



Developer Guide: Hybrid Apps

SAP Mobile Platform 2.3 SP04

DOCUMENT ID: DC01920-01-0234-01

LAST REVISED: March 2014

Copyright © 2014 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase trademarks can be viewed at the Sybase trademarks page at <http://www.sybase.com/detail?id=1011207>. Sybase and the marks listed are trademarks of Sybase, Inc. ® indicates registration in the United States of America.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

Java and all Java-based marks are trademarks or registered trademarks of Oracle and/or its affiliates in the U.S. and other countries.

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

Introduction to Developer Guide for Hybrid Apps	1
Documentation Roadmap for SAP Mobile Platform	1
Introduction to Developing Hybrid Apps With SAP	
Mobile Platform	3
Hybrid Web Container Architecture	3
Hybrid App Development Task Flow	6
Hybrid App Development Task Flow Using Third- Party Web Frameworks and MBOs	6
Hybrid App Development Task Flow With the Designer	7
Develop Hybrid Apps Using Third-party Web Frameworks	9
Develop MBO-based Hybrid Apps	9
Creating a Mobile Application Project	9
Developing a Mobile Business Object	10
Deploying a Mobile Application Project	11
MBO Examples	13
Generating JavaScript MBO Access API	23
Processing Responses From the Server	41
Error Handling	41
URL Parameters	42
Develop OData-based Hybrid Apps	43
Connect to an OData Source	43
Datajs OData Client Authentication in Hybrid Apps	45
Implementing Push	64
Enabling the Datajs Library on Windows Mobile	64
Hybrid Web Container and Hybrid App JavaScript APIs	65
anonymous namespace	68

- AppLog namespace84
- HttpsConnection namespace97
- hwc namespace106
- activationRequired method249
- clearCache method249
- clearCacheItem method249
- closeWorkflow method250
- expireCredentials method250
- getXMLHttpRequest method250
- guid method250
- logToWorkflow method250
- markAsActivated method251
- markAsProcessed method251
- processDataMessage method251
- processWorkflowMessage method252
- saveLoginCertificate method252
- saveLoginCredentials method252
- showCertificatePicker method253
- showUrlInBrowser method253
- Source code253
- MBO Access JavaScript API Samples594
- MediaCache Examples600
- Null Value Support600
- Calling the Hybrid Web Container602
- AttachmentViewer and Image Limitations604
- Package Hybrid Apps605
 - Packaging Hybrid Apps Using the Packaging Tool605
 - Packaging Hybrid Apps Manually607
- Deploying a Hybrid App Package with the Deploy Wizard629
- Develop a Hybrid App Using the Hybrid App Designer . 631**
 - Deploy the Hybrid App Package to SAP Mobile Server631

Generating Hybrid App Files and Deploying a Package	631
Hybrid App Patterns	632
Online Lookup	634
Server Notification	639
Cached Data	644
Hybrid App Package Customization	652
Customizing Generated Code	653
Adding Local Resources to a Hybrid App Project	653
Generated Hybrid App Files	654
Reference	661
Using Third-Party JavaScript Files	679
Repackaging Hybrid App Package Files	679
Common Customizations	680
Security	684
Credentials	684
Configuring the Hybrid App to Use Credentials	688
Content Security on Devices	694
Localization and Internationalization	700
Localization Limitations	701
Localizing a Hybrid App Package	701
Hybrid App Package Internationalization	707
Internationalization on the Device	709
Test Hybrid App Packages	710
Testing Server-Initiated Hybrid App Packages	711
Launching a Server-initiated Hybrid App on the Device	712
Debugging Custom Code	712
Manage a Hybrid App Package	717
Registering or Reregistering Application Connections	717
Setting General Application Properties	719
Application ID and Template Guidelines	720
Enabling and Configuring the Notification Mailbox	721

Assigning and Unassigning a Hybrid App to an Application Connection	722
Activating the Hybrid App	722
Configuring Context Variables for Hybrid App Packages	723
Changing Hard Coded User Credentials	724
Adding a Certificate File to the Hybrid App Package	725
End to End Trace and Performance	725
Enabling the Performance Agent on the Device	726
Tracing Application Connections	726
Build a Customized Hybrid Web Container Using the Provided Source Code	729
Building the Android Hybrid Web Container Using the Provided Source Code	729
Building the Android Hybrid Web Container Outside of Eclipse	730
Building the BlackBerry Hybrid Web Container Using the Provided Source Code	731
Supplying a Signing Key	731
Building the iOS Hybrid Web Container Using the Provided Source Code	732
Building the Windows Mobile Hybrid Web Container Using the Provided Source Code	733
Install and Configure the Hybrid Web Container On the Device	735
Preparing Android Devices for the Hybrid Web Container	735
Installing the Hybrid Web Container on Android Devices	735
Configuring the Android Emulator	735
Preparing BlackBerry Devices for the Hybrid Web Container	738

Installing the Hybrid Web Container on BlackBerry Devices Over the Air	739
Enabling Hybrid Web Container Message Notification	739
Configuring the BlackBerry Simulator for Hybrid Web Containers	740
Preparing iOS Devices for the Hybrid Web Container	740
Installing the Hybrid Web Container on the iOS Device	741
Preparing Windows Mobile Devices for the Hybrid Web Container	742
Installing the Hybrid Web Container on Windows Mobile Devices	742
Installing Microsoft Synchronization Software ...	743
Installing the Hybrid Web Container on the Windows Mobile Emulator	744
Configure Connection Settings on the Device	746
Configuring Android Connection Settings	746
Configuring BlackBerry Connection Settings	747
Configuring iOS Connection Settings	748
Configuring Windows Mobile Connection Settings	749
Install and Test Certificates on Simulators and Devices	751
Installing X.509 Certificates on Windows Mobile Devices and Emulators	751
Installing X.509 Certificates on Android Devices and Emulators	752
Installing X.509 Certificates on BlackBerry Simulators and Devices	753
Installing X.509 Certificates on iOS Devices	754
Uninstall the Hybrid Web Container from the Device ..	758
Removing the Hybrid Web Container From the BlackBerry Device	758

Hybrid Web Container Customization	761
Adding a Custom Icon for the Hybrid App Package	
Using the Packaging Tool	761
Manually Adding a Custom Icon to the	
Manifest.xml File	762
Changing the Hybrid App Package Icon	763
Android Hybrid Web Container Customization	764
Android Customization Touch Points	764
Testing Android Hybrid Web Containers	797
Upgrading the PhoneGap Library Used by the	
Android Hybrid Web Container	797
BlackBerry Hybrid Web Container Customization	798
BlackBerry Customization Touch Points	799
Upgrading the PhoneGap Library Used by the	
BlackBerry Hybrid Web Container	827
iOS Hybrid Web Container Customization	828
iOS Customization Touch Points	829
Upgrading the PhoneGap Library Used By the	
iOS Hybrid Web Container	843
Windows Mobile Hybrid Web Container	
Customization	848
Windows Mobile Customization Touch Points . . .	848
Look and Feel Customization of the Windows	
Mobile Hybrid Web Container	849
Default Behavior Customization of the Windows	
Mobile Hybrid Web Container	852
Packaging a CAB File	857
Prepackaged Hybrid Apps	857
Including a Prepackaged Hybrid App in the	
Android Hybrid Web Container	858
Including a Prepackaged Hybrid App in the	
BlackBerry Hybrid Web Container	858
Including a Prepackaged Hybrid App in the iOS	
Hybrid Web Container	859

Including a Prepackaged Hybrid App in the Windows Mobile Hybrid Web Container	860
Adding Native Device Functionality to the Hybrid Web Container	864
Supported JavaScript PhoneGap APIs	864
Implementing PhoneGap	874
PhoneGap Custom Plug-ins	875
Initializing the PhoneGap Library for the Windows Mobile Hybrid Web Container	886
PhoneGap Library Downgrade	886
Using the HTTPS Proxy Exposed by the PhoneGap Plugin	888
Hybrid App Configuration for Data Change Notification	893
Extending Data Change Notification to Hybrid Apps .	893
Non HTTP Authentication Hybrid App DCN Request .	895
Sending Hybrid App DCN to Users Regardless of Individual Security Configurations	896
Hybrid App DCN Request Response	896
Hybrid App DCN Design Approach and Sample Code	897
Comparing Hybrid App DCN With and Without Payload	897
Sample Java Function for Generating Hybrid App DCN	900
Index	903

Contents

Introduction to Developer Guide for Hybrid Apps

This developer guide provides information about using SAP® Mobile Platform features to create Hybrid App packages. The audience is Hybrid App developers.

This guide describes requirements for developing a Hybrid App package, how to generate Hybrid App package files, and how to deploy the Hybrid App to the server. It also provides information about how to customize the provided source code for Hybrid Web Containers.

Companion guides include:

- *SAP Mobile WorkSpace - Mobile Business Object Development*
- *SAP Mobile WorkSpace - Hybrid App Package Development*
- *System Administration*
- *SAP® Control Center for SAP Mobile Platform*
- *Tutorial: Hybrid App Package Development*
- *Troubleshooting*
- *Mobile Application Life Cycle*
- *Developer Guide: Migrating to SAP Mobile SDK*

Documentation Roadmap for SAP Mobile Platform

SAP® Mobile Platform documents are available for administrative and mobile development user roles. Some administrative documents are also used in the development and test environment; some documents are used by all users.

See *Documentation Roadmap* in *Fundamentals* for document descriptions by user role.

Check the Product Documentation Web site regularly for updates: <http://sybooks.sybase.com/sybooks/sybooks.xhtml?id=1289&c=firsttab&a=0&p=categories>, then navigate to the most current version.

Introduction to Developing Hybrid Apps With SAP Mobile Platform

A Hybrid App includes both business logic (the data itself and associated metadata that defines data flow and availability), and device-resident presentation and logic.

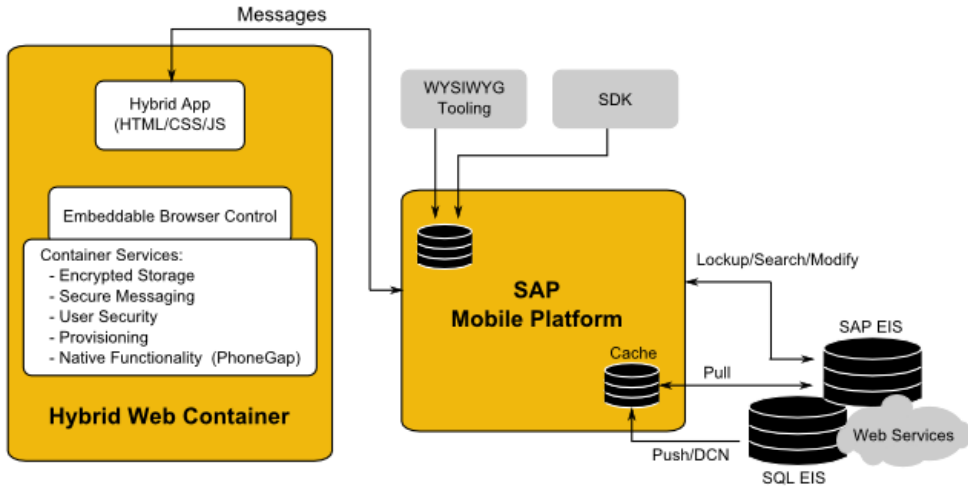
You can develop Hybrid Apps using third-party Web frameworks, enabling you to access gateway datasources through the Hybrid Web Container.

SAP Mobile Platform, development tools enable both aspects of Hybrid App development:

- The data aspects of the Hybrid App are called mobile business objects (MBO), and “MBO development” refers to defining object data models with back-end enterprise information system (EIS) connections, attributes, operations, and relationships. Hybrid Apps can reference one or more MBOs and can include load parameters, personalization, and error handling.
- Once you have developed MBOs and deployed them to SAP Mobile Server, develop device-resident presentation and logic for your Hybrid App. See *SAP Mobile WorkSpace - Mobile Business Object Development* for procedures and information about creating and deploying MBOs.
- A second data option is to access OData sources from your Hybrid Apps with the Dataajs library.
- OData sources and MBOs can be used together in a Hybrid App.

Hybrid Web Container Architecture

The Hybrid Web Container is the runtime on the device within which Hybrid Apps are executed.



Hybrid Web Container Customization

A Hybrid Web Container is a native application designed to process generic function calls from a Hybrid App. The Hybrid Web Container embeds a browser control supplied by the device OS, which allows you to build applications with simplicity of Web development but utilize the power of native device services. By using the Hybrid Web Container for each device type supported in a business mobility environment, you can create a single HTML5 application that performs advanced, device specific operations on all the different devices.

Hybrid App Development

The Hybrid Web Container supports workflow type applications, which are applications that participate in a lifecycle flow involving special notifications (modified in the Transform Queue), application flow, and finally submission of form data to matching server components (through the Response Queue).

The Hybrid Web Container also supports applications that do not participate in a workflow type process. In other words, applications that are not triggered by notifications (no Transform Queue), and that do not submit asynchronous “submit” responses through queuing (no Response Queue). These applications do not communicate with the server for data access, but use the messaging channel for deployment, provisioning, and application life cycle management.

Write Hybrid Apps in standards-based HTML5, JavaScript (the standard scripting language used to create Web applications), and Cascading Style Sheets (CSS). These are technologies familiar to web developers. This enables Web developers to incorporate open source frameworks and also select their preferred development environment, for example, Sencha and JQuery Mobile.

Hybrid App Designer

The Hybrid App Designer uses the Hybrid Web Container as the runtime for Hybrid App packages. The Hybrid App Designer included with SAP Mobile Platform is a tool that helps you design the user interface and test the flow of the business process for an application. Using the Hybrid App Designer allows you to develop application screens that can call on the create, update, and delete operations, as well as object queries, of a mobile business object.

Hybrid App package files are generated using the Hybrid App Package generation wizard in the Hybrid App designer. The generated Hybrid App package contains files that reference a mobile business object (MBO) package, an MBO in that package, and the operation or object query to call along with a mapping of which key values map to parameter values. The generated Hybrid App package's output is translated to HTML\CSS\JavaScript. The logic for accessing the data and navigating between screens is exposed as a JavaScript API.

The Hybrid App packages generated by the Hybrid App designer are not proprietary, they are identical to what would need to be produced when using other tools and Web application frameworks. Hybrid App Designer-generated packages use jQuery Mobile as their primary Web application framework on most platforms.

Deploy Hybrid App packages to SAP Mobile Server and assign to users using the Hybrid App Designer in Eclipse.

Generated Customization Files

The Hybrid Web Container uses HTML, JavaScript, and CSS Web technologies, which allow you to customize the generated files with JavaScript code.

- **HTML** – the generated files depend on the device platform. You can open these files with a third-party Web-development tool and modify them, but they are overwritten if generated from the Hybrid App deployment tool. The Hybrid App Designer also includes a HTMLView user interface element that can be placed on a screen, and in which custom HTML code can be inserted, which will be published in-line when the file is re-generated.
- **JavaScript** – the JavaScript API exposes customization points for navigation events, and allows access to data-access functions for requests and cached values. Customization of the HTML page should be executed using the embedded jQuery in these customization points. For example, execute jQuery logic to modify the toolbar in `customBeforeHybridAppLoad()`. You can add additional custom JavaScript files to the Hybrid App package in the Eclipse Workspace.

Note: In prior releases, JavaScript files for customization were automatically included in the generated Hybrid App HTML files. The JavaScript files are still added to the generated package, but no longer referenced in the HTML.

- **CSS** – the Hybrid Web Container uses a third-party CSS library, which enables you to modify the look-and-feel of the HTML page. The jQueryMobile CSS file is embedded as the default look-and-feel, which allows you to select from the variety of themes within the jQueryMobile framework, or use your own CSS rules for skinning pages and screen

elements. These can be device operating system-specific. You can also leverage existing CSS style rules from your own organization's Web standards.

