minor>.<patch>
Permission Options		The permissions required by the application. These are presented to the user when installing the application and are used in conjunction with native device functionality

Windows Specific tab

Name	Mandatory	Description
Company Name	✓	The name of the company distributing the application
Application Description	✓	A short description of the application.
Copyright	✓	The copyright information to include.
Trademark	✓	The trademark information to include.
Application Version	✓	Format <major>.<minor>.<patch>
Manifest GUID	✓	This field is shown only when "development mode" is checked

Note that there are no Permission Options for this profile. This is because the Windows app does not allow access to the native features of the device through cordova plugins, like contacts, camera, accelerometer, because cordova is not available for this approach.

Manifest GUID

This field is only shown when "development mode" is checked. It is not required for publishing to the Windows Store because only the GUID generated by the store should be used in this case.

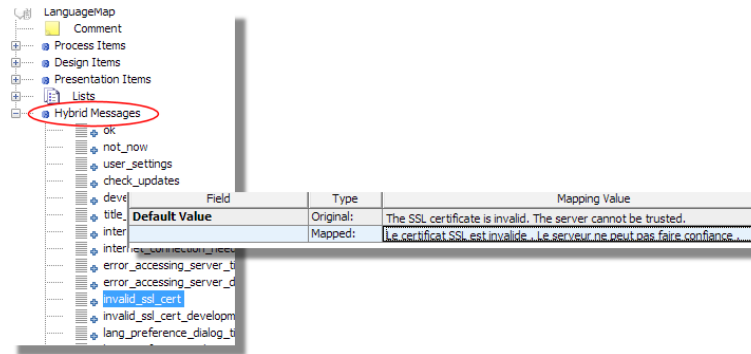
During development, this allows you to install more than one app on the same machine. The manifest GUID uniquely identifies an app in development mode. If you need two versions of the same app installed in parallel on one development machine, then you must generate a new GUID (click Generate New GUID button).



Messages

A SmartHybrid app has a number of messages that are using under certain conditions like when a phone has no network or for error conditions. The deployer allows you to change the default text and within the language maps you can setup specific language variants.

The Message Editor allows you to change the message text for messages in the shell applications. To open the Message Editor click **Window -> Hybrid Messages Editor** on the Deployer toolbar. If you wish to display these messages in other languages, you need to create a language map for the solution and edit the values using the Language Editor IDE.

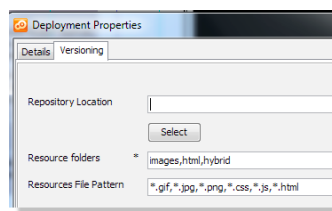


Versioning

The versioning information is part of the project properties. This tab will only be displayed when deploying a hybrid project. You will need to enter this information before deploying the project. You will be prompted to enter this information in the Deployment Wizard if you have not already done so.

To Add Versioning Information

- 1 Open the **.dsf** file in edgeConnect Deployer
- 2 Click **Deployment** on the Deployer toolbar, select **Properties** from the drop down menu
- 3 Click **Versioning** tab



- 4 Enter versioning information
 - **Repository Location:** The location for the versioning repository (a git repository). This will be used to determine what HTML Resources have been changed between releases.
 - **Resource Folders:** Folders in the project which contain HTML resources and should be searched for any changes.
 - **Resource File Pattern:** The extensions for any HTML resource files.
- 5 Click **OK** to save changes.

Deployment Wizard

Running the Deployment Wizard is the final part of the deployment process which brings together the edgeConnect solution, the profiles and the shell application to produce the hybrid application.



To Deploy your Project

- 1 Open the **Deployer IDE**
- 2 Click **Deployment** on the main toolbar
- 3 Select **Deploy Project**

The deployment wizard introduction is displayed.

- 4 Click **Next**

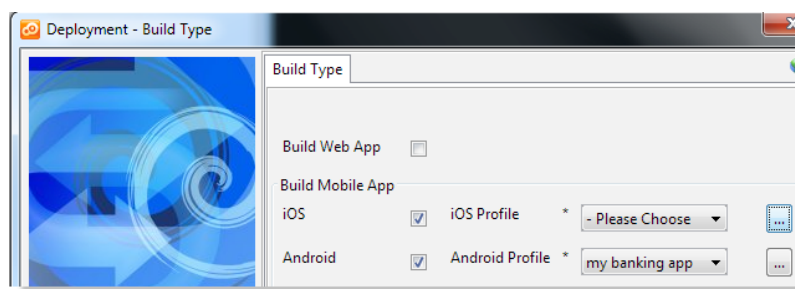
The project dependencies are displayed.

- 5 Click **Next**

The build type details need to be completed.

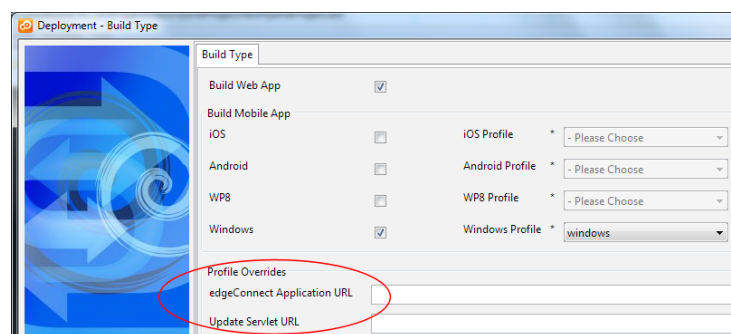
- 6 **Build Mobile App**: check Android, iOS or both.
- 7 Select the appropriate profile from the dropdown list

If you have not created an appropriate profile, the list will be empty. You will need to exit the Deployment Wizard and create a profile for your solution using the Hybrid Profiles Editor as described previously in this document.