The generated files are documented in the *Reference* section of this guide.

Management

You can deploy Hybrid App packages in Eclipse and manage them through the SAP® Control Center console. No device interaction is required from the administrator. Once a Hybrid App package is deployed into an existing installation, the administrator can configure the Hybrid App package and assign it to any active user in the system.

Offline Capabilities

Server-initiated notifications extract data from the backend and SAP Mobile Platform sends them to the client device. The client device does not need to be online at the time the notification is sent—the message is received as soon as the client device comes online. Submit actions on the client can also be sent while the device is offline. They will be sent to the server as soon as the device comes online. These notifications are made available offline for processing once they are delivered to the device.

Online Request actions only work when the device is online. The results of object queries run by these types of actions can be cached on the client so that the next time the same query is invoked with the same parameters it is able to get those results from the client-side cache without needing to go to the server. This is achieved by specifying a non-zero cache timeout for the action.

You can also store data locally (when the device is offline) using the `SUPStorage` JavaScript API.

Hybrid App Development Task Flow

This task flow describes task flows for the different Hybrid App development options.

Hybrid App Development Task Flow Using Third-Party Web Frameworks and MBOs

This describes the basic steps for developing Hybrid Apps that access MBO operations and object queries using a third-party tool, or by hand.

1. Define the data you want to use from your backend system and to expose through your Hybrid App, and the methods and operations to perform.
2. *Create a Mobile Application project* on page 9.
3. *Develop the Mobile Business Objects* on page 10.
4. *Deploy the Mobile Application Project* on page 9.
5. *Generate the JavaScript API* on page 23.

6. *Package the Hybrid App files.* on page 605
7. *In SAP Control Center, deploy the Hybrid App package to SAP Mobile Server.* on page 629

Hybrid App Development Task Flow With the Designer

Developing a Hybrid App includes these basic tasks.

1. Open or import a mobile application project with predefined mobile business objects (MBOs).
2. Deploy the Mobile Application Project:
 - b. On the Target Server page, select the server and connect to it.
 - c. On the Server Connection Mapping page, map the database connection profile to the server.
3. Create the application connection in SAP Control Center.

Note: This step is normally performed by the system administrator.

4. Use the Hybrid App Designer to create a new Hybrid App.

Note: Optionally, you can create a Hybrid App manually, however, using the Hybrid App Designer, automates many tasks and provides integration across different device platforms.

5. Use the Hybrid App Designer to generate screens by dragging and dropping MBOs and MBO operations from WorkSpace Navigator to the Flow Design page.
6. Create, delete, and edit screens, controls, menus, screen navigations, and so on.
7. Generate the Hybrid App files.
8. (Optional) Customize the generated `Custom.js` file.
9. (Optional) If you customize the Hybrid App files, re-generate and repackage the files.
10. Deploy the Hybrid App package to SAP Mobile Server.
11. Install and configure the Hybrid Web Container on the device or simulator.
12. In SAP Control Center, assign the Hybrid App to the device user.
13. On the device or simulator, run, test and debug the Hybrid App.

Note: See *SAP Mobile WorkSpace - Mobile Business Object Development* for procedures and information about creating and deploying MBOs.

Identify a Business Process for a Hybrid App That Uses a Lifecycle Flow

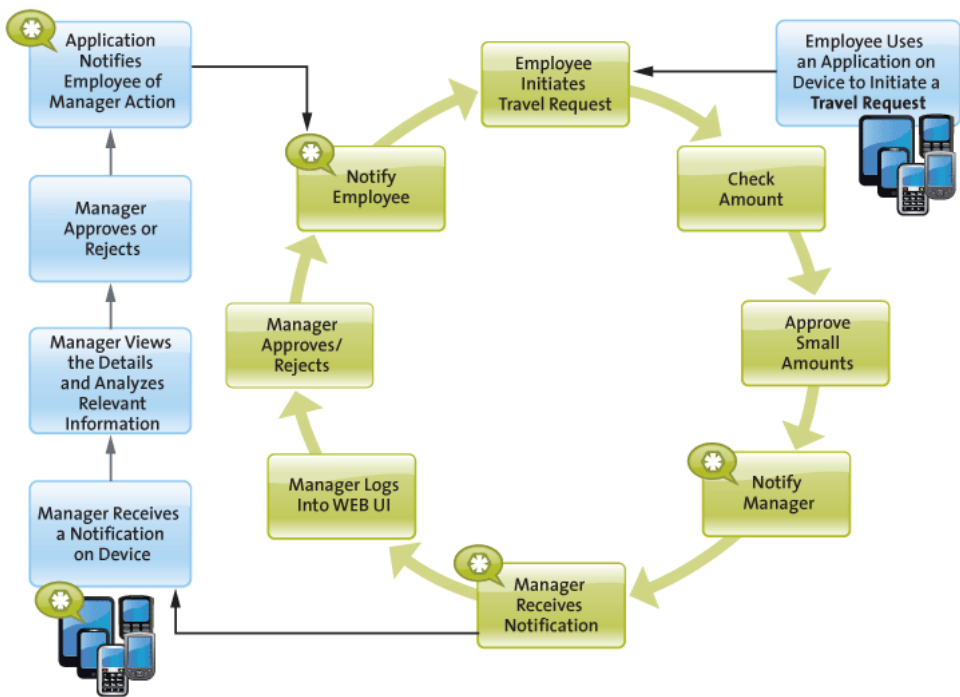
The first step in developing a Hybrid Apps that participate in a lifecycle flow involving special e-mail messages (modified in the Transform Queue), application flow, and finally submission of form data to matching server components (through the Response Queue) is identifying whether a Hybrid App can implement a decision point in a particular business process.

Hybrid Apps enable a decision step or triggering of a business process, essentially mobilizing a small decision window in a business process. While some business processes require a thick application with business logic and access to reference data, some others do not. Sometimes a

business process can be made mobile simply by providing the ability to capture a single "Yes" or "No" from a user, or by providing the ability to send data in structured form into the existing backend systems.

A typical Hybrid App allows creating, updating or deleting of data in a backend data source (EIS), either directly or through the SAP Mobile Server, and retrieving that data, then displaying that information in a decision step. A more complex Hybrid App could involve an application that uses online request menu items to invoke various create, update, or delete operations and/or object queries all in the same flow.

An example of a business process that would be a suitable Hybrid App would be the ability of an employee to use a mobile device to submit an expense report while out of office, or to report on their project activities, or to make a request for travel.



Develop Hybrid Apps Using Third-party Web Frameworks

Developing Hybrid Apps this way allows you to use a greater variety of application designs, from using different HTML formatting to using different Web application frameworks, and beyond.

Note: When writing your own HTML and JavaScript to create a Hybrid App package manually, there is one absolute requirement—you must implement the following JavaScript function:

```
function processDataMessage (incomingWorkflowMessage)
```

The Hybrid Web Container needs to call this function when online request processing is complete. The incoming message is an XML-formatted string.

Develop MBO-based Hybrid Apps

Develop Hybrid Apps using mobile business objects (MBO) to define object data models with back-end enterprise information system (EIS) connections, attributes, operations, and relationships.

A project in SAP Mobile WorkSpace must contain the MBOs to use in your application. See *SAP Mobile WorkSpace - Mobile Business Object Development*.

The JavaScript APIs in the Mobile SDK are located in `<SMP_HOME>|UnwiredPlatform|MobileSDK<version>|HybridApp|API`. It is split into two categories:

- **Container** – these APIs are fundamentally independent of the UI framework you choose to use (if any). There is no reference to screens. These APIs are considered mandatory when building your Hybrid App.
- **AppFramework** – these APIs are an optional add-on to the Container APIs that give you functionality to navigate between screens, represent the messages sent to and from the server in developer-friendly form, and bind the UI to and from those messages automatically. These APIs do make some assumptions about how your UI is constructed/manipulated, and those assumptions are not necessarily true for all UI frameworks, Sencha among them.

Creating a Mobile Application Project

A mobile application project is the container for the mobile business objects that forms the business logic of mobile applications.

You must create a mobile application project before you can create mobile business objects. See *Eclipse Basics* for information about projects.

1. Select **File > New > Mobile Application Project** from the main menu bar.
2. Enter a:
 - Name
 - Location (if other than the default).
3. Click **Finish**.

The Mobile Application Project is created and an empty Mobile Application Diagram opens.
4. (Optional) Modify the Mobile Application Project configuration properties by right-clicking the project in WorkSpace Navigator, selecting **Properties**, and selecting **Mobile Application Project Configuration**. When modifying the configuration properties, keep in mind that:
 - The default application ID and Display name are the same as the project. The description is "Default application ID".
 - Follow these guidelines when changing application ID, application name, display name, and description:
 - **ID** – less than 64 bytes, begin with an alphabetic character, followed by alphanumeric characters, a dot, or underscores, and not contain consecutive dots or underscores.
 - **Display name** – string, length less than 80 bytes.
 - **Description** – string, length less than 255 bytes.

All added applications must have a name (display name and description can be empty), but are assigned a name at runtime when the application is created.

Developing a Mobile Business Object

You can define attributes and operations of a mobile business object (MBO) without immediately binding them to a data source, define them from and bind them to a data source, or create an MBO that does not bind to a data source (local business object, or uses only DCN as the refresh mechanism).

Prerequisites

Before developing MBOs, understand the key concepts and principals described in *Understanding Fundamental Mobile Development Concepts*. Also, see the companion guide, *Mobile Data Models: Using Mobile Business Objects*, for a deeper understanding of how to build an efficient MBO model.

Task

The attributes and operations that define an MBO must be bound to a data source at some point in the development process, unless it is a local business object, or the MBOs data is to be loaded only through Data Change Notification (DCN). If you already have a connection to the data source through a connection profile, you can quickly generate attribute and operation bindings based on the data source. However, if you do not have access to the required data

source, you define the MBO, but bind your operations and attributes to the data source at a later point. The difference between the two development approaches is when you create and bind the attributes and operations:

- Create an MBO and bind to a data source immediately – includes two methods:
 1. Drag and drop the data source onto the Mobile Application Diagram, which launches the appropriate wizards and automatically creates bindings based on the selected data source.
 2. Create an MBO and its operations and attributes using the Mobile Application Diagram and palette that launches a set of wizards and allows you to bind them directly to a data source.
- Create an MBO and defer data source binding – create an MBO and its operations and attributes using the Mobile Application Diagram and palette that launches a set of wizards and allows you to bind the MBO to a data source at a later time. After you define the data source, you bind the MBO to it from the Properties view.
- Create an MBO using a DCN cache group policy without data source binding – when an MBO's CDB data is to be filled only through DCN, a data source binding is not necessary. In these cases, the MBO must reside in a cache group that uses the DCN policy.
- Create a local business object – create a local business object by clicking the local business object icon in the palette then click the object diagram. Local business objects can only run on the client and cannot be synchronized. It can contain attributes and operations that run on the device. For example, the local business object could be combined with other MBOs, where the local business object runs an object query against results returned by other MBOs.

Deploying a Mobile Application Project

Deploy a Mobile Application project directly to an SAP Mobile Server, and optionally create a reusable deployment profile.

To avoid errors or inconsistent behavior, client applications must be regenerated whenever a package has been redeployed. Restarting the client application is not sufficient to reset the client for a package that has been redeployed.

1. Right-click the Mobile Application project and select **Deploy Project**.

Alternatively, you can launch the deployment wizard, which automatically sets the SAP Mobile Server portion of the wizard, by dragging a Mobile Application project folder from Workspace Navigator and dropping it on the SAP Mobile Server in Enterprise Explorer to which you are deploying.

Note: As an option, you can press F9 when your cursor is in the Mobile Application Diagram to launch the Deployment wizard for the corresponding project. If a deployment profile exists for the project, F9 performs quick deployment of the project according to the profile.

2. Select a deployment mode (**Update**, **Replace**, or **Verify**), target version, Package name, and click **Next**.
3. Select the MBOs from each Synchronization Group to be deployed and click **Next**.

Note: If any selected MBOs contain errors, the **Next** and **Finish** buttons are disabled.

4. Create or add required JAR files for MBOs that use Resultset Filters or Custom Result Checkers and click **Next**.
5. Select a target server, click **Connect**, and select a Domain and Security Configuration for the deployment package and click **Next**. (Optional) If no SAP Mobile Server connection exists, click **Create** and define a connection profile for one to which you can connect and deploy the deployment package.
6. Deploy applications to SAP Mobile Server – select the applications to deploy to SAP Mobile Server. A unique Application ID identifies the application and uses the project name by default.

SAP Mobile WorkSpace lists not only local applications defined through the mobile application project's context menu **Properties > Mobile Application Project Configuration**, but all applications already assigned to the selected domain of the target server (available applications), whether those existing applications contain this current mobile application project or not. SAP Mobile WorkSpace validates the projects for:

- If the local and server applications are the same, but the display name or description differ, they display in the target applications list, but a validation error appears because the assigned application ID must be unique.
 - When deploying the project/package with "Replace" mode, if the project/package already exists in an available application that exists on the server, but that application is not selected as the target application, a warning indicates that the server will remove the project/package from the existing application.
 - If a local application is added to the target applications list, and a server application with the same ID but different display name/description is not assigned, a warning indicates that you can modify the display name/description of the existing sever application with that of the local application.
7. Map connection profile to server connections – you must map design-time connection profiles to server-side (runtime) enterprise information system (EIS) data sources referenced by the MBOs in the project. Deployment fails if the EIS data sources are not running and available to connect to. To map the connection profile to a server connection, select the connection profile from the list of available connection profiles then select the corresponding server connection to which it maps, or select **<New Server Connection...>** to create a new server connection.

Contact the system administrator in cases where your development environment permits access to systems that the SAP Mobile Server prohibits.

Note: You can also modify server connection properties (Web service connections only).

8. If a logical role is defined in your MBO, map logical roles to physical roles. If there are no logical roles defined, this page is skipped. Click **Next**.
9. (Optional) Specify the name and location for the new deployment profile. This is useful for troubleshooting MBO and deployment errors.
 - Save the deployment settings as a deployment profile – if you do not save your settings to a deployment profile, they are lost when you exit the Deploy wizard.
 - Enter or select the parent folder – by default, Deployment is the folder in which the deployment profile is saved.
 - File name – the name of this deployment profile. The deployment profile is assigned a `.deploy` extension.
10. Click **Finish** to deploy the project to the SAP Mobile Server's Packages folder.

MBO Examples

This section shows examples of how to implement different patterns and functionality. These are examples only. You must modify the procedures based on the actual MBOs, object queries, and parameters you are using.

Implementing Online Lookup for Hybrid Apps

In this example, online lookup provides direct interaction between the data requester (client) and the enterprise information system (EIS), supplying real-time EIS data rather than cached data.