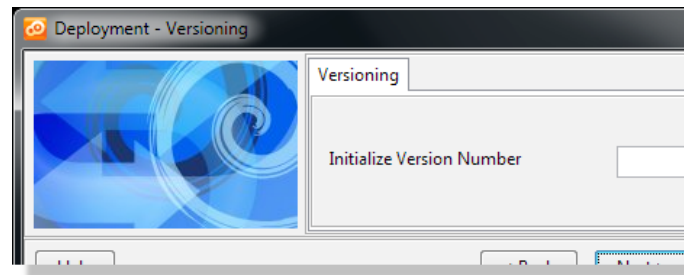


Note that you can edit the profile you have selected by clicking the ellipsis button to the right of the selection.

Overrides for edgeConnect Application URL and Update Servlet URL are available if at least one hybrid app build is selected. If any URL is set in these override fields, it will be used instead of other URLs from the selected profiles.

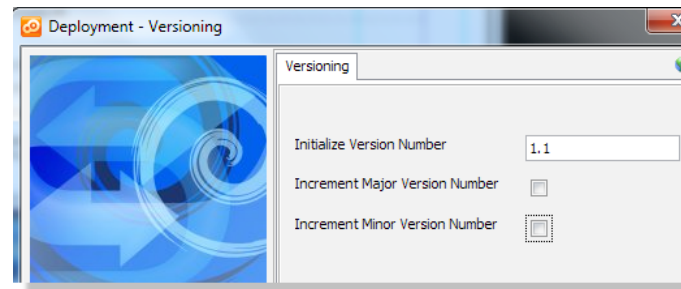


- 8 Click **Next**
- 9 Enter the version number for the HTML Resources. This should be in n.n format. Versioning information is optional.



10 Click **Next**

Additional fields are displayed if versioning information has been entered.



- 11 Complete these fields with the information as described in the table below. Those fields marked as mandatory are only required if the Initial Version Number is entered.

Name	Mandatory	Description
Initialize Version Number	X	The initial value for the version number
Increment Major Version Number	X	Check to increment the major version number. Incrementing the major version number will invite the mobile user to update the application through the app store. The check is performed at application startup.
Increment Minor Version Number	X	Check to increment the minor version number. Incrementing the minor version number will invite the mobile user to download the HTML resources which have been updated. The check is performed at application startup.

When the project is successfully deployed, you can check the output of versioning which is contained in the folder "update_log" in the edgeConnect WAR file. There will be a .json file for each version which contains information about the files that have changed between versions.

For example:

```
{
  "filesToUpdate": [
    "html/css/Default.css",
    "html/css/af.ui.css"
  ],
  "majorVersionUpdate": false,
  "serverVersion": "1.4",
  "updateSize": "74192B"
}
```




In the above example, the files “Default.css” and “af.ui.css” have changed. This means that the shell application will download these files upon application start-up.

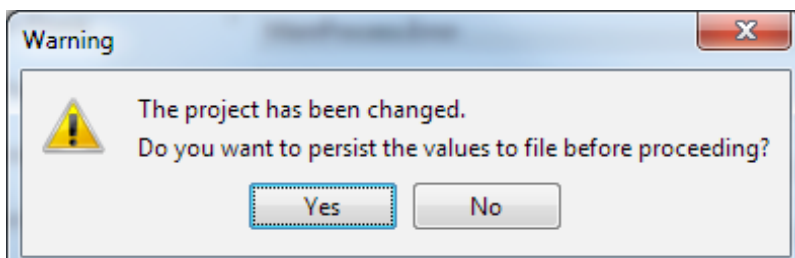
12 Click **Next**

13 Specify **Error Phases** and **Presentation** to use when generating the pages.

The SSL and Jailbreak error phases are not mandatory.

14 Click **Finish**

If you cancel the deployment before it has finished and any field values have changed, you will have the opportunity to save the changes via a yes/no confirmation prompt.



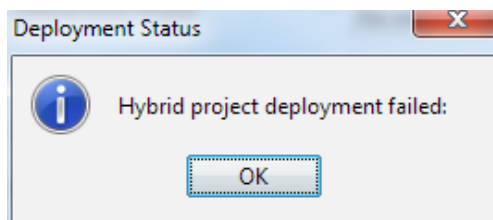
Deployment Results

A build can take up to 3 minutes. Once completed, the outputs are:

- A message inside the Deployer (see error messages)
- Packed Applications inside the appropriate destination folders
 - %DEPLOYMENT_LOCATION%/APPLICATION_NAME%.apk
 - %DEPLOYMENT_LOCATION%/APPLICATION_NAME%.ipa
 - %DEPLOYMENT_LOCATION%/APPLICATION_NAME%.zip
- Log files
 - %DEPLOYMENT_LOCATION%/hybrid_deployment.log
 - %INSTALL%/RTE/Deployments/hybrid_deployment_%project_name%.log
 - %DEPLOYMENT_LOCATION%/build_%platform%.log
(%platform% = android and/or ios)

Deployment Error Messages

If you see the message below after your deployment has completed, you will need to find the error message and fix the problem accordingly.





The table below shows possible deployer error messages, the related module and remedial action to be taken in the first instance.

Code	Message	Related Module	Action
	Build Failed	Mobile platform	Examine log files
	Failed to connect to server	Jenkins master	Verify master connection
	JSON parser error	Jenkins client	
	Build results not found	Jenkins master	
	Android slave offline	Jenkins slave	Verify slave connection
	IOS Slave offline	Jenkins slave	Verify slave connection
	Failed to retrieve build information	Jenkins master	
	Timed out	Jenkins master	Increase timeout
	Build aborted	Jenkins client	
	File not found on server	Mobile Platform	Examine log files



Publishing Hybrid Solutions

When publishing a hybrid solution you will need some or all of the following resources:

- Publisher account with appropriate platform
- Application icon(s) in appropriate sizes
- Splashscreen - shown on application startup
- Screen shots, application meta-data

It is important to spend some time thinking about the appropriate screenshots to use, as these are often the only thing that a customer can use to decide whether to purchase or download your application or not.

- Certificate
- tested edgeConnect hybrid application, deployed with correct publishing profile

The way they are configured differs for the platform you are publishing to. Further details for

- Android
- iOS
- Windows Phone 8

are given below.

Android

Publisher Account

In order to publish an application on the Google Play store, you will need to have a Google Play publisher account.

See <https://developer.android.com/distribute/googleplay/start.html> for a guide to publishing on Google Play.

Icons

- The icon filename is **android_icon.png**.

You need to make sure that your application ships with the correct sizes of the artwork.

See <http://developer.android.com/design/style/iconography.html> for an overview on what icon sizes to offer.

Splashscreen

The splashscreen is shown on application startup. The splashscreen is the file named "android_splash.png". It is recommended that you use the 9-patch image format, a stretchable bitmap image because this can be stretched without distortions.

See http://developer.android.com/guide/practices/screens_support.html for an overview of what image sizes you can provide for the splashscreen. For further explanation of 9-patch, see <http://developer.android.com/guide/topics/graphics/2d-graphics.html#nine-patch>.

Assets needed for publishing

Screenshots

Each application can have between two and eight screenshots for the Google Play store listing. The **Store Listing** page in the Google Play Developer Console offers comprehensive information on this.

Metadata

Before you submit your application, it is a good idea to have your application's metadata at hand. This includes:

- application name
- application type,



- category
- content rating
- short description
- full descriptions
- website, email, phone and privacy policy

The **Store Listing** page in the Google Play Developer Console offers comprehensive information on this.

Signing

Google **requires** that the apk file is signed. You should use the same certificate throughout the expected lifespan of your application. You can even sign all of your apps with the same certificate. There is no need to buy a certificate, a self-signed certificate can be used for this purpose.

See <http://developer.android.com/tools/publishing/app-signing.html#signing-manually>, paragraph “Signing Considerations” for further information.

edgeConnect Hybrid is supplied with a self-signed certificate in the location `PROJECT_HOME%\c\hybrid\android\certificates\keystore`. However, this can only be used for testing your application and not for publication.

To generate a certificate for publication you can use the keytool utility that comes with JDK. For a windows machine, for example, you can use the batch file:

```
set JAVA_HOME=D:\java\jdk\jdk1.6.0_31
%JAVA_HOME%\bin\keytool -genkey -v -keystore keystore_name -alias alias_name -keyalg RSA -
keysize 2048 -validity 10000
```

In the example above:

- **keystore-name** - the name for the generated file
- **alias_name** - the alias for the certificate
- **validity** - 10000 days. The certificate should not expire too soon.

This launches an interactive command, you will be asked for details about you and your organization and will need to supply passwords for the certificate and alias. Follow the steps to obtain a functional self-signed certificate for publishing to the Google Play store.

Deploying

When you deploy your hybrid solution in the edgeConnect Deployer, it is important to make sure you use the appropriate **publishing** settings in the Android profile:

- In the **General** tab, make sure that you **uncheck** “Development Mode”
- In the **Certificates** tab, insert the details of the self-signed certificate you are using
- In the **Android Specific** tab, make sure that the package name is unique in the Google Play store and that it refers to your organization.
- In the **Android Specific** tab, set the version code and version name. When updating the app, you must increment the version code, and is recommended that you increment also the version name.
- In the **Android Specific** tab, only select permissions which are required by your application. Unnecessary permissions may deter the user from installing the application.

In the deployment wizard, in the **Versioning** screen, increment the minor version number if anything has changed in the html resources, usually required every time. Increment the major version only when deploying with a new release of edgeConnect and if Temenos advises to do so.



Publishing

To Publish with Google Play Store

- 1 Login in to the Google Play Developer Console
- 2 Add a new application

Enter the default language and application title.

- 3 Click **Upload apk**
- 4 Prepare Store Listing

Add a

- short description
- long description
- icons for different resolutions
- screenshots
- a feature graphic
- a promo graphic.

All these artefacts are needed for displaying the app in the store.

- 5 Complete the pricing and distribution
- 6 Set optional “In-app Products” and “Services and APIs”.
- 7 Publish the application

You can either upload for production or for Beta or Alpha testing. Within a few hours, the application will be approved and available in the Play Store.

iOS

You will require a Mac with XCode installed.

Publisher Account

In order to publish an application on the App Store, you will need to be a registered iOS developer. This means that you have enrolled in Apple’s iOS Developer Program. You do this by visiting <https://developer.apple.com/programs/ios> and clicking the **Enroll Now** button.



Icons

You need to make sure that your application ships with the correct sizes of the artwork.

See <https://developer.apple.com/library/IOs/documentation/UserExperience/Conceptual/MobileHIG/IconMatrix.html> for an overview on what icon sizes to offer.