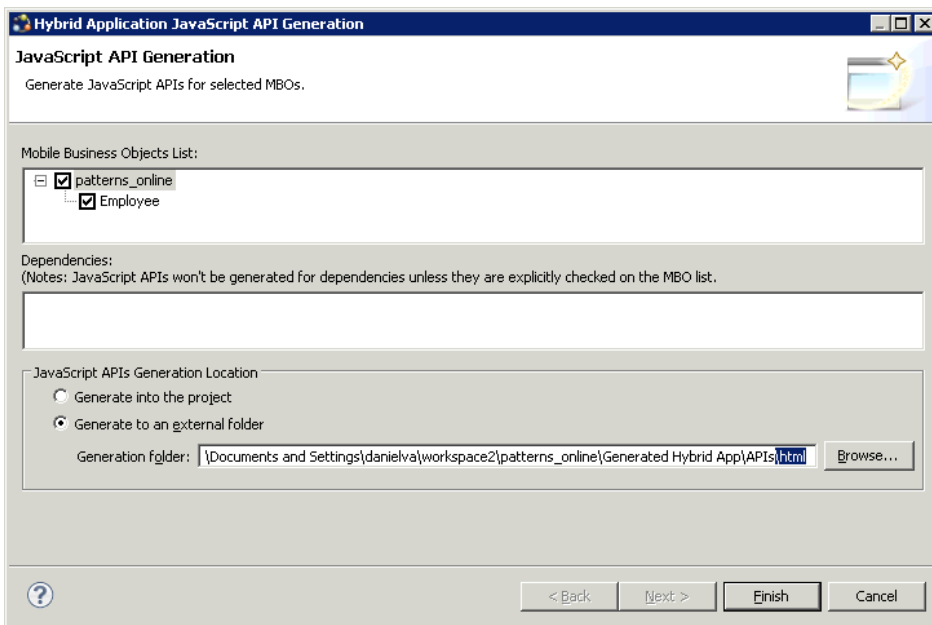
Prerequisites

Complete the procedure in *Defining Load Arguments from Mapped Propagate to Attributes* on page 635 so that you have an MBO with the required attributes.

Task

This section describes how to invoke the Employee's `findByParameter` method.

1. Right-click on the mobile application project and choose **Generate Hybrid App API**.
2. Select the Employee MBO, choose **Generate to an external folder**, and add `\html` to end of the folder name.



3. Right-click on the generated **html** folder and select **New > Other > General > File**.
4. Enter `online.html` for the file name.
5. Open the `online.html` file and add this code:

```

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
    <meta name="HandheldFriendly" content="True" />
    <meta http-equiv="PRAGMA" content="NO-CACHE" />
    <meta name="viewport" content="initial-scale =
1.0 ,maximum-scale = 1.0" />
    <script src="js/PlatformIdentification.js"></script>
    <script src="js/hwc-api.js"></script>
    <script src="js/hwc-comms.js"></script>
    <script src="js/hwc-utils.js"></script>
    <script src="js/WorkflowMessage.js"></script>
    <script src="js/HybridApp.js"></script>

    <script>
      function findEmp() {
        var deptID = document.getElementById("deptID").value;
        emp = new Employee();
        emp.deptIdLP = deptID;
        employee_findByParameter(emp,
"supusername=supAdmin&suppassword=s3pAdmin", "onError");
      }

      function onError(e) {
        alert("An error occurred");
      }
    </script>
  </head>
</html>

```


12. Click **Generate** to generate a deployable Hybrid App package.
13. Deploy and assign the Hybrid App package using SAP Control Center.

Implementing Server Notification for Hybrid App Clients

Configure matching rules for MBO-related data on SAP Mobile Server.

Prerequisites

Complete the procedure in *Defining the Mobile Business Object* on page 639 so that you have an MBO with the required attributes.

Task

Any data changes matching these rules trigger a notification from SAP Mobile Server to the Hybrid App client. This section describes how to write HTML, JavaScript, and modify the `WorkflowClient.xml` to display the results of a server notification.

1. Right-click on the mobile application project and choose **Generate Hybrid App API**.
2. Select the Sales MBO, choose **Generate to an external folder**, and add `\html` to end of the folder name.
3. Right-click on the generated **html** folder and select **New > Other > General > File**.
4. Enter `notification.html` for the file name.
5. Open the `notification.html` file and add this code:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
    <meta name="HandheldFriendly" content="True" />
    <meta http-equiv="PRAGMA" content="NO-CACHE" />
    <meta name="viewport" content="initial-scale =
1.0 ,maximum-scale = 1.0" />
    <script src="js/PlatformIdentification.js"></script>
    <script src="js/hwc-api.js"></script>
    <script src="js/hwc-comms.js"></script>
    <script src="js/hwc-utils.js"></script>
    <script src="js/WorkflowMessage.js"></script>
    <script src="js/HybridApp.js"></script>
    <script>
      hwc.processDataMessage = function
(incomingDataMessageValue) {
        if (incomingDataMessageValue.indexOf("<M>") != 0) {
          alert("An error occurred! " +
incomingDataMessageValue);
        }
        var workflowMessage = new
WorkflowMessage(incomingDataMessageValue);
        var values = workflowMessage.getValues();
        var salesOrderList = values.getData("Sales_order");
        var salesOrder = salesOrderList.value[0];
        var salesOrderId =
```

```

salesOrder.getData("Sales_order_id_attribKey").value;
    var custId =
salesOrder.getData("Sales_order_cust_id_attribKey").value;
    alert("The customer id for sales order " + salesOrderId
+ " is " + custId);
    }
</script>
</head>
<body onload="hwc.onHybridAppLoad_CONT()">
    <h3>Server Notification Sample</h3>
    <button id="closeHybridApp" onclick="hwc.close()">Close</
button>
</body>
</html>

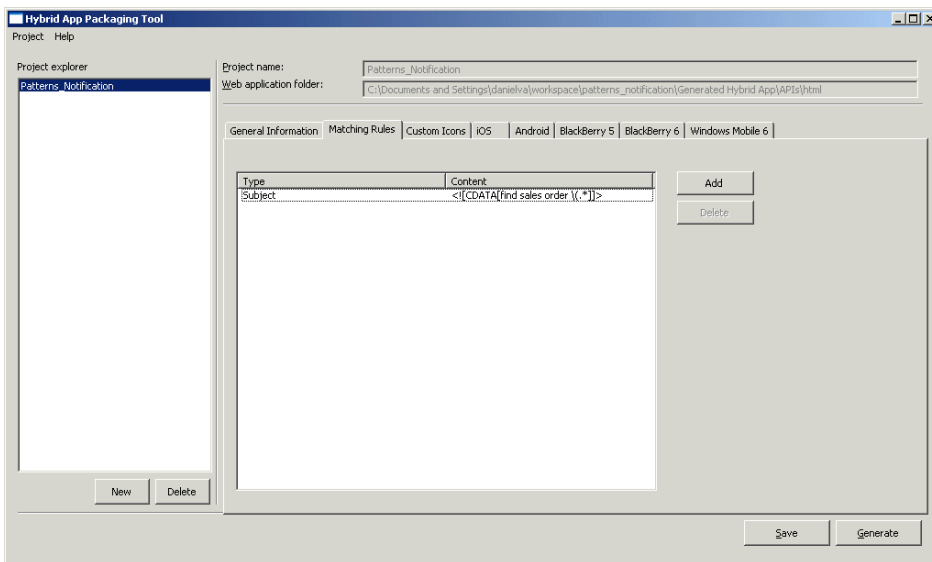
```

Five of the included files are from the <SMP_HOME>\MobileSDK23\HybridApp\API folder. The file named HybridApp.js is generated based on the operations and object queries of the MBOs selected in the Generate Hybrid App API wizard. In the onload event, the method hwc.onHybridAppLoad_CONT() is called. For server-initiated applications this returns the data message associated with this instance of the server-initiated application as a parameter to hwc.processDataMessage(). In processDataMessage, some of the data is extracted from the application message and displayed.

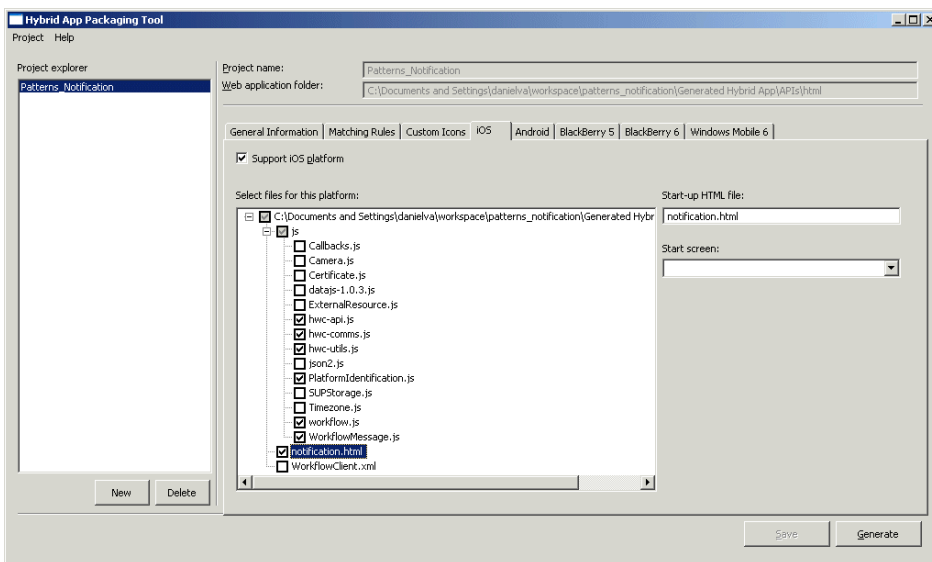
6. Navigate to SMP_HOME\MobileSDK23\HybridApp\PackagingTool and double-click the packagingtool.bat file.
7. Click **Browse** to enter the file path for your project and click **OK**.
8. Select **Project > New**.
9. Fill in Patterns_Notification and the location of where the generated files currently exist (the same location specified as the Generation folder above) for the Project name and Web root directory.
10. Fill in the MBO package name and version to match the deployed package.
11. Specify a matching rule for the subject:

```
<![CDATA[find sales order \(.*)]]>
```

Develop Hybrid Apps Using Third-party Web Frameworks



12. Specify the files to include in the Hybrid App for each supported platform. Only the selected files will appear in the `manifest.xml` file.



13. Open the generated `WorkflowClient.xml` file and update the Notifications section:

```
<Notifications>
  <Notification type="onEmailTriggered" targetscreen="Salesorder"
  asyncRequestErrorScreen="" errorNotificationSubjectLine=""
  errorNotificationFromLine="" asyncRequestErrorlogs=""
  asyncRequestErrorLogMessage=""
```

```

asyncRequestErrorLogMessageAsList="">
  <Transformation>
    <Rule type="regex-extract" source="subject"
workflowKey="order_id" workflowType="number" beforeMatch="find
sales order \( " afterMatch="\)" format=""/>
  </Transformation>
  <Methods>
    <Method name="findByParameter" type="search"
mbo="Sales_order" package="patterns_notification:1.0">
      <InputBinding opname="findByParameter" optype="none">
        <Value sourceType="Key" workflowKey="order_id"
contextVariable="" paramName="order_id" attribName="id"
mboType="int" convertToLocalTime="false"/>
      </InputBinding>
      <OutputBinding generateOld="true">
        <Mapping workflowKey="Sales_order" workflowType="list"
mboType="list">
          <Mapping workflowKey="Sales_order_id_attribKey"
workflowType="number" attribName="id" mboType="int"/>
          <Mapping workflowKey="Sales_order_cust_id_attribKey"
workflowType="number" attribName="cust_id" mboType="int"/>
          <Mapping workflowKey="Sales_order_order_date_attribKey"
workflowType="date" attribName="order_date" mboType="date"/>
          <Mapping
workflowKey="Sales_order_fin_code_id_attribKey"
workflowType="text" attribName="fin_code_id" mboType="string"/>
          <Mapping workflowKey="Sales_order_region_attribKey"
workflowType="text" attribName="region" mboType="string"/>
        </Mapping>
      </OutputBinding>
    </Method>
  </Methods>
</Notification>
</Notifications>

```

14. Save and close the file.

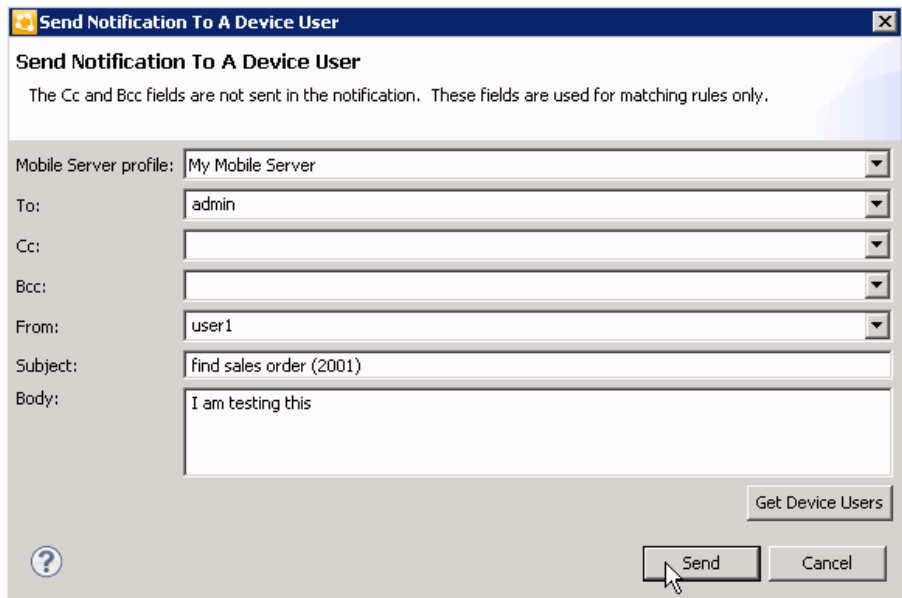
15. In the Hybrid App Packaging Tool, click **Generate** to create a deployable package.

16. Login in to SAP Control Center to deploy and assign the Hybrid App package.

17. Send a notification to the device.

Typically this is triggered by a database trigger or by the EIS sending a DCN. You can also use the Send a Notification wizard in the Hybrid App designer.

- a) In the Hybrid App designer, click **Flow Design**.
- b) Right-click in the Flow Design page and select **Send a notification**.



Implementing the Cached Data Pattern for MBO-based Hybrid Apps

For access to cached data, define a menu action and bind it to the `findByDeptId` object query.

Prerequisites

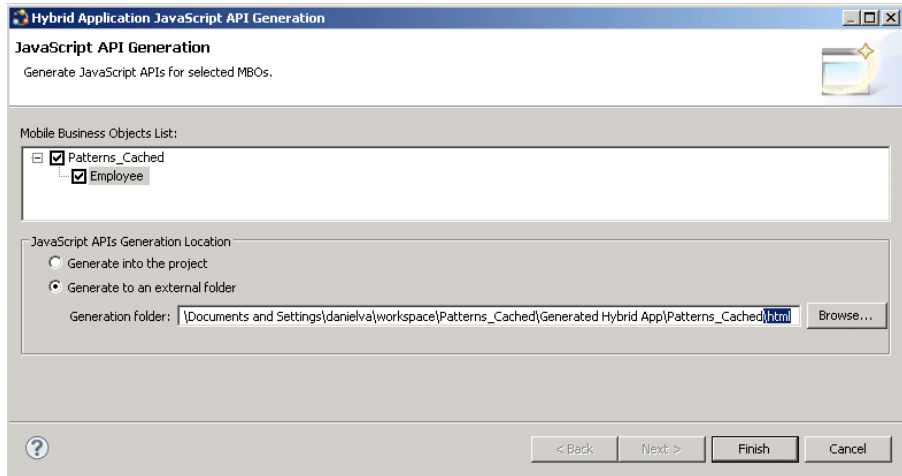
Complete the procedure in *Defining the Mobile Business Object* on page 645 so that you have an MBO with the required attributes.

Task

Using cached data is efficient when access to cached data is sufficient to meet business needs. For example, it may be sufficient to refresh the cache once a day for noncritical MBO data that changes infrequently.

1. Generate the Hybrid App API:

- a) Right-click the mobile application project and choose **Generate Hybrid App API**.
- b) In the tree view, select the **Employee** MBO, which contains the `findByDeptId` object query.
- c) Choose **Generate to an external folder** and add `"\html"` to end of the folder name.



By default, the wizard creates a Generated Hybrid App folder under the project and a sub folder named APIs.

d) Click **Finish**.

2. Right-click the `html` folder and choose **New > Other > General > File**, and enter `cached.html` for the file name.
3. Copy and paste the following contents into the `cached.html` file:

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
    <meta name="HandheldFriendly" content="True" />
    <meta http-equiv="PRAGMA" content="NO-CACHE" />
    <meta name="viewport" content="initial-scale =
1.0 , maximum-scale = 1.0" />
    <script src="js/PlatformIdentification.js"></script>
    <script src="js/hwc-comms.js"></script>
    <script src="js/hwc-utils.js"></script>
    <script src="js/WorkflowMessage.js"></script>
    <script src="js/HybridApp.js"></script>

    <script>
function findByDept() {
    var deptID = document.getElementById("deptID").value;
    emp = new Employee();
    emp.deptIDLp = deptID;
    employee_findByDeptId(emp,
"supusername=supAdmin&suppassword=s3pAdmin", "onError");
}

function onError(e) {
    alert("An error occurred");
}

hwc.processDataMessage = function
```


Generating JavaScript MBO Access API

Generate JavaScript API for MBOs to use in your custom code.