Assets needed for publishing

Screenshots

Each application can have up to five screenshots and you must provide at least one. If you are developing a universal application, then you need to provide separate screenshots for iPhone/iPod Touch and iPad/iPad Mini. In addition, you can optionally include separate screenshots for each screen size.

Metadata

Before you submit your application, it is a good idea to have your application's metadata at hand. This includes:

- application name
- version number,
- primary (and an optional secondary) category
- concise description
- keywords
- support URL.

If you are submitting an update, then you can also provide information for the **What's new in this Version** section.

If your application require users to sign in, you need to provide Apple with a test or demo account to make sure that the review team can immediately sign in and use your application without first having to sign up for an account.

Deploying

When you deploy your hybrid solution in the edgeConnect Deployer, it is important to make sure you use the appropriate **publishing** settings in the iOS profile:

- In the **General** tab, make sure that you **uncheck** "Development mode"
- In the **Certificates** tab, insert the **iOS distribution certificate** you created and downloaded from your Apple developer account. Make sure that you first install the iOS distribution certificate on your Mac. For the edgeConnect deployment, create a clone of your certificate by exporting it from the KeychainAccess utility of your Mac.
- In the **Certificates** tab, specify the **certificate name** and the **password** you set up when exporting the iOS distribution certificate.
- In the **Certificates** tab, insert the **distribution provisioning profile** that matches the iOS distribution profile
- In the **Certificates** tab, make sure that the **package name** is unique in the App Store and that it refers to your organization
- In the **iOS Specific** tab, in the Build Type field select **Distribution**
- In the **iOS Specific** tab, set the **Application Version Name**. When you update the application in the App Store you must increment Application Version Name. It is recommended that the Application Version Name matches the version number of the iTunes Connect record you created.

In the deployment wizard, in the **Versioning** screen, increment the minor version number if anything has changed in the html resources, usually required every time. Increment the major version only when deploying with a new release of edgeConnect and if Temenos advises so.

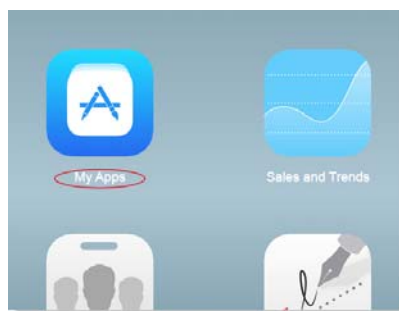
Publishing

You can now validate and submit an application using Xcode, for example.

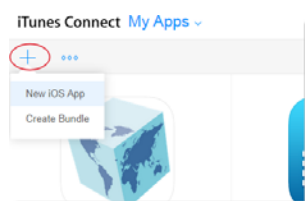
First, however, you need to create your application in iTunes Connect.



- 1 Login in to iTunes Connect with your iOS developer account
- 2 Click **MyApps**



- 3 Click the + sign in the top left



- 4 Select **New iOS App**, and fill out the form.

Basic information

- **App Name**, which needs to be unique, is the name of your application as it will appear in the App Store. This can be different than the name that is displayed below your application icon on the home screen, but it is recommended to choose the same name.
- **SKU Number** is a unique string that identifies your application.
- **Bundle ID** of your application.

This means selecting the (wildcard or explicit) App ID that you created earlier from the drop down menu.

Price and availability

Apple works with price tiers. This is so that you don't have to specify a price for each country that Apple operates in. You can also specify in which stores your application should - or shouldn't - be available. The information that you enter in this step can be modified once your application is live in the App Store i.e. you can change the price and availability of an application without having to submit an update.

Rating

An application is given a rating based on your application's content and functionality. This rating is not only useful for telling users about your application's content and features, it is also used by the operating system for the parental control features. It is strongly recommended that you rate the application appropriately. Apple will reject your application if it does not agree with the rating that you have set

Upload Binary

To Submit your Application

- 1 Open the **archive** of your application in XCode

Xcode's Organizer should automatically open and show you your archive
- 2 Select the archive from the list and click the **Distribute...**
- 3 Select **Submit to the iOS App Store**



After entering your iOS developer account credentials and selecting the **Application** and **Code Signing Identity**, the application binary is uploaded to Apple's servers. During this process, your application is also validated. If an error occurs during the validation, the submission process will fail. The validation process is very useful as it will tell you if there is something wrong with your application binary that would otherwise result in a rejection by the App Store review team.

If the submission process proceeds without problems, the application's status will change to **Waiting for Review**. The time it takes Apple to review your application will vary, but typically takes several days.

Windows Phone 8

Publisher Account

In order to publish an app on the Windows Phone Store, you need to have a company account. See <http://msdn.microsoft.com/en-GB/library/windows/apps/jj863494.aspx> for more details about Windows account types and how to create an account.

A guide on publishing on Windows Phone Store can be found at <http://msdn.microsoft.com/en-us/library/windows/apps/jj206724%28v=vs.105%29.aspx>.

Icons

- The icon filename is **ApplicationIcon.png**.
- The icon should be a png file of **100 x 100 px**.

It is important to make sure that your application ships with the correct size of artwork.

Splashscreen

- The splashscreen filename is **SplashScreenImage.jpg**.
- The splashscreen should be a file of **768 x 1280 px**.

Windows Phone 8 will scale down this file for lower resolutions.

Tile Images

A Tile is an image that represents your app on the Start screen. All apps have at least one Tile, known as the default Tile, which displays on the Start screen when a customer pins your app from the App list. There are several tile templates that we could use for a Windows Phone app. The hybrid app uses the "Flip Template".

Below is the table with the tile images that are needed:

Tile name	File name in hybrid	File size
Small	SmallBackground.png	159 × 159 pixels
Medium	Background.png	336 × 336 pixels
Wide	WideBackground.png	691 × 336 pixels

Tile Name	Filename in hybrid	File Size
Small	SmallBackground.jpg	159x159px
Medium	Background.jpg	336x336px
Wide	WideBackground.jpg	691x336px



Assets needed for publishing

Screenshots

Each application can have up to 8 screenshots for the store listing. The “Upload and describe your package” page in the Windows Phone Developer Console offers comprehensive information on what resolutions are needed.

Metadata

This includes:

- application’s name
- application type
- category
- subcategory
- version
- description
- keywords.

See the Windows Phone Developer Console for further information.

Signing

No signing is required for Windows Phone. Microsoft will sign the code prior to deployment.

Deploying

When you deploy your hybrid solution in the edgeConnect Deployer, it is important to make sure you use the appropriate **publishing** settings in the Windows profile:

- In the **General** tab, make sure that you **uncheck** “Development Mode”
- In the **Windows Specific** tab, make sure that the “Application GUID” is unique. Press the “Generate new GUID” button to create a unique id. The Manifest GUID does not have to be unique for the store, Windows Phone Store will generate and manage this for you.
- In the **Windows Specific** tab, setting the Application Version is not needed for the store, you will have to manually specify it when deploying or updating.
- In the **Windows Specific** tab, only select permissions which are required by your application. Unnecessary permissions may deter the user from installing the application.

In the deployment wizard, in the **Versioning** screen, increment the minor version number if anything has changed in the html resources, usually required every time. Increment the major version only when deploying with a new release of edgeConnect and if Temenos advises to do so.

Publishing

Note that:

- you are publishing on the Windows Phone Store, not the Windows Store.
- you may be required to activate Silverlight in the browser.

To Publish with Windows Phone Store

- 1 Login in to the Windows Phone Developer Console <https://dev.windowsphone.com/en-us/dashboard>
- 2 Submit a new application

Enter the **App Info**.

- 3 Click **Save**
- 4 Continue to **Upload and describe your package**

Add a new package by uploading the xap file.



Add a

- version number
- description
- keywords
- icon
- screenshots in different resolutions

All these artefacts are needed for displaying the app in the store.

- 5 Click **Save**
- 6 Set optional “In-app Advertising” and “Pricing”.
- 7 Click **Review** and **Submit** in the application’s dashboard screen.
- 8 Click **Submit**

Within a few hours to a day, the application will be approved and available in the Windows Phone Store.

Windows

Publisher Account

In order to publish an app on the Windows Store, you need to have a Windows Store developer account. For more details about the Windows account types and how to create one see <https://msdn.microsoft.com/en-us/library/windows/apps/hh868184.aspx>. A guide on publishing on Windows Store can be found at <https://msdn.microsoft.com/en-us/library/windows/apps/jj657972.aspx>.

Customisable Resources

Icons

You need to make sure that your application ships with the correct sizes of the artwork. The store logo resources are the files:

- StoreLogo*.png.

The resources for the icons in the Windows 8.1 start menu are the files:

- SmallLogo*.png
- Logo*.png.

Splash screen

The splashscreen is shown on application startup. The splash screen resources are the files:

- SplashScreen*.png.

Assets needed for publishing

Screenshots

Each application can have multiple screenshots in .png format for the store listing. Use one of the following resolutions:

- 1366x768
- 768x1366.



It is important to spend some time thinking about the screenshots. Your application's screenshots are often the only thing that a customer can use to decide whether she purchases or downloads your application or not.

Metadata

Before you submit your application, it is a good idea to have your application's metadata at hand. This includes:

- application name
- application type
- category
- subcategory
- version
- description
- keywords.

The Windows Store Dashboard offers comprehensive information on this.

Associating the app with the Store

On the machine where the Jenkins master is installed, you will need to modify the Windows project in order to associate the project with the Windows Store. This assumes that you have a Windows Store developer account.

The Windows project is located in:

```
$INSTALL$\jenkins\home\workspace\shell\platforms\windowsUniversal
```

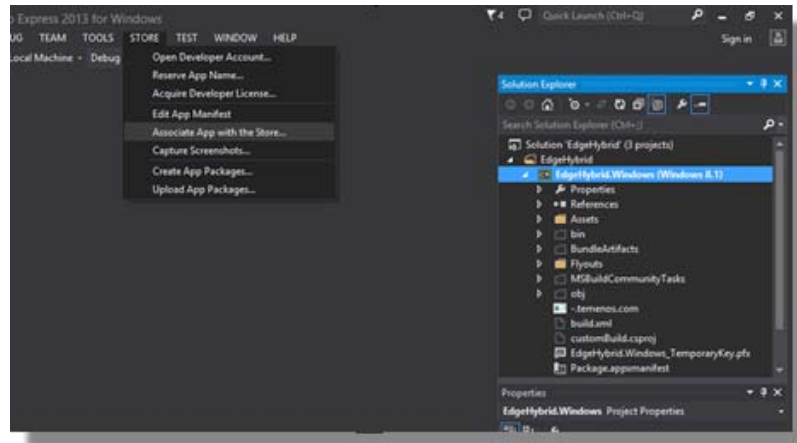
If one of the following is true:

- the Jenkins master is located on a different machine than the Jenkins Windows slave
- you cannot install "Visual Studio Express 2013"
(A guide on how to install "Visual Studio Express 2013" is the document
\$INSTALL\$\install\VisualStudioExpress2013.pdf)
- Visual Studio 2013 is not on the Jenkins master machine