The generated API automatically includes, and utilizes, the Container APIs, along with the message manipulation APIs from the AppFramework portion of the Mobile SDK. The wizard also generates the `WorkflowClient.xml` file, which is required to support those functions.

Note: The generated `WorkflowClient.xml` file does not include a completed Notification, so if you want a server-initiated Hybrid App, you must do this by hand.

1. In WorkSpace Navigator, right-click the Mobile Application project for which to generate the JavaScript, and choose **Generate Hybrid App API**.
2. In the tree view, select the MBOs for which to generate the JavaScript.
3. Accept the default location for the files, or specify the location for the generated files and click **Finish**.

By default, the wizard creates a `Generated Hybrid App` folder under the project, and a subfolder named `APIs`.

Generated Hybrid App Files

Examine the generated files.

- `WorkflowClient.xml` – this file establishes the mapping between the XML messages and JSON calls to and from the SAP Mobile Server server.

Note: The generated `WorkflowClient.xml` does not include a completed notification, so if you want the Hybrid App to be server-initiated, you must write the Notification section. See *Creating Notifications to Make the Hybrid App Server-Initiated*.

- `WorkflowMessage.js` – defines some convenient functions for accessing incoming application messages.
- `Workflow.js` – contains functions map to the MBO's operations and object queries. The contents depend on the MBOs you select when you run the wizard, since the wizard generates only the JavaScript API functions for the selected MBOs.
- These files are static, container related, commonly used JavaScript libraries and are copied from the `<SMP_HOME>\UnwiredPlatform\Mobile SDK HybridApp\API\Container` folder.
 - `Callbacks.js` – Hybrid Web Container callback functions.
 - `SUPStorage.js` – client side key/value storage.
 - `hwc-comms.js` – native container functions that are invoked from the custom code.
 - `Camera.js` – functions for accessing the device's native camera functionality.
 - `Timezone.js` – utility functions for getting the local time.

Develop Hybrid Apps Using Third-party Web Frameworks

- `hwc-utils.js` – native Hybrid Web Container utility functions.
- `Certificate.js` – functions for processing certificates.
- `json2.js` – functions for processing JSON data.
- `ExternalResource.js` – functions for accessing external resources.
- `datajs-1.0.2.js` – functions for communicating through an OData protocol.

- `PlatformIdentification.js` – utility functions for checking the platform.
- `hwc-api.js` – native Hybrid Web Container functions that allow users access to Hybrid App metadata and notifications from the custom code.

HybridApp.js

In the `HybridApp.js` file, helper JavaScript structures are generated for the selected MBOs, and for the MBOs that have one-to-one, or one-to-many relationships.

Functions against selected MBO operations and object queries are also generated.

This is an example of the generated JavaScript for the Department MBO and Employee MBO in which the Department MBO has a one-to-many relationship with the Employee MBO.

```
/**
 * Returns The constructor of an mbo structure. This is helper
 function for creating MBO's operations or namedQuery input
 structure
 * @param attributes The parameters of an mbo operation, separated by
 one space. If the parameters map to MBO's attributes, use attributes
 name instead.
 * @param children The relationship names of an mbo operation's
 parameters or the array type of parameters, separated by one
 space.
 */
function makeClass(attributes, children) {
    var attributeNames = attributes.split(' ');
    var attributeCount = attributeNames.length;
    var childrenNames = children.split(' ');
    var childrenCount = childrenNames.length;

    function constructor() {
        for (var i = 0; i < attributeCount; i++)
        {
            var name = attributeNames[i];
            var subAttr = null;

            //If the name contains . which should be structure,
            while( name.indexOf('.') >0 )
            {
                var part = name.substring( 0,
name.indexOf('.'));
                if ( subAttr )
                {
                    subAttr.part = new Object();
                    subAttr = subAttr.part;
                }
            }
        }
    }
}
```

```

        }else
        {
            this[part]= new Object();
            subAttr = this[part];
        }
        name = name.substring( name.indexOf('.')+1,
name.length);
    }

    if ( subAttr )
    {
        subAttr[name]= new Object();
    }else {
        this[name] = new Object();
    }
}

for (var i = 0; i < childrenCount; i++) {
    this[childrenNames[i]] =[];
    this['OldValue_' + childrenNames[i]] =[];
}

this['__state'] ="";
this['pks'] = {};

var self = this;

this['pks'].put = function(pkName, pkValue ) {
    self['pks'][pkName] = pkValue ;
}
}
return constructor;
}

```

Set the "__state" field to "add," "delete," or "update" to add or delete an MBO, or to update a child MBO to a parent MBO, respectively.

Use the "pks" field to set values for operation parameters that have personalization keys.

This example shows the JavaScript structures generated for a Department MBO and Employee JavaScript:

```

/**
 * Returns Department MBO structure.
 * Used by JavaScript functions of
department_create_submit,department_create_onlineRequest,department
_update_submit,department_update_onlineRequest,department_delete_su
bmit,department_delete_onlineRequest,department_findAll,department_
findByPrimaryKey
 * @param dept_id The "dept_id" is attribute field of MBO
Department
 * @param dept_name The "dept_name" is attribute field of MBO
Department
 * @param dept_head_id The "dept_head_id" is attribute field of MBO
Department
 * @param Employee is MBO Employee javaScript structure array which

```

```

representing the MBO Department to MBO Employee one to many
relationship
*/
Department = makeClass( "dept_id dept_name dept_head_id",
"Employee" );
/**
 * Returns Employee MBO structure.
 * Used by JavaScript functions of
employee_create_submit,employee_create_onlineRequest,employee_updat
e_submit,employee_update_onlineRequest,employee_delete_submit,emplo
yee_delete_onlineRequest,employee_findAll,employee_findByPrimaryKey

 * @param emp_id The "emp_id" is attribute field of MBO Employee
 * @param manager_id The "manager_id" is attribute field of MBO
Employee
 * @param emp_fname The "emp_fname" is attribute field of MBO
Employee
 * @param emp_lname The "emp_lname" is attribute field of MBO
Employee
 * @param dept_id The "dept_id" is attribute field of MBO Employee
 * @param street The "street" is attribute field of MBO Employee
 * @param city The "city" is attribute field of MBO Employee
 * @param state The "state" is attribute field of MBO Employee
 * @param zip_code The "zip_code" is attribute field of MBO
Employee
 * @param phone The "phone" is attribute field of MBO Employee
 * @param status The "status" is attribute field of MBO Employee
 * @param ss_number The "ss_number" is attribute field of MBO
Employee
 * @param salary The "salary" is attribute field of MBO Employee
 * @param start_date The "start_date" is attribute field of MBO
Employee
 * @param termination_date The "termination_date" is attribute field
of MBO Employee
 * @param birth_date The "birth_date" is attribute field of MBO
Employee
 * @param bene_health_ins The "bene_health_ins" is attribute field of
MBO Employee
 * @param bene_life_ins The "bene_life_ins" is attribute field of MBO
Employee
 * @param bene_day_care The "bene_day_care" is attribute field of MBO
Employee
 * @param sex The "sex" is attribute field of MBO Employee
 */
Employee = makeClass( "emp_id manager_id emp_fname emp_lname dept_id
street city state zip_code phone status ss_number salary start_date
termination_date birth_date bene_health_ins
bene_life_insbene_day_care sex" , "" );

```

If there is one parameter that does not map to the MBO's attribute, the JavaScript structure for the MBO's function input parameters is generated. This example shows an MBO called Banks where the dataSource is an SAP® object. In addition to the Banks JavaScript structure, the BANK_LIST and Banks_getList JavaScript structures are also generated.

```

/***** DEFINITION OF MBO JAVASCRIPT STRUCTURE
*****/
/**
 * Returns BANK_LIST structure
 * @param BANK_CTRY The "BANK_CTRY" is the parameter field of
BANK_LIST.
 * @param BANK_NAME The "BANK_NAME" is the parameter field of
BANK_LIST.
 */
BANK_LIST = makeClass("BANK_CTRY BANK_NAME" , "" )
/**
 * Returns Banks_getList structure. Used by functions of
banks_getList_submit and banks_getList_onlineRequest
 * @param BANK_CTRY The "BANK_CTRY " is the parameter field of the
operation of getList.
 * @param BANK_LIST The "BANK_LIST " is the array parameter field of
the operation of getList.
 */
Banks_getList = makeClass( "BANK_CTRY", "BANK_LIST" );

/**
 * Returns Banks MBO structure.
 * Used by JavaScript functions of banks_findAll,banks_getByName
 * @param BANK_CTRY The "BANK_CTRY" is attribute field of MBO Banks
 * @param BANK_KEY The "BANK_KEY" is attribute field of MBO Banks
 * @param BANK_NAME The "BANK_NAME" is attribute field of MBO Banks
 * @param CITY The "CITY" is attribute field of MBO Banks
 * @param BAPI_BANK_GETLIST_BANK_LIST1 is MBO
BAPI_BANK_GETLIST_BANK_LIST1 javaScript structure array which
representing the MBO Banks to MBO BAPI_BANK_GETLIST_BANK_LIST1 one to
many relationship
 */
Banks = makeClass( "BANK_CTRY BANK_KEY BANK_NAME CITY" ,
"BAPI_BANK_GETLIST_BANK_LIST1" );

```

Global variables are generated for each MBO operation. You can reference these global variables in your code when you process incoming data to check which action was performed for the incoming message.

```

/*
 * Global variables for Customer actions
 */
Customer.createAction = "Customer_create";
Customer.updateAction = "Customer_update";
Customer.deleteAction = "Customer_delete";
Customer.findAllAction = "Customer_findAll";
Customer.findByPrimaryKeyAction = "Customer_findByPrimaryKey" ;

```

Two versions of JavaScript functions are generated for the MBO's create, read, update, delete operations. For example, for a **create** operation there is a **create_submit** function and **create_onlinerequest** function generated. This example shows the generated JavaScript function for the Department **create** operation:

Develop Hybrid Apps Using Third-party Web Frameworks

```
/**
 * Returns void. This is submit operation, therefore no message will
 be sent back to user by the hybrid web container
 * @param departmentObj which is the instance of Department
 JavaScript structure. Values should be set for this instance.
 * @param credInfo, It is string value , should be something look
 like "supusername=username&suppassword=password".
 * @param keepOpen, If this set to true, the Hybrid App will be kept
 open, otherwise, the hybrid web container will close the Hybrid
 App.
 */

function department_create_submit(departmentObj, credInfo,
keepOpen)
{
    //Collect values from departmentObj customerObj and fill the
action parameters
    var keys = ["Department_create_dept_id_paramKey",
"Department_create_dept_name_paramKey",
"Department_create_dept_head_id_paramKey"];
    var types = ["int", "string", "int"];
    var objValues = [departmentObj.dept_id, departmentObj.dept_name,
departmentObj.dept_head_id];

    var workflowMessageToSend = new WorkflowMessage("");
    workflowMessageToSend.setHeader("");

workflowMessageToSend.setRequestAction( "Department_create");

    createMessageValues( workflowMessageToSend.getValues(), keys,
types, objValues );

    if ( departmentObj.Employee && departmentObj.Employee.length >
0 )
// we have list object array
    {
        var department_employees = new MessageValue();
        department_employees.key = "Department_employees";
        department_employees.isNull = false;
        department_employees.type = "LIST";

        var employeekeys = ["Employee_emp_id_attribKey",
"Employee_manager_id_attribKey", "Employee_emp_fname_attribKey",
"Employee_emp_lname_attribKey", "Employee_dept_id_attribKey",
"Employee_street_attribKey", "Employee_city_attribKey",
"Employee_state_attribKey", "Employee_zip_code_attribKey",
"Employee_phone_attribKey", "Employee_status_attribKey",
"Employee_ss_number_attribKey", "Employee_salary_attribKey",
"Employee_start_date_attribKey",
"Employee_termination_date_attribKey",
"Employee_birth_date_attribKey",
"Employee_bene_health_ins_attribKey",
"Employee_bene_life_ins_attribKey",
"Employee_bene_day_care_attribKey", "Employee_sex_attribKey"];
    }
```

```

        var employeetypes = ["int", "int", "string", "string", "int",
"string", "string", "string", "string", "string", "string",
"string", "decimal", "DateTime", "DateTime", "DateTime", "string",
"string", "string", "string"];

        var employeeValues = [];

        for( var employeei = 0 ; employeei <
departmentObj.Employee.length ; employeei ++ )
        {
            var employeelc = new MessageValueCollection();
            employeelc.key = guid();
            employeelc.parent = "Department_employees";
            employeelc.parentValue = department_employees
            employeelc.state =
departmentObj.Employee[employeei].__state;

            var employeeObjValues = [];

            employeeObjValues.push( departmentObj.Employee[employeei].emp_id);

            employeeObjValues.push( departmentObj.Employee[employeei].manager_id);

            employeeObjValues.push( departmentObj.Employee[employeei].emp_fname);

            employeeObjValues.push( departmentObj.Employee[employeei].emp_lname);

                employeeObjValues.push( departmentObj.dept_id);

            employeeObjValues.push( departmentObj.Employee[employeei].street);

            employeeObjValues.push( departmentObj.Employee[employeei].city);

            employeeObjValues.push( departmentObj.Employee[employeei].state);

            employeeObjValues.push( departmentObj.Employee[employeei].zip_code);

            employeeObjValues.push( departmentObj.Employee[employeei].phone);

            employeeObjValues.push( departmentObj.Employee[employeei].status);

            employeeObjValues.push( departmentObj.Employee[employeei].ss_number

```

```

);

employeeObjValues.push( departmentObj.Employee[employeei].salary);

    employeeObjValues.push( departmentObj.Employee[employeei].start_da
te);

employeeObjValues.push( departmentObj.Employee[employeei].terminati
on_date);

employeeObjValues.push( departmentObj.Employee[employeei].birth_dat
e);

employeeObjValues.push( departmentObj.Employee[employeei].bene_heal
th_ins);

employeeObjValues.push( departmentObj.Employee[employeei].bene_life
_ins);

employeeObjValues.push( departmentObj.Employee[employeei].bene_day_c
are);

    employeeObjValues.push( departmentObj.Employee[employeei].sex);

        createMessageValues( employeelc ,employeekeys ,
employeetypes, employeeObjValues );

        //Find this Employee old values if it has.
        for( var oldValueemployeei = 0 ; oldValueemployeei <
departmentObj.OldValue_Employee.length ;
            oldValueemployeei ++ )
        {
            if
( departmentObj.OldValue_Employee[ oldValueemployeei ].emp_id ===
            departmentObj.Employee[ employeei].emp_id
            )
            {
                var oldValue_employeekeys =
["_old.Department.emp_id", "_old.Department.manager_id",
"_old.Department.emp_fname", "_old.Department.emp_lname",
"_old.Department.dept_id", "_old.Department.street",
"_old.Department.city", "_old.Department.state",
"_old.Department.zip_code", "_old.Department.phone",
"_old.Department.status", "_old.Department.ss_number",
"_old.Department.salary", "_old.Department.start_date",
"_old.Department.termination_date", "_old.Department.birth_date",
"_old.Department.bene_health_ins", "_old.Department.bene_life_ins",
"_old.Department.bene_day_care", "_old.Department.sex"];
                var oldValue_employeetypes = ["INT", "INT",
"STRING", "STRING", "INT", "STRING", "STRING", "STRING",
"STRING", "STRING", "STRING", "DECIMAL", "DATE", "DATE", "DATE",
"STRING", "STRING", "STRING", "STRING"];
                var oldValue_employeeValues = [];