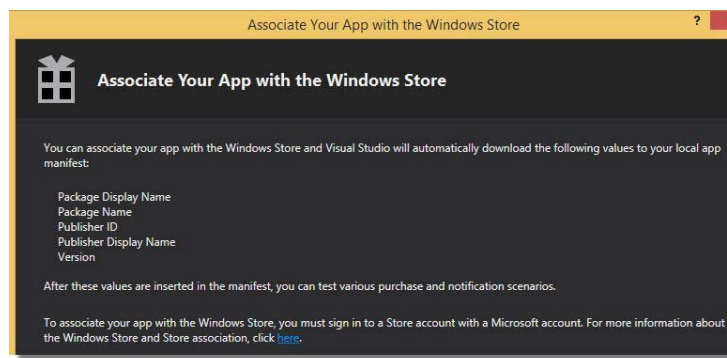
you can copy the Windows project on the Jenkins slave machine, alter it and copy it back to the Jenkins master.

To Associate the app with the Store

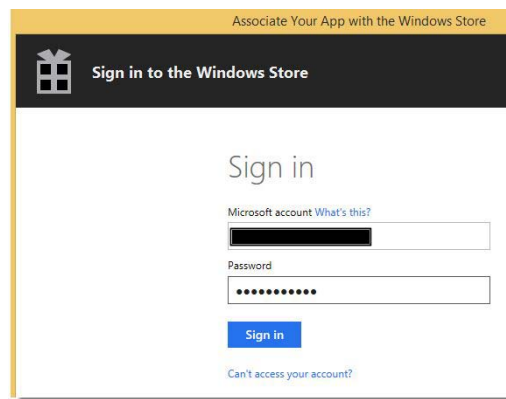
- 1 Open the project with Visual Studio Express 2013 by double-clicking the project file EdgeHybrid.sln
- 2 In the right pane, select the project "EdgeHybrid.Windows (Windows 8.1)" and select menu "Store->Associate App with the Store":



The Store Association wizard launches.



- 3 Click **Next**
- 4 Sign in with your developer account



- 5 Fill in the e-mail address associated with your account. Click **Next**.



You will receive an e-mail with a security code for account verification.

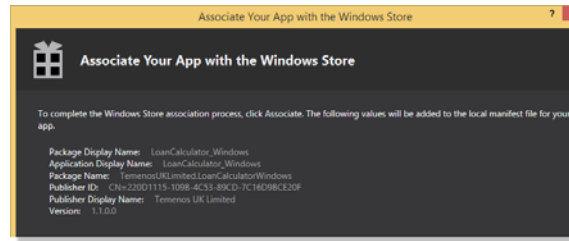
- 6 Enter the security code. Click **Submit**.

- 7 Select one of your existing apps in the Windows Store or reserve a new app name

App Name	Package Identity in the Windows Store
Loan Calculator POC	None
Loan Calculator POC	None
LoanCalculator_Windows	TemenosUK\imtest\LoanCalculatorWindk
Temenos Loan Calculator POC	None
Triple'A Plus	None

- 8 Click **Associate**

This will update the app's manifest and create a new signature key for publishing the app to the Windows Store.



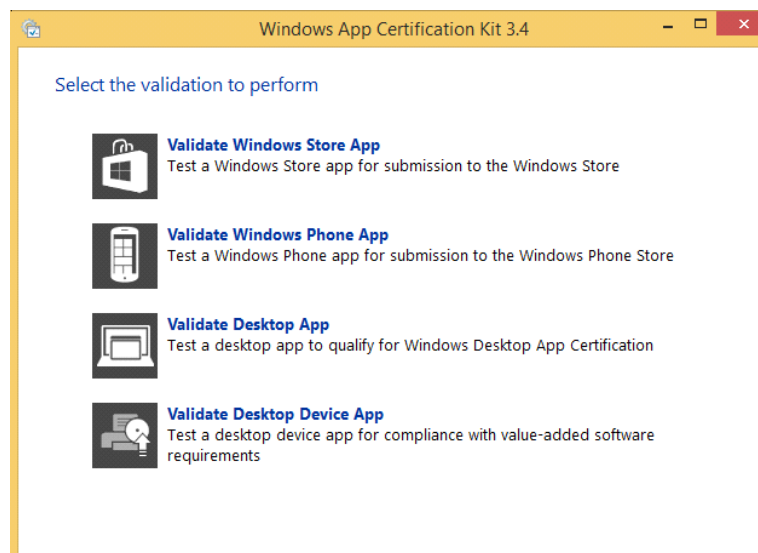
Verifying the app with the Windows App Certification Kit

Windows App Certification Kit is a tool that comes with Visual Studio 2013 (<https://msdn.microsoft.com/en-us/library/windows/apps/hh694081.aspx>). It is recommended you run a test with this tool; you will get immediate feedback on the problems, including detailed explanations, of what might cause the app to be rejected when trying to publish to the Windows Store. The Windows Store itself does not give many details on why an app does not pass certification.