```



```
oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].emp_id);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].manager_id);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].emp_fname);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].emp_lname);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].dept_id);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].street);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].city);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].state);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].zip_code);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].phone);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].status);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].ss_number);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].salary);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].start_date);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].termination_date);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].birth_date);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].bene_health_ins);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].bene_life_ins);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
lueemployeei].bene_day_care);

oldValue_employeeValues.push( departmentObj.OldValue_Employee[oldVa
lueemployeei].sex);

createMessageValues( employeeelc,oldValue_employeekeys ,
oldValue_employeetypes, oldValue_employeeValues );

                break;
            }
        }
// end of old values --->

        employeeValues.push( employeeelc);
    }

    department_employees.setValue( employeeValues);

    workflowValues.add( department_employees.getKey(),
department_employees);
}
hwc.doSubmitWorkflow_CONT( credInfo,
workflowMessageToSend.serializeToString(),workflowMessageToSend.get
HasFileMessageValue());
}
/**
 * Returns void. This is an onlineRequest operation, therefore the
message will be sent back to the user by the Hybrid Web Container.
Handle the result in the function
customAfterDataReceived(incomingWorkflowMessage)defined in
Custom.js.
 * @param departmentObj, which is the instance of Department
JavaScript structure. Values should be set for this instance.
 * @param credInfo, which is a string value, and should look like
"supusername=username&suppassword=password".
 * @param errorCallback, name of the function to be called if an
online request fails.
 */

function department_create_onlineRequest(departmentObj, credInfo ,
errorCallback)
{
    var keys = ["Department_create_dept_id_paramKey",
"Department_create_dept_name_paramKey",
"Department_create_dept_head_id_paramKey"];
    .....
    ....
    ..

```

WorkflowClient.xml

The `WorkflowClient.xml` file defines all of an application's action mappings that correspond to selected MBO operations and named queries.

Below is part of an example of the generated `WorkflowClient.xml` for the create operation on the Department MBO. Since the department has a one-to-many relationship to the Employee MBO, all input mappings for Department MBO and Employee MBO are also defined.

```
<?xml version="1.0" encoding="utf-8"?>
<Workflow xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="WorkflowClient.xsd" >
  <Globals>
    <DefaultScreens activation="" credentials=""/>
  </Globals>
  <Triggers>
    <Actions>
      <Action name="Department_create" sourcescreen=""
targetscreens="" errorscreens="">
        <Methods>
          <Method type="replay" mbo="Department"
package="apiTestttDepartmentOneToMany:1.0"
showCredScreenOnAuthFailure="true" >
            <InputBinding optype="create" opname="create"
generateOld="false">
              <Value sourceType="Key"
workflowKey="Department_create_dept_id_paramKey"
paramName="dept_id" attribName="dept_id" mboType="int"/>
              <Value sourceType="Key"
workflowKey="Department_create_dept_name_paramKey"
paramName="dept_name" attribName="dept_name" mboType="string"/>
              <Value sourceType="Key"
workflowKey="Department_create_dept_head_id_paramKey"
paramName="dept_head_id" attribName="dept_head_id" mboType="int"/>
            >
              <Value sourceType="Key"
workflowKey="Department_employees" relationShipName="employees"
mboType="list">
                <InputBinding actiontype="create" optype="create"
opname="create" generateOld="true">
                  <Value sourceType="Key"
workflowKey="Employee_emp_id_attribKey" contextVariable=""
paramName="emp_id" attribName="emp_id" mboType="int"/>
                  <Value sourceType="Key"
workflowKey="Employee_manager_id_attribKey" contextVariable=""
paramName="manager_id" attribName="manager_id" mboType="int"/>
                  <Value sourceType="Key"
workflowKey="Employee_emp_fname_attribKey" contextVariable=""
paramName="emp_fname" attribName="emp_fname" mboType="string"/>
                  <Value sourceType="Key"
workflowKey="Employee_emp_lname_attribKey" contextVariable=""
paramName="emp_lname" attribName="emp_lname" mboType="string"/>
                  <Value sourceType="Key"
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
workflowKey="Employee_dept_id_attribKey" contextVariable=""
paramName="dept_id" attribName="dept_id" mboType="int"/>
    <Value sourceType="Key"
workflowKey="Employee_street_attribKey" contextVariable=""
paramName="street" attribName="street" mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_city_attribKey" contextVariable=""
paramName="city" attribName="city" mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_state_attribKey" contextVariable=""
paramName="state" attribName="state" mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_zip_code_attribKey" contextVariable=""
paramName="zip_code" attribName="zip_code" mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_phone_attribKey" contextVariable=""
paramName="phone" attribName="phone" mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_status_attribKey" contextVariable=""
paramName="status" attribName="status" mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_ss_number_attribKey" contextVariable=""
paramName="ss_number" attribName="ss_number" mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_salary_attribKey" contextVariable=""
paramName="salary" attribName="salary" mboType="decimal"/>
    <Value sourceType="Key"
workflowKey="Employee_start_date_attribKey" contextVariable=""
paramName="start_date" attribName="start_date" mboType="date"/>
    <Value sourceType="Key"
workflowKey="Employee_birth_date_attribKey" contextVariable=""
paramName="birth_date" attribName="birth_date" mboType="date"/>
    <Value sourceType="Key"
workflowKey="Employee_bene_health_ins_attribKey" contextVariable=""
paramName="bene_health_ins" attribName="bene_health_ins"
mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_bene_life_ins_attribKey" contextVariable=""
paramName="bene_life_ins" attribName="bene_life_ins"
mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_bene_day_care_attribKey" contextVariable=""
paramName="bene_day_care" attribName="bene_day_care"
mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_sex_attribKey" contextVariable=""
paramName="sex" attribName="sex" mboType="string"/>
    </InputBinding>
    <InputBinding optype="none">
    <Value sourceType="Key"
workflowKey="Employee_emp_id_attribKey" attribName="emp_id"
mboType="int"/>
    <Value sourceType="Key"
workflowKey="Employee_manager_id_attribKey" attribName="manager_id"
mboType="int"/>
    <Value sourceType="Key"
```

```

workflowKey="Employee_emp_fname_attribKey" attribName="emp_fname"
mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_emp_lname_attribKey" attribName="emp_lname"
mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_dept_id_attribKey" attribName="dept_id"
mboType="int"/>
    <Value sourceType="Key"
workflowKey="Employee_street_attribKey" attribName="street"
mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_city_attribKey" attribName="city"
mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_state_attribKey" attribName="state"
mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_zip_code_attribKey" attribName="zip_code"
mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_phone_attribKey" attribName="phone"
mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_status_attribKey" attribName="status"
mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_ss_number_attribKey" attribName="ss_number"
mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_salary_attribKey" attribName="salary"
mboType="decimal"/>
    <Value sourceType="Key"
workflowKey="Employee_start_date_attribKey" attribName="start_date"
mboType="date"/>
    <Value sourceType="Key"
workflowKey="Employee_birth_date_attribKey" attribName="birth_date"
mboType="date"/>
    <Value sourceType="Key"
workflowKey="Employee_bene_health_ins_attribKey"
attribName="bene_health_ins" mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_bene_life_ins_attribKey"
attribName="bene_life_ins" mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_bene_day_care_attribKey"
attribName="bene_day_care" mboType="string"/>
    <Value sourceType="Key"
workflowKey="Employee_sex_attribKey" attribName="sex"
mboType="string"/>
</InputBinding>
<InputBinding actiontype="update" optype="update"
opname="update" generateOld="true">
    <Value sourceType="Key"
workflowKey="Employee_manager_id_attribKey" contextVariable=""
paramName="manager_id" attribName="manager_id" mboType="int"/>

```

```

        <Value sourceType="Key"
workflowKey="Employee_emp_fname_attribKey" contextVariable=""
paramName="emp_fname" attribName="emp_fname" mboType="string"/>
        <Value sourceType="Key"
workflowKey="Employee_emp_lname_attribKey" contextVariable=""
paramName="emp_lname" attribName="emp_lname" mboType="string"/>
        <Value sourceType="Key"
workflowKey="Employee_dept_id_attribKey" contextVariable=""
paramName="dept_id" attribName="dept_id" mboType="int"/>
        <Value sourceType="Key"
workflowKey="Employee_street_attribKey" contextVariable=""
paramName="street" attribName="street" mboType="string"/>
        <Value sourceType="Key"
workflowKey="Employee_city_attribKey" contextVariable=""
paramName="city" attribName="city" mboType="string"/>
        <Value sourceType="Key"
workflowKey="Employee_state_attribKey" contextVariable=""
paramName="state" attribName="state" mboType="string"/>
        <Value sourceType="Key"
workflowKey="Employee_zip_code_attribKey" contextVariable=""
paramName="zip_code" attribName="zip_code" mboType="string"/>
        <Value sourceType="Key"
workflowKey="Employee_phone_attribKey" contextVariable=""
paramName="phone" attribName="phone" mboType="string"/>
        <Value sourceType="Key"
workflowKey="Employee_status_attribKey" contextVariable=""
paramName="status" attribName="status" mboType="string"/>
        <Value sourceType="Key"
workflowKey="Employee_ss_number_attribKey" contextVariable=""
paramName="ss_number" attribName="ss_number" mboType="string"/>
        <Value sourceType="Key"
workflowKey="Employee_salary_attribKey" contextVariable=""
paramName="salary" attribName="salary" mboType="decimal"/>
        <Value sourceType="Key"
workflowKey="Employee_start_date_attribKey" contextVariable=""
paramName="start_date" attribName="start_date" mboType="date"/>
        <Value sourceType="Key"
workflowKey="Employee_birth_date_attribKey" contextVariable=""
paramName="birth_date" attribName="birth_date" mboType="date"/>
        <Value sourceType="Key"
workflowKey="Employee_bene_health_ins_attribKey" contextVariable=""
paramName="bene_health_ins" attribName="bene_health_ins"
mboType="string"/>
        <Value sourceType="Key"
workflowKey="Employee_bene_life_ins_attribKey" contextVariable=""
paramName="bene_life_ins" attribName="bene_life_ins"
mboType="string"/>
        <Value sourceType="Key"
workflowKey="Employee_bene_day_care_attribKey" contextVariable=""
paramName="bene_day_care" attribName="bene_day_care"
mboType="string"/>
        <Value sourceType="Key"
workflowKey="Employee_sex_attribKey" contextVariable=""
paramName="sex" attribName="sex" mboType="string"/>
        <Value sourceType="Key"
workflowKey="Employee_emp_id_attribKey" contextVariable=""

```

```

paramName="emp_id" attribName="emp_id" mboType="int"/>
  </InputBinding>
  <InputBinding actiontype="delete" optype="delete"
opname="delete" generateOld="true">
  </InputBinding>
  </Value>

</InputBinding>
</OutputBinding/>
</Method>
</Methods>
</Action>

```

By default, the MBO has two named queries—`findById` and `findAll`. The method, input and output binding keys, and all of the dependency's key bindings are generated.

```

<Action name="Department_findByPrimaryKey" sourcescreen=""
targetscreen="" errorscreen="">
  <Methods>
    <Method name="findByIdPrimaryKey" type="search"
mbo="Department" package="apiTesttDepartmentOneToMany:1.0"
showCredScreenOnAuthFailure="true" >
      <InputBinding optype="none" opname="findByIdPrimaryKey"
generateOld="true">
        <Value sourceType="Key"
workflowKey="Department_dept_id_attribKey" paramName="dept_id"
attribName="dept_id" mboType="int"/>
        </InputBinding>
        <OutputBinding generateOld="true">
          <Mapping workflowKey="Department_dept_id_attribKey"
workflowType="number" attribName="dept_id" mboType="int"/>
          <Mapping workflowKey="Department_dept_name_attribKey"
workflowType="text" attribName="dept_name" mboType="string"/>
          <Mapping workflowKey="Department_dept_head_id_attribKey"
workflowType="number" attribName="dept_head_id" mboType="int"/>
          <Mapping workflowKey="Department_employees"
workflowType="list" relationshipName="employees" mboType="list">
            <Mapping workflowKey="Employee_emp_id_attribKey"
workflowType="number" relationshipName="employees"
attribName="emp_id" mboType="int"/>
            <Mapping workflowKey="Employee_manager_id_attribKey"
workflowType="number" relationshipName="employees"
attribName="manager_id" mboType="int"/>
            <Mapping workflowKey="Employee_emp_fname_attribKey"
workflowType="text" relationshipName="employees"
attribName="emp_fname" mboType="string"/>
            <Mapping workflowKey="Employee_emp_lname_attribKey"
workflowType="text" relationshipName="employees"
attribName="emp_lname" mboType="string"/>
            <Mapping workflowKey="Employee_dept_id_attribKey"
workflowType="number" relationshipName="employees"
attribName="dept_id" mboType="int"/>
            <Mapping workflowKey="Employee_street_attribKey"
workflowType="text" relationshipName="employees" attribName="street"
mboType="string"/>
            <Mapping workflowKey="Employee_city_attribKey"

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
workflowType="text" relationshipName="employees" attribName="city"
mboType="string"/>
    <Mapping workflowKey="Employee_state_attribKey"
workflowType="text" relationshipName="employees" attribName="state"
mboType="string"/>
    <Mapping workflowKey="Employee_zip_code_attribKey"
workflowType="text" relationshipName="employees"
attribName="zip_code" mboType="string"/>
    <Mapping workflowKey="Employee_phone_attribKey"
workflowType="text" relationshipName="employees" attribName="phone"
mboType="string"/>
    <Mapping workflowKey="Employee_status_attribKey"
workflowType="text" relationshipName="employees" attribName="status"
mboType="string"/>
    <Mapping workflowKey="Employee_ss_number_attribKey"
workflowType="text" relationshipName="employees"
attribName="ss_number" mboType="string"/>
    <Mapping workflowKey="Employee_salary_attribKey"
workflowType="number" relationshipName="employees"
attribName="salary" mboType="decimal"/>
    <Mapping workflowKey="Employee_start_date_attribKey"
workflowType="date" relationshipName="employees"
attribName="start_date" mboType="date"/>
    <Mapping workflowKey="Employee_birth_date_attribKey"
workflowType="date" relationshipName="employees"
attribName="birth_date" mboType="date"/>
    <Mapping
workflowKey="Employee_bene_health_ins_attribKey"
workflowType="text" relationshipName="employees"
attribName="bene_health_ins" mboType="string"/>
    <Mapping
workflowKey="Employee_bene_life_ins_attribKey" workflowType="text"
relationshipName="employees" attribName="bene_life_ins"
mboType="string"/>
    <Mapping
workflowKey="Employee_bene_day_care_attribKey" workflowType="text"
relationshipName="employees" attribName="bene_day_care"
mboType="string"/>
    <Mapping workflowKey="Employee_sex_attribKey"
workflowType="text" relationshipName="employees" attribName="sex"
mboType="string"/>
    </Mapping>
    </OutputBinding>
</Method>
</Methods>
</Action>
```

Note: By default, the <Notifications> section of the generated WorkflowClient.xml is empty, so you must write the <Notification> section for a server-initiated Hybrid App.

Creating Notifications to Make the Hybrid App Server-Initiated

To make the Hybrid App server-initiated, you must modify the `WorkflowClient.xml` file and create a notification.

By default, the `<Notifications>` section is empty.

1. Create a notification.

Each notification has two child nodes—`Transformation` and `Methods`.

2. Create a notification node, for example:

```
<Notifications>
  <Notification type="onEmailTriggered" targetscreens=" ">
    </Notification>
</Notifications>
```

You can simply copy the `Methods` from the appropriate object query (for example, `findByPrimaryKey`) that is generated automatically in the `WorkflowClient.xml` file, for example:

```
<Notifications>
  <Notification type="onEmailTriggered" targetscreens=" ">
    <Methods>
      <Method name="findByPrimaryKey" type="search"
mbo="Department" package="apiTesttDepartmentOneToMany:1.0"
showCredScreenOnAuthFailure="true" >
        <InputBinding optype="none" opname="findByPrimaryKey"
generateOld="true">
          <Value sourceType="Key"
workflowKey="Department_dept_id_attribKey" paramName="dept_id"
attribName="dept_id" mboType="int"/>
        </InputBinding>
        <OutputBinding generateOld="true">
          <Mapping workflowKey="Department_dept_id_attribKey"
workflowType="number" attribName="dept_id" mboType="int"/>
          [...]
        </OutputBinding>
      </Method>
    </Methods>
  </Notification>
</Notifications>
```

3. Create a Transformation node.

You must manually write the `Transformation` section. The contents depend on how many input parameters the object query has. For each input parameter, you need a corresponding `Rule` node as a child of the `Transformation` node. The `workflowKey` of the `Rule` node corresponds to the `InputBinding`'s `Value` for that input parameter. For example:

```
<Notifications>
  <Notification type="onEmailTriggered" targetscreens=" ">
    <Transformation>
  </Transformation>
  <Methods>
```

```

    <Method name="findByPrimaryKey"
      type="search" mbo="Department"
      package="apiTesttDepartmentOneToMany:1.0"
      showCredScreenOnAuthFailure="true" >
      <InputBinding optype="none" opname="findByPrimaryKey"
        generateOld="true">
        <Value sourceType="Key"
          workflowKey="Department_dept_id_attriKey"
          paramName="dept_id" attribName="dept_id"
          mboType="int"/>
        </InputBinding>
        <OutputBinding
          generateOld="true">
          <Mapping
            workflowKey="Department_dept_id_attriKey"
            workflowType="number" attribName="dept_id"
            mboType="int"/>
            [...]
          </OutputBinding>
        </Method>
      </Methods>
    </Notification>
  </Notifications>

```

4. For each input parameter in the object query, create a corresponding Rule and make sure the workflowKey of the Rule matches the Value of the InputBinding. For example:

```

<Notifications>
  <Notification type="onEmailTriggered" targetscreen=" " >
    <Transformation>
      <Rule type="regex-extract"
        source="subject" workflowKey="dept_id"
        workflowType="number" beforeMatch="dept_id =
          " afterMatch="" format=""/>
      </Transformation>
    <Methods>
      <Method name="findByPrimaryKey"
        type="search" mbo="Department"
        package="apiTesttDepartmentOneToMany:1.0"
        showCredScreenOnAuthFailure="true" >
        <InputBinding optype="none" opname="findByPrimaryKey"
          generateOld="true">
          <Value sourceType="Key"
            workflowKey="dept_id"
            paramName="dept_id" attribName="dept_id" mboType="int"/>
          </InputBinding>
          <OutputBinding
            generateOld="true">
            <Mapping
              workflowKey="Department_dept_id_attriKey"
              workflowType="number" attribName="dept_id"
              mboType="int"/>
              [...]
            </OutputBinding>
          </Method>
        </Methods>

```

```
</Notification>
</Notifications>
```

5. Save the file.

Processing Responses From the Server

There are a couple of approaches for handling callback functions.

If you want to use the JavaScript APIs generated by the wizard, for online request functions, you must implement the function:

```
hwc.processDataMessage = function processDataMessage
(incomingDataMessageValue, noUI, loading, fromActivationFlow,
dataType) {

    // for example,
    // var workflowMessage = new
WorkflowMessage(incomingWorkflowMessage);

    //if ( workflowMessage.getRequestAction() ==
Customer.findByPrimaryKeyAction ){
    //so this workflow message is returned by calling
customer_findByPrimaryKey function

    //TODO; do whatever you want to do with the return data....

}
```

You can choose, instead, to take advantage of the other functions in the `SMP_HOME \UnwiredPlatform\MobileSDK<version>\` folder, specifically the files under the `AppFramework` folder. In these, the incoming and outgoing messages, how they are bound to the UI, and how navigation works are handled by the functions defined in the `API.js` and `Utils.js` files. You can add your custom code into your own JavaScript file. You must still create the UI and do so in a way that is compatible with the `AppFramework`.

Error Handling

Usually, errors come from the Hybrid Web Container or from the back-end server side.

For online requests, when the error comes from the Hybrid Web Container, handle it in the `errorCallback` function, for example:

```
department_create_onlineRequest(dep1,
                                "",
                                function(errorMessage) {
                                    //TODO: error occurred...
                                }
);
```

An error message passed as an incoming Hybrid App message in the user-defined function of `processDataMessage` is another type of error that comes from the back-end server. The `incomingDataMessageValue` should be similar to this:

```

<M><H><S>...</S> ..<V k="ErrorLogMessage" t="T">ERROR:.....</V><V
k=ErrorLogMessageAsList> .....</V>..</M>

hwc.processDataMessage = function processDataMessage
(incomingDataMessageValue, noUI, loading, fromActivationFlow,
dataType) {

    //// var workflowMessage = new
WorkflowMessage(incomingWorkflowMessage);

    //if ( workflowMessage.getRequestAction() ==
Customer.findByPrimaryKeyAction ){
    //    var detailErrorMsg =
workflowMessage.getValues().getData("ErrorLogMessage").getValue();
    // }
}

```

URL Parameters

When writing your own HTML and JavaScript, when the document is loaded, these URL parameters are present.

You can find an example of how to use these URL parameters in the `onHybridAppLoad()` function in the `Utils.js` file.

URL parameter	Description
loglevel	Current device log level.
screenToShow	Name of the screen to show.
supusername	User name of the current Hybrid App (if available).
lang	Current language of the device.
isalreadyprocessed	Indicates whether or not the Hybrid App message has been processed. The JavaScript can, for example, choose to show all controls as read-only if it has already been processed but viewed again.
loadtransformdata	Indicates that the JavaScript should request the transform data (contents of the e-mail message) from the Hybrid Web Container using the <code>loadtransformdata</code> query type. For information about the query types, see the topic <i>Calling the Hybrid Web Container</i> .

URL parameter	Description
ignoretransformsreen	Indicates that the JavaScript should ignore the <code>RequestScreen</code> tag in the transform data (contents of the message). This is set to true when the screen to show is either the Activation or Credentials screen.

Develop OData-based Hybrid Apps

The Hybrid App SDK includes the open source Datajs JavaScript library, which you can include as part of your application to access OData stores.

This library, along with the rest of the Hybrid App JavaScript API, is in `<SMP_HOME>\MobileSDK<version>\HybridApp\API\Container\Datajs-1.x.x.js`. As of this writing, the latest version of Datajs is 1.0.3.

The Datajs library is used to fetch the data used in your Hybrid App. This data can be displayed in your Hybrid App using a variety of UI frameworks such as jQuery Mobile, Sencha, or your favorite Web-based UI framework. Using OData in Hybrid Apps is similar to using the Rest API, described in *Developer Guide: REST API Applications*.

If the back-end OData service has support, you can use the Datajs library to read, modify, and delete data using standard HTTP methods (POST, PUT, DELETE, and so on).

The basic steps for developing an OData-based Hybrid App are:

1. Add the `<SMP_HOME>\UnwiredPlatform\MobileSDK<version>\HybridApp\API\Containers\datajs-1.0.3.js` to your Hybrid App.
2. Create a Hybrid App user interface with your preferred UI framework.
3. Use the Datajs library for create, read, update, and delete operations to the OData or HTTP end point and bind it to the UI.
4. Use the packaging tool to generate the `manifest.xml` file and Hybrid App ZIP package.
5. Use the Deploy Wizard in SAP Control Center to deploy the Hybrid App ZIP file.

Connect to an OData Source

The Datajs JavaScript library supports reading and writing to an OData service using both the JSON and ATOM-based formats.

The endpoint is an HTTP based URI exposed by the server.

You can use the `OData.read` API with a URI to read data from a server. To add, update, or delete data, the `ODATA.request` API can be used along with a POST, PUT, or DELETE method.

You can see examples at <http://datajs.codeplex.com/wikipage?title=OData%20Code%20Snippets&referringTitle=Using%20OData>

In your Hybrid App, you can connect to the Proxy endpoint exposed by SAP Mobile Platform using the Datajs library. This gives administrators and developers control over the endpoint as only white listed endpoints are accessible from the Hybrid App and also restricts who is able to access the endpoint based on built in SAP Mobile Platform security mechanisms.

When using Datajs to access an OData service from the Hybrid Web Container, you must employ POST tunneling to use the PUT, MERGE, and DELETE methods. There is an explanation of how to use POST tunneling with Datajs here: <http://datajs.codeplex.com/wikipage?title=Frequently%20Asked%20Questions#post-tunneling>.

Creating a Proxy Connection (Whitelisting)

Create a new connection in SAP Control Center to allow a proxy connection (authenticated or anonymous) through SAP Mobile Platform.

Note: When you set the proxy property with the endpoint address in the application template (either as part of the application creation or editing the application template created for that application), a proxy connection is generated automatically.

1. In the left navigation pane, expand the **Domains** folder, and select the default domain.
2. Select **Connections**.
3. In the right administration pane, select the **Connections** tab, and click **New**.
4. Enter a unique Connection pool name.
The Connection pool must have the same name as the application ID.
5. Select the Proxy **Connection pool** type.
6. Select the appropriate template for the data source target from the **Use template** menu.
7. Set the **Address** property by clicking the corresponding cell and entering the address of the proxy endpoint. For example, `http://odata.example.com/v2/Catalog/`
8. Configure the proxy properties you require. For a complete list, see *Proxy Properties in SAP Control Center for SAP Mobile Platform*

Note:

- To access an external service, you must configure the `http.proxyHost` and `http.proxyPort` properties during server configuration in **SAP Control Center > Configuration > General > Performance > Java Virtual Machine > Properties > JVM Properties > User options**. If you set or change the setting for `http.proxyHost` and `http.proxyPort`, you must restart the services using the stop/start service scripts. For more information, see *Administer > SAP Mobile Server > Configuring SAP Mobile Server to Securely Communicate With an HTTP Proxy in SAP Control Center for SAP Mobile Platform*.
- Ensure that enough work processes exist in both the Gateway system and in any SAP EIS systems (for example, SAP ERP or CRM) to handle the peak load. To throttle the

number of connections used by SAP Mobile Platform, use the Pool Size property for your Proxy connection pool on each SAP Mobile Server node.

- On a proxy connection, if the header for X-SUP-BACKEND-URL is not NULL, or EnableURLRewrite is false then no URL rewrite occurs for either the request or response content.
 - To access the external services, you must configure EnableHttpProxy = True, ProxyHost = proxy, ProxyPort = 8080 in the connection pool.
 - In REST services, the proxy URL is fetched from the application ID which is sent from the client device. The same application ID is also present in the connection pool. This proxy URL is used for request/response.
 - The "Username" and "Password" fields of a Proxy Connection Profile are only valid when Anonymous access is used: "AllowAnonymousAccess" is set to True. If set to False, the end user must provide basic authentication credentials.
-

9. Click **OK** to register the connection pool.

Datajs OData Client Authentication in Hybrid Apps

Several authentication schemes are available when accessing a protected OData service through an SAP Mobile Platform proxy, from a Hybrid App, in JavaScript using Datajs.

- **Basic authentication** – Provide a username and password to login. This method is available when connecting through HTTP and one-way HTTPS.
- **SSO token** – Provide an SSO token to login. This method is available when connecting through HTTP and HTTPS and a token validation service is available and configured.
- **X.509 Mutual authentication through intermediary** – Provide a forwarded client certificate to login using the SSL_CLIENT_CERT header name containing forwarded a PEM-encoded client certificate. This method is available only through an appropriately configured HTTPS listener. The certificate forwarder must have the "SUP Impersonator" role to be authorized for this type of login. The certificate of the actual "SUP Impersonator" user cannot be used as a regular user certificate.

In each case, if common additional JavaScript is required for every OData.read or OData.request call, this is best implemented in a Datajs custom HTTP client. This is a wrapper and extension of the OData.defaultHttpClient using the JavaScript proxy pattern. See <http://datajs.codeplex.com/wikipedia?title=Custom%20OData%20httpClient>

Basic Authentication

The Datajs JavaScript library internally uses the XMLHttpRequest (XHR) object to handle the underlying HTTP or HTTPS requests/responses on the client.

The XHR API's open method optionally accepts user name and password credentials passed through parameters. Likewise, the Datajs' request object can take user and password members that map to those parameters. If credentials are not passed and basic authentication is required, the client is challenged with HTTP status 401. If credentials are passed to the XHR object, internally it does not automatically send them on the first request. It submits the

credentials only if challenged. If this standard procedure is all that is required from the calling OData script, normally additional script can be avoided.