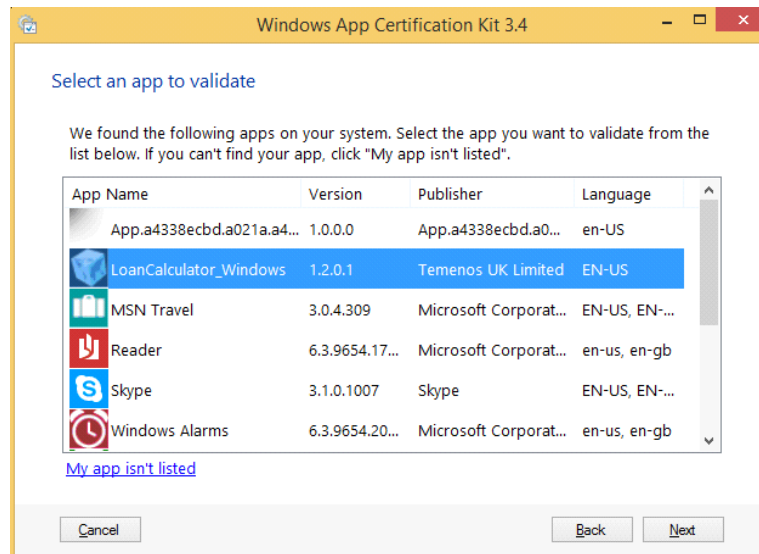
To Verify your app

Firstly, you will have to install the app on your Windows 8.1 machine by performing a “side load install”. This process is described in the hybrid guide.

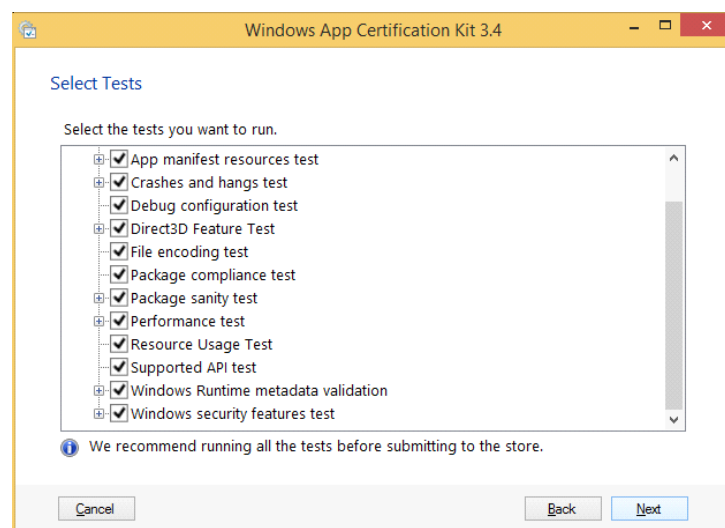
- 1 Start the “Windows App Cert Kit” program
- 2 Select **Validate Windows Store App**



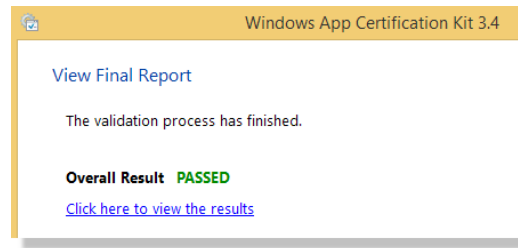
- 3 Select your app (which was previously side loaded)
Alternatively click on **My app isn't listed** and select your app, if you cannot see it.



4 Select everything and click **Next**



It will take a few minutes for the tests to run, and the app will launch multiple times. After the process is completed, you will be asked to save the report's xml file and in the final screen you can see if the validation was successful or not.



Click **Click here to view the results** to view the problems if the verification failed.

Signing the appxupload file

No signing is needed for Windows. Microsoft will sign the code prior to deployment.

Deployer IDE

The Windows deployment process is described above. For publishing, the following settings are important:

- In the “General” tab, make sure that you uncheck “Development Mode”
- In the “Windows Specific” tab, when setting the Application Version, make sure that this is a higher version than the version in the store (if the app was published before).

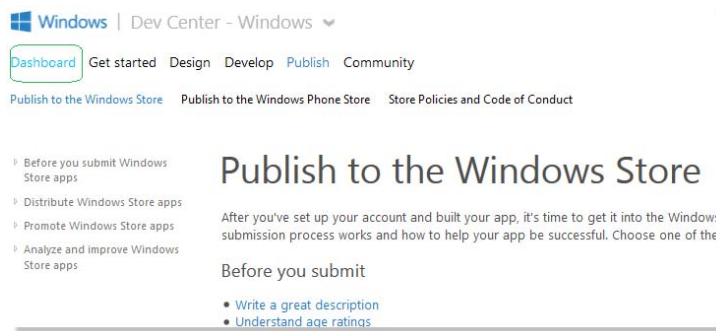
In the deployment wizard, in the “Versioning” screen, increment the minor version number if anything changed in the html resources, which usually happens every time. Increment the major version only when deploying with a new release of EdgeConnect and if advised to do so.

Publishing in the Windows Store

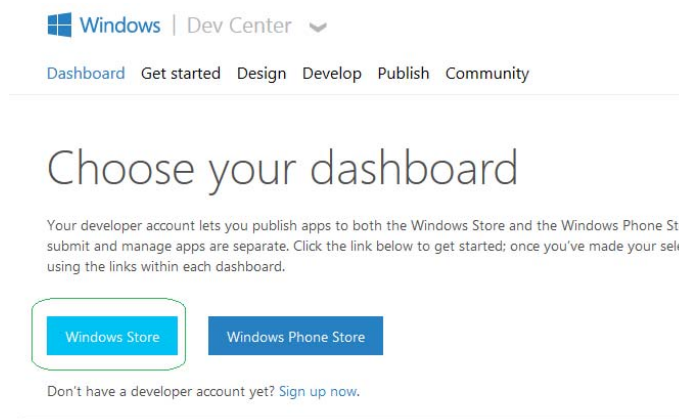
Make sure that you are publishing on the Windows Store, not on the Windows Phone Store. You might be required to activate Silverlight in the browser.