The below sample script shows possible alternative approaches for handling a 401 status manually, or, in cases where the authentication needs to be centralized.

```
/**
 * Sybase Hybrid App version 2.2
 *
 * Dataajs.SSO.js
 * This file will not be regenerated, and it is expected that the user
 * may want to
 * include customized code herein.
 *
 * The template used to create this file was compiled on Mon Jul 9
 * 19:54:04 CST 2012
 *
 * Copyright (c) 2012 Sybase Inc. All rights reserved.
 */

// Capture dataajs' current http client object.
var oldClient = OData.defaultHttpClient;

var sso_username = "";
var sso_password = "";
var sso_session = "";
var sso_token = "";

// Creates new client object that will attempt to handle SSO
// authentication, specifically SiteMinder login,
// in order to gain access to a protected URL.
var ssoClient = {
    request: function (request, success, error) {

        // For basic authentication, XMLHttpRequest.open method can
        // take varUser and varPassword parameters.
        // If the varUser parameter is null ("") or missing and the
        // site requires authentication, the
        // component displays a logon window. Although this method
        // accepts credentials passed via parameter,
        // those credentials are not automatically sent to the server
        // on the first request. The varUser and
        // varP assword parameters are not transmitted unless the
        // server challenges the client for credentials
        // with a 401 - Access Denied response. But SiteMinder may
        // require additional steps, so save for
        // later...
        if (request.user != undefined && request.password !=
        undefined) {
            sso_username = request.user;
            sso_password = request.password;
        }

        var onSuccess = function (data, response) {
            // Browser control will automatically cache cookies, with
```



```

possible token, for next time, so
// parsing Set-Cookie in HTTP response headers unnecessary
here.
    //var setCookieHeader = response.headers["Set-Cookie"];
    //var setCookies = [];
    //parseSetCookies(setCookieHeader, setCookies);

    //for(var i=0; i < setCookies.length; i++)
    //{
    //    if (setCookies[i].substr(0, 9) === "SMSESSION")
    //        sso_session = setCookies[i];
    //    else if (setCookies[i].substr(0, 9) === "MYSAPSSO2")
    //        sso_token = setCookies[i];
    //}

    // Call original success
    alert("Calling original success");
    success(data, response);
}

var onError = function (sso_error) {
    if (sso_error.response.statusCode == 0) {
        // Attempt to parse error from response.body, e.g. sent
        from SAP NetWeaver as HTML page.
        if (sso_error.response.body.indexOf("401") != -1 &&
            (sso_error.response.body.indexOf("Unauthorized") !=
-1 ||
-1)) {
            sso_error.response.body.indexOf("UNAUTHORIZED") !=
            alert("SSO challenge detected");
            sso_error.response.statusCode = 401;
        }
    }

    // Ensure valid response. Expecting either HTTP status 401
    for SMCHALLENGE or 302 for redirection.
    if (sso_error.response.statusCode != 401 &&
        sso_error.response.statusCode != 302) {
        alertText(sso_error.response.statusText);
        error(sso_error);
        return;
    }

    // 401 may include SMCHALLENGE=YES in Set-Cookie, so need
    to return along with Authorization
    // credentials to acquire SMSESSION cookie.
    if (sso_error.response.statusCode === 401) {
        // Browser control will automatically cache cookies,
        with possible token, for next time,
        // so parsing Set-Cookie in HTTP response headers
        unnecessary here.
        //var setCookieHeader =
sso_error.response.headers["Set-Cookie"];
        //var setCookies = [];
        //parseSetCookies(setCookieHeader, setCookies);
    }
}

```

```

        // Append existing headers.
        var newHeaders = [];
        if (request.headers) {
            for (name in request.headers) {
                newHeaders[name] = request.headers[name];
            }
        }
        // Browser control should include SMCHALLENGE cookie.
        //newHeaders["Cookie"] = "SMCHALLENGE=YES";
        var enc_username = window.btoa(sso_username);
        var enc_password = window.btoa(sso_password);
        var basic_auth = "Basic " + enc_username + ":" +
enc_password;
        newHeaders["Authorization"] = basic_auth;

        // Redo the OData request for the protected resource.
        var newRequest = {
            headers: newHeaders,
            requestUri: request.requestUri,
            method: request.method,
            user: sso_username,
            password: sso_password
        };

        oldClient.request(newRequest, onSuccess, error);
    }

    // 302 indicates that the requested information is located
at the URI specified in the Location
    // header. The default action when this status is received
is to follow the Location header
    // associated with the response. When the original request
method was POST, the redirected request
    // will use the GET method.
    if (sso_error.response.statusCode === 302) {
        // Get the redirection location.
        var siteminder_url =
sso_error.response.headers["Location"];

        // Open a connection to the redirect and load the login
form.
        // That screen can be used to capture the required form
fields.

        var httpRedirect = getXMLHttpRequest();

        httpRedirect.onload = function () {

            if (this.status < 200 || this.status > 299) {
                alert("Error: " + this.status);
                alertText(this.statusText);
                error({ message: this.statusText });
                return;
            }

            var sm_form_response = this.responseXML;

```

```

        var siteminder_tags = {};

        getSiteMinderTags(sm_form_response,
siteminder_tags);

        // Create the form data to post back to SiteMinder.
        Two ContentTypes are valid for sending
        // POST data. Default is application/x-www-form-
        urlencoded and form data is formatted
        // similar to typical querystring. Forms submitted
        with this content type are encoded as
        // follows: Control names and values are escaped.
        Space characters are replaced by `+',
        // reserved characters are escaped as described in
        [RFC1738], section 2.2:
        // non-alphanumeric characters are replaced by `
        %HH', representing ASCII code of character.
        // Line breaks are represented as CRLF pairs (i.e.,
        `%0D%0A'). Control names/values are
        // listed in order they appear in document. Name is
        separated from value by '=' and name/
        // value pairs are separated from each other by `&'.
        Alternative is multipart/form-data.
        //var formData = new FormData();
        var postData = "";

        for (var inputName in siteminder_tags) {
            if (inputName.substring(0, 2).toLowerCase() ===
"sm") {
                postData += inputName + "=" +
encodeURIComponent(siteminder_tags[inputName]) + "&";
                // formData.append(inputName,
siteminder_tags[inputName]);
            }
        }
        postData += "postpreservationdata=&";
        postData += "USER=" +
encodeURIComponent(sso_username) + "&";
        postData += "PASSWORD=" +
encodeURIComponent(sso_password);

        // Submit data back to SiteMinder.
        var httpLogin = getXMLHttpRequest();

        httpLogin.onload = function () {

            if (this.status < 200 || this.status > 299) {
                alert("Error: " + this.status);
                alertText(this.statusText);
                error({ message: this.statusText });
                return;
            }

            // Browser control should cache required cookies
            so no need to parse HTTP response
            // headers.

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
        //var sm_cookie_response = this.response;
        //var setCookieHeader =
this.getResponseHeader("Set-Cookie");
        //var setCookies = [];
        //parseSetCookies(setCookieHeader, setCookies);

        // Locate the URI to access next.
var newUrl = this.getResponseHeader("Location");

        // Append existing headers.
var newHeaders = [];
        if (request.headers) {
            for (name in request.headers) {
                newHeaders[name] = request.headers[name];
            }
        }
        // Browser control should include SMSESSION
cookie.
        //newHeaders["Cookie"] = setCookieHeader;

        // Redo the OData request for the protected
resource.
        var newRequest = {
            headers: newHeaders,
            requestUri: newUrl,
            method: request.method,
            user: sso_username,
            password: sso_password
        };

        oldClient.request(newRequest, onSuccess, error);
    }

    httpLogin.open("POST", siteminder_url, true);
    httpLogin.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
    httpLogin.withCredentials = "true";
    httpLogin.send(postData);
    //httpLogin.send(formData);
}

    httpRedirect.open("GET", siteminder_url, true);
    httpRedirect.responseType = "document";
    httpRedirect.send();

}

}

// Call back into the original http client.
var result = oldClient.request(request, success, onError);
return result;
}
};

// Parses Set-Cookie from header into array of setCookies.
function parseSetCookies(setCookieHeader, setCookies) {
```

```

if (setCookieHeader == undefined)
    return;

var cookieHeaders = setCookieHeader.split(", ");

// verify comma-delimited parse by ensuring '=' within each token
var len = cookieHeaders.length;
if (len > 0) {
    setCookies[0] = cookieHeaders[0];
}
var i, j;
for (i = 1, j = 0; i < len; i++) {
    if (cookieHeaders[i]) {
        var eqdex = cookieHeaders[i].indexOf('=');
        if (eqdex != -1) {
            var semidex = cookieHeaders[i].indexOf(';');
            if (semidex == -1 || semidex > eqdex) {
                setCookies[++j] = cookieHeaders[i];
            }
            else {
                setCookies[j] += ", " + cookieHeaders[i];
            }
        }
        else {
            setCookies[j] += ", " + cookieHeaders[i];
        }
    }
}
}

// Parses response HTML document and returns array of INPUT tags.
function getSiteMinderTags(response, tags) {

    var inputs = new Array();
    inputs = response.getElementsByTagName("input");

    // get the 'input' tags
    for (var i = 0; i < inputs.length; i++) {
        var element = inputs.item(i).outerHTML;
        var value = "";

        // filter out inputs with type=button
        var stridex = element.indexOf("type=");
        if (stridex != -1) {
            var typ = element.substring(stridex + 5);
            stridex = typ.indexOf(' ');
            typ = typ.substring(0, stridex);

            if (typ.toLowerCase() === "button") {
                continue;
            }
        }

        stridex = element.indexOf("value=")
        if (stridex != -1) {

```

```
        value = element.substring(stridex + 6);
        stridex = value.indexOf(' ');
        value = value.substring(0, stridex);
    }

    tags[inputs.item(i).name] = value;
}
}

function alertText(error) {

    var txt = JSON.stringify(error);
    alert("Error:\n" + txt);

    var length = txt.length;
    var sectionLength = 300;
    var index = Math.floor(length / sectionLength);
    for (i = 0; i <= index; i++) {
        var start = i * sectionLength;
        var end = (i + 1) * sectionLength;
        var segLength = sectionLength;
        if (end > length) segLength = length - start;
        alert(txt.substr(start, segLength));
    }
}

// Can either pass ssoClient explicitly, or set it globally for the
// page as the default:
OData.defaultHttpClient = ssoClient;
```

Authentication Against an OData Source

Hybrid Apps pass user name and password information using HTTP basic authentication, by setting the Authorization HTTP header.

It is recommended to use this in combination with SSL/TLS, otherwise user names and passwords are passed in cleartext. For example:

```
var strUsername = "odata";
var strPassword = "password";
var oHeaders = {};
oHeaders['Authorization'] = "Basic " + btoa(strUsername + ":" +
strPassword);
var request = {
    headers : oHeaders, // object that contains HTTP headers as
name value pairs
    requestUri : sUrl, // OData endpoint URI
    method : "GET"
};

OData.read( request, function (data, response) {

    // do something with the response
},
```

```
function( err ) {
  // handle error reading the data
};
```

SSO Token, Including SAP SSO2 and SiteMinder/Network Edge

As in basic authentication, the `Datajs` JavaScript library internally uses the `XmlHttpRequest` (XHR) object to handle the underlying HTTP or HTTPS requests/responses on the client.

From the XHR object's API, `Datajs` uses `setRequestHeader()` and `getAllResponseHeaders()` to send and read the HTTP headers in the request and response. For Single Sign-On and Network Edge authentication, issuers of SSO tokens, including SAP SSO2 logon tickets (MYSAPSSO2), as well as SiteMinder tokens (SMCHALLENGE, SMSESSION, and so on) normally use the "Set-Cookie" field in the HTTP header to send the token to the client, and expect the "Cookie" in the header to receive the token from the client.

However, these specific headers are omitted from JavaScript access. See the W3C spec (<http://www.w3.org/TR/XMLHttpRequest/>). Instead, these headers are designed to be controlled by the user agent, in this case the browser control hosted by the Hybrid Web Container, to protect the client from rogue sites. According to the W3C spec it is the job of the user agent to support HTTP state management: to persist, discard, and send cookies, as received in the Set-Cookie response header, and sent in the Cookie header, as applicable. One possible exception allows cookie handling in JavaScript to set up a CORS request on the client and server, using the XHR's "withCredentials" property.

Considering the reliance on the Hybrid Web Container-hosted browser control to handle the required SSO tokens, it is important to note the same origin policy surrounding automatic cookie management. That means from the client's perspective, the domain from where the cookie-based token originates must be the same as where it needs to be redirected to access the protected OData endpoint, such as the SAP NetWeaver Gateway, while authenticated. For the domain to be the same to the client, the URL pattern specifying transport protocol, servername, domain, and port number must match between token issuer and endpoint. This should be possible using proxy mappings in the Relay Server or reverse proxy.

Regarding the SiteMinder component of Network Edge, its Policy Server supports a variety of authentication schemes, including Basic Authentication and HTML Forms-based Authentication. The sample script below demonstrates an approach to handling a Basic 401 challenge from SiteMinder, as well as possible Forms authentication, involving HTTP status 302 indicating redirection. The script involving cookie handling is commented out and just informational, since this is managed by the user agent as described previously.

```
/**
 * SAP Hybrid App version 2.2
 *
 * Datajs.SSO.js
 * This file will not be regenerated, and it is expected that the user
 * may want to
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
* include customized code herein.
*
* The template used to create this file was compiled on Mon Jul 9
19:54:04 CST 2012
*
* Copyright (c) 2012 SAP Inc. All rights reserved.
*/

// Capture datajajs' current http client object.
var oldClient = OData.defaultHttpClient;

var sso_username = "";
var sso_password = "";
var sso_session = "";
var sso_token = "";

// Creates new client object that will attempt to handle SSO
authentication, specifically SiteMinder login,
// in order to gain access to a protected URL.
var ssoClient = {
    request: function (request, success, error) {

        // For basic authentication, XMLHttpRequest.open method can
take varUser and varPassword parameters.
        // If the varUser parameter is null ("" or missing and the
site requires authentication, the
        // component displays a logon window. Although this method
accepts credentials passed via parameter,
        // those credentials are not automatically sent to the server
on the first request. The varUser and
        // varP assword parameters are not transmitted unless the
server challenges the client for credentials
        // with a 401 - Access Denied response. But SiteMinder may
require additional steps, so save for
        // later...
        if (request.user != undefined && request.password !=
undefined) {
            sso_username = request.user;
            sso_password = request.password;
        }

        var onSuccess = function (data, response) {
            // Browser control will automatically cache cookies, with
possible token, for next time, so
            // parsing Set-Cookie in HTTP response headers unnecessary
here.
            //var setCookieHeader = response.headers["Set-Cookie"];
            //var setCookies = [];
            //parseSetCookies(setCookieHeader, setCookies);

            //for(var i=0; i < setCookies.length; i++)
            //{
            //    if (setCookies[i].substr(0, 9) === "SMSESSION")
            //        sso_session = setCookies[i];
            //    else if (setCookies[i].substr(0, 9) === "MYSAPSSO2")
            //        sso_token = setCookies[i];
        }
    }
}
```



```

    //}

    // Call original success
    alert("Calling original success");
    success(data, response);
}

var onError = function (sso_error) {
    if (sso_error.response.statusCode == 0) {
        // Attempt to parse error from response.body, e.g. sent
        // from SAP NetWeaver as HTML page.
        if (sso_error.response.body.indexOf("401") != -1 &&
            (sso_error.response.body.indexOf("Unauthorized") !=
-1 ||
-1)) {
                sso_error.response.body.indexOf("UNAUTHORIZED") !=
-1)) {
                    alert("SSO challenge detected");
                    sso_error.response.statusCode = 401;
                }
            }
        }

    // Ensure valid response. Expecting either HTTP status 401
    // for SMCHALLENGE or 302 for redirection.
    if (sso_error.response.statusCode != 401 &&
        sso_error.response.statusCode != 302) {
        alertText(sso_error.response.statusText);
        error(sso_error);
        return;
    }

    // 401 may include SMCHALLENGE=YES in Set-Cookie, so need
    // to return along with Authorization
    // credentials to acquire SMSESSION cookie.
    if (sso_error.response.statusCode === 401) {
        // Browser control will automatically cache cookies,
        // with possible token, for next time,
        // so parsing Set-Cookie in HTTP response headers
        // unnecessary here.
        //var setCookieHeader =
        sso_error.response.headers["Set-Cookie"];
        //var setCookies = [];
        //parseSetCookies(setCookieHeader, setCookies);

        // Append existing headers.
        var newHeaders = [];
        if (request.headers) {
            for (name in request.headers) {
                newHeaders[name] = request.headers[name];
            }
        }
        // Browser control should include SMCHALLENGE cookie.
        //newHeaders["Cookie"] = "SMCHALLENGE=YES";
        var enc_username = window.btoa(sso_username);
        var enc_password = window.btoa(sso_password);
        var basic_auth = "Basic " + enc_username + ":" +

```

```

enc_password;
        newHeaders["Authorization"] = basic_auth;

        // Redo the OData request for the protected resource.
        var newRequest = {
            headers: newHeaders,
            requestUri: request.requestUri,
            method: request.method,
            user: sso_username,
            password: sso_password
        };

        oldClient.request(newRequest, onSuccess, error);
    }

    // 302 indicates that the requested information is located
    at the URI specified in the Location
    // header. The default action when this status is received
    is to follow the Location header
    // associated with the response. When the original request
    method was POST, the redirected request
    // will use the GET method.
    if (sso_error.response.statusCode === 302) {
        // Get the redirection location.
        var siteminder_url =
sso_error.response.headers["Location"];

        // Open a connection to the redirect and load the login
form.
        // That screen can be used to capture the required form
fields.
        var httpRedirect = getXMLHttpRequest();

        httpRedirect.onload = function () {

            if (this.status < 200 || this.status > 299) {
                alert("Error: " + this.status);
                alertText(this.statusText);
                error({ message: this.statusText });
                return;
            }

            var sm_form_response = this.responseXML;
            var siteminder_tags = {};

            getSiteMinderTags(sm_form_response,
siteminder_tags);

            // Create the form data to post back to SiteMinder.
            Two ContentTypes are valid for sending
            // POST data. Default is application/x-www-form-
            urlencoded and form data is formatted
            // similar to typical querystring. Forms submitted
            with this content type are encoded as
            // follows: Control names and values are escaped.
            Space characters are replaced by `+',