To Publish your app

- 1 Log in to the Windows Phone Developer Console <https://dev.windows.com/publish>, select **Dashboard** in the top menu

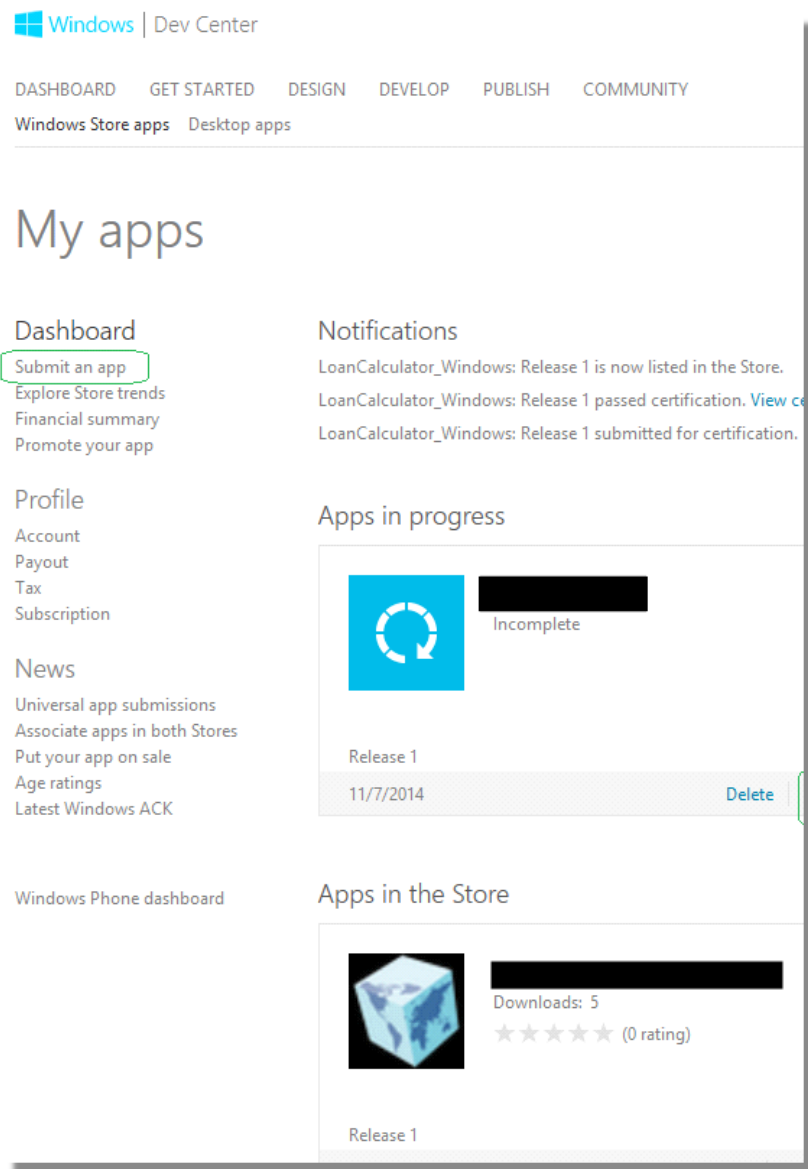


- 2 In the Dashboard screen, choose the **Windows Store** dashboard





3 Choose either to submit a new app or to edit an existing app



Note: In the following screens do not forget to click **Save** when they are completed.

- 4 **App Name** screen: select the existing or previously reserved app name
- 5 **Selling Details** screen: specify the pricing for the app and the countries in which the app will be available.
- 6 **Services** screen: click **Save**
- 7 **Age Rating** screen: specify the age ratings for your app.
- 8 **Cryptography** screen: specify that the app uses encryption (SSL), and that it uses encryption only for the purposes listed on the screen.



This app is considered to use encryption even if another entity performs the encryption, such as the operating system, an external library, a third-party product, or a cryptographic processor.

Does this app call, support, contain, or use cryptography or encryption? *

- ☒ Yes
☐ No

Is the cryptography or encryption limited to one or more of the tasks listed here? *

- Password encryption
- Copy protection
- Authentication
- Digital rights management
- Using digital signatures

If this app calls, supports, or uses cryptography or encryption for any task that is not in this list your answer to this question is No.

- ☒ Yes
☐ No

☒ I confirm that this app is widely distributable to all jurisdictions without government review, approval, license or technology-based restriction. *

- 9 Packages** screen: upload the file "WindowsHybrid_AnyCPU_bundle.appxupload", extracted from the zip with the hybrid build result for the windows profile.
- 10 Description** screen: add the required information for each language supported by your application.
- 11 Note to Testers** screen: add any notes relevant for testers.
- 12** Click on **Submit for Certification**

Certification will take one or two days and then the app will be published in the Windows Store if everything is in order.



Temenos Headquarters SA
18 Place des Philosophes
CH-1205 Geneva
Switzerland

Tel: +41 22 708 1150

Fax: +41 22 708 1160

www.temenos.com

TEMENOS™, TEMENOS T24™ and T24™ are registered trademarks of Temenos Headquarters SA
The Triple'A Plus™ and WealthManager™ trademarks belong to the Temenos Group of Companies

Information in this document is subject to change without notice.

No part of this document may be reproduced or transmitted in any form or by any means,
for any purpose, without the express written permission of TEMENOS HEADQUARTERS SA.

© 2010 Temenos Headquarters SA - all rights reserved.