```

```

        // reserved characters are escaped as described in
[RFC1738], section 2.2:
        // non-alphanumeric characters are replaced by `
%HH', representing ASCII code of character.
        // Line breaks are represented as CRLF pairs (i.e.,
`%0D%0A'). Control names/values are
        // listed in order they appear in document. Name is
separated from value by '=' and name/
        // value pairs are separated from each other by `&'.
Alternative is multipart/form-data.
        //var formData = new FormData();
        var postData = "";

        for (var inputName in siteminder_tags) {
            if (inputName.substring(0, 2).toLowerCase() ===
"sm") {
                postData += inputName + "=" +
encodeURIComponent(siteminder_tags[inputName]) + "&";
                // formData.append(inputName,
siteminder_tags[inputName]);
            }
            postData += "postpreservationdata=&";
            postData += "USER=" +
encodeURIComponent(sso_username) + "&";
            postData += "PASSWORD=" +
encodeURIComponent(sso_password);

            // Submit data back to SiteMinder.
            var httpLogin = getXMLHttpRequest();

            httpLogin.onload = function () {

                if (this.status < 200 || this.status > 299) {
                    alert("Error: " + this.status);
                    alertText(this.statusText);
                    error({ message: this.statusText });
                    return;
                }

                // Browser control should cache required cookies
so no need to parse HTTP response
                // headers.
                //var sm_cookie_response = this.response;
                //var setCookieHeader =
this.getResponseHeader("Set-Cookie");
                //var setCookies = [];
                //parseSetCookies(setCookieHeader, setCookies);

                // Locate the URI to access next.
                var newUrl = this.getResponseHeader("Location");

                // Append existing headers.
                var newHeaders = [];
                if (request.headers) {
                    for (name in request.headers) {

```

```

        newHeaders[name] = request.headers[name];
    }
}
// Browser control should include SMSESSION
cookie.
//newHeaders["Cookie"] = setCookieHeader;
// Redo the OData request for the protected
resource.
var newRequest = {
    headers: newHeaders,
    requestUri: newUrl,
    method: request.method,
    user: sso_username,
    password: sso_password
};

oldClient.request(newRequest, onSuccess, error);
}

httpLogin.open("POST", siteminder_url, true);
httpLogin.setRequestHeader("Content-Type",
"application/x-www-form-urlencoded");
httpLogin.withCredentials = "true";
httpLogin.send(postData);
//httpLogin.send(formData);
}

httpRedirect.open("GET", siteminder_url, true);
httpRedirect.responseType = "document";
httpRedirect.send();

}
}

// Call back into the original http client.
var result = oldClient.request(request, success, onError);
return result;
}
};

// Parses Set-Cookie from header into array of setCookies.
function parseSetCookies(setCookieHeader, setCookies) {

    if (setCookieHeader == undefined)
        return;

    var cookieHeaders = setCookieHeader.split(", ");

    // verify comma-delimited parse by ensuring '=' within each token
    var len = cookieHeaders.length;
    if (len > 0) {
        setCookies[0] = cookieHeaders[0];
    }
    var i, j;
    for (i = 1, j = 0; i < len; i++) {

```

```

        if (cookieHeaders[i]) {
            var eqdex = cookieHeaders[i].indexOf('=');
            if (eqdex != -1) {
                var semdex = cookieHeaders[i].indexOf(';');
                if (semdex == -1 || semdex > eqdex) {
                    setCookies[++j] = cookieHeaders[i];
                }
                else {
                    setCookies[j] += ", " + cookieHeaders[i];
                }
            }
            else {
                setCookies[j] += ", " + cookieHeaders[i];
            }
        }
    }
}

// Parses response HTML document and returns array of INPUT tags.
function getSiteMinderTags(response, tags) {

    var inputs = new Array();
    inputs = response.getElementsByTagName("input");

    // get the 'input' tags
    for (var i = 0; i < inputs.length; i++) {
        var element = inputs.item(i).outerHTML;
        var value = "";

        // filter out inputs with type=button
        var stridex = element.indexOf("type=");
        if (stridex != -1) {
            var typ = element.substring(stridex + 5);
            stridex = typ.indexOf(' ');
            typ = typ.substring(0, stridex);

            if (typ.toLowerCase() === "button") {
                continue;
            }
        }

        stridex = element.indexOf("value=")
        if (stridex != -1) {
            value = element.substring(stridex + 6);
            stridex = value.indexOf(' ');
            value = value.substring(0, stridex);
        }

        tags[inputs.item(i).name] = value;
    }
}

function alertText(error) {

    var txt = JSON.stringify(error);
    alert("Error:\n" + txt);
}

```

```
var length = txt.length;
var sectionLength = 300;
var index = Math.floor(length / sectionLength);
for (i = 0; i <= index; i++) {
    var start = i * sectionLength;
    var end = (i + 1) * sectionLength;
    var segLength = sectionLength;
    if (end > length) segLength = length - start;
    alert(txt.substr(start, segLength));
}

// Can either pass ssoClient explicitly, or set it globally for the
page as the default:
OData.defaultHttpClient = ssoClient;
```

Server Certificate Validation Over HTTPS

In this pattern, which uses the `CertificateAuthenticationLoginModule`, the server sends the client a certificate with which to authenticate itself.

The client uses the certificate to authenticate the identity the certificate claims to represent. An SSL-enabled client goes through these steps to authenticate a server's identity:

1. Is today's date within the valid period?
2. Is the issuing certificate authority (CA) a trusted one? Each SSL-enabled client maintains a list of trusted CA certificates. This list determines which server certificates the client accepts. Validation continues if the distinguished name (DN) of the issuing CA matches the DN of a certificate authority on the client's list of trusted certificate authorities.
3. Does the issuing certificate authority's public key validate the issuer's digital signature?
4. Does the domain name in the server's certificate match the domain name of the server itself?
5. The server is authenticated. The client proceeds with the SSL handshake. If the client does not get to step 5 for any reason, the server that is identified by the certificate cannot be authenticated, and the user is warned of the problem and informed that an encrypted and authenticated connection cannot be established.

Similar to cookie-based tokens, certificate authentication is also outside the scope of pure JavaScript which has no access to certificates, and similarly falls under the control of the user agent, in this case again the browser control, and its interface directly with the user.

X.509 SSO Authentication

For certificate based SSO authentication, due to the restriction from handling certificates in pure JavaScript, a native counterpart on the device must be interfaced, such as the Hybrid Web Container, using its existing `Certificate.js`.

In this sample script, a Datajs custom HTTP client is used to encapsulate the client certificate component of certificate based SSO. You can provision signed certificate from a local file, a

server, or from Afaria, based on the device platform, using the existing Certificate API. You can choose to set the results of the API call as the password.

```
/**
 * SAP Hybrid App version 2.2
 *
 * Datajcs.Certificate.js
 * This file will not be regenerated, and it is expected that the user
may want to
 * include customized code herein.
 *
 * The template used to create this file was compiled on Mon Aug 23
16:43:02 CST 2012
 *
 * Copyright (c) 2012 SAP Inc. All rights reserved.
 */

// Capture datajcs' current http client object.
var oldClient = OData.defaultHttpClient;

var cert_username = "";
var cert_password = "";

// Creates new client object that will attempt to handle Certificate
authentication.
var certClient = {
  request: function (request, success, error) {

    if (request.requestUri.substr(0, 8) === "https://")
    {
      if (request.password !== undefined)
      {
        // The following script gets the signed certificate data
for the first
        // p12 file found on the sdcard
        var certStore = CertificateStore.getDefault();
        var certPaths =
certStore.listAvailableCertificatesFromFileSystem("/sdcard/",
"p12");
        var cert =
certStore.getSignedCertificateFromFile(certPaths[0],
request.password);

        var cert_username = cert.subjectCN;
        var cert_password = cert.signedCertificate;

        // Redo the OData request for the protected resource
        var newRequest = {
          headers : request.headers,
          requestUri : request.requestUri,
          method : request.method,
          user : cert_username,
          password : cert_password
        };
      }
    }
  }
};
```

```
        // Call back into the original http client.
        return oldClient.request(newRequest, success, error);
    }
}

return oldClient.request(request, success, error);
}
};

// Can either pass certClient explicitly, or set it globally for the
page as the default:
OData.defaultHttpClient = certClient;
```

When sending a forwarded client certificate through an intermediary, set the value to “SSL_CLIENT_CERT” in the XHR’s HTTP request header, as shown in this example:

```
/**
 * SAP Hybrid App version 2.2
 *
 * Dataajs.Certificate.js
 * This file will not be regenerated, and it is expected that the user
may want to
 * include customized code herein.
 *
 * The template used to create this file was compiled on Mon Aug 23
16:43:02 CST 2012
 *
 * Copyright (c) 2012 SAP Inc. All rights reserved.
 */

// Capture dataajs' current http client object.
var oldClient = OData.defaultHttpClient;

// Creates new client object that will attempt to handle Certificate
authentication.
var certClient = {
    request: function (request, success, error) {

        if (request.requestUri.substr(0, 8) === "https://")
        {
            if (request.user != undefined && request.password !=
undefined)
            {
                // The following script gets the signed certificate
data for the first
// p12 file found on the sdcard
var certStore = CertificateStore.getDefault();
var certPaths = certStore.listAvailableCertificatesFromFileSystem("/
sdcard/", "p12");
var cert = certStore.getSignedCertificateFromFile(certPaths [0] ,
request.password);

                // Append existing headers.
                var newHeaders = [];
                if (request.headers) {
```



```

        for (name in request.headers) {
            newHeaders[name] = request.headers[name];
        }
    }
    //
    newHeaders["SSL_CLIENT_CERT"] = cert.signedCertificate;

    // Redo the OData request for the protected resource
    var newRequest = {
        headers : newHeaders,
        requestUri : request.requestUri,
        method : request.method,
        user : request.user,
        password : request.password
    };

    // Call back into the original http client.
    return oldClient.request(newRequest, success, error);
    }
}

return oldClient.request(request, success, error);
}
};

// Can either pass certClient explicitly, or set it globally for the
page as the default:
OData.defaultHttpClient = certClient;

```

Sending Requests Over HTTPS

Datajs custom clients can replace the default `odata.DefaultHttpClient` with `overrideHttpClientForDatajs` to send requests to SAP Mobile Server through an HTTPS connection.

```

var oDataDefaultHttpClient;

function overrideHttpClientForDatajs()
{
    if (HttpsConnection.supportHttpsInBrowser === true )
    {
        oDataDefaultHttpClient = odata.defaultHttpClient;
        OData.defaultHttpClient = new
        {
            request: function (request, success, error)
            {
                {
                    if (request.requestURI.search("/https/i"))
                    {
                        // invoke HTTPS proxy api
                        // return result
                    }
                }
            }
            else
            {
                // using original http client
            }
        }
    }
}

```

```
oDataDefaultHttpClient.request( request, success,
error );
    }
    }
    }
}
```

Implementing Push

The backend OData source can proactively send notifications to Hybrid Apps.

SAP Mobile Platform enables this by exposing an HTTP based push interface `http://supserver:port/notifications/ApplicationConnectionID`.

The Hybrid App must inform the backend of its `ApplicationConnectionID`, usually on startup. You can obtain this by using the `hwc.getApplicationConnectionID()` JavaScript API. The backend service exposes an endpoint where said `ApplicationConnectionID` can be sent when the Hybrid App starts up or "subscribes." When the push notification is received, it can be handled in native code or JavaScript.

Enabling the Datajs Library on Windows Mobile

To enable the `datajs-<version>.js` library on Windows Mobile 6.0 and Windows Mobile 6.1, you must add some custom code into the file where the Hybrid App is first launched.

For Windows Mobile 6.5, you need only to include the `datajs-<version>.js` library in your HTML file.

1. Open the JavaScript file where the Hybrid App is first launched, for example, `Custom.js`, which is located in `<SMP_HOME>\MobileSDK<version>\HybridApp\API\AppFramework`.
2. Add this code:

```
///Begin, This code enable datajs library on Windows 6.0 and
Windows6.1
window.oldActiveXObject = window.ActiveXObject;
window.ActiveXObject = function(id) {
try{ return new window.oldActiveXObject(id); }
catch (exception) {
if(isWindowsMobile()){
try{
if(id == "Msxml2.XMLHTTP.6.0" || id == "Msxml2.XMLHTTP.3.0")
{ return new window.oldActiveXObject("Microsoft.XMLHTTP"); }
if(id == "Msxml2.DOMDocument.6.0" || id == "Msxml2.DOMDocument.
3.0"){ return new window.oldActiveXObject("Microsoft.XMLDOM"); }
}
catch(e){ throw e; }
}
throw exception;
}
```

```
};
//End
```

3. Save the file.
4. Rebuild the Hybrid App project.

Hybrid Web Container and Hybrid App JavaScript APIs

The container and framework JavaScript APIs provide functionality that the Hybrid Apps can access.

Hybrid Web Container JavaScript APIs

The files where the Hybrid Web Container JavaScript APIs are defined are located in <SMP_HOME>\MobileSDK<version>\HybridApp\API\Container. The generated JavaScript API reference documents are located in <SMP_HOME>\MobileSDK<version>\HybridApp\API\API.

Class	Description	Defined in
<code>hwc.CallbackSet ()</code>	Use for event handlers that are asynchronous.	<code>Callbacks.js</code>
<code>hwc.CertificateStore</code>	Create a user interface in HTML and JavaScript that uses X.509 certificates as the Hybrid App credentials.	<code>Certificate.js</code>
<code>hwc.ConnectionSettings</code>	The JavaScript class for the Hybrid Web Container connection settings manages the connection between applications and the server.	<code>hwc-api.js</code>
<code>hwc.CustomIcon</code>	The JavaScript class for the Hybrid Web Container custom icon, lists custom icons.	<code>hwc-api.js</code>
<code>hwc.e2eTrace</code>	Allows for an end to end trace of data communication from the client to the back-end.	<code>hwc-api.js</code>
<code>hwc.getExternalResource</code>	Access resources on external HTTP servers.	<code>ExternalResource.js</code>

Develop Hybrid Apps Using Third-party Web Frameworks

Class	Description	Defined in
<code>hwc.getCurrentLocale</code>	The date/time functions allow you to extract and format the date and time for the Hybrid App.	<code>Timezone.js</code>
<code>hwc.getPicture</code>	Provides access to the device's default camera application or device's photo library for retrieving a picture asynchronously.	<code>Camera.js</code>
<code>hwc.HybridApp</code>	Javascript class for the Hybrid App object. Lists installed Hybrid Apps.	<code>hwc-api.js</code>
<code>hwc.LogEntry</code>	Javascript class for LogEntry object.	<code>hwc-api.js</code>
<code>hwc.MediaCache</code>	Used within the JavaScript to wrap the source of an image element. Fetches media content from a cache or the server using a URI.	<code>hwc-api.js</code>
<code>hwc.MenuItemCollection</code>	Represents a collection of menu items.	<code>hwc-comms.js</code>
<code>hwc.Message</code>	This is the class to encapsulate an incoming message object. When a new message arrives, a notification is sent to users through custom code.	<code>hwc-api.js</code>
<code>hwc.MessageFilter</code>	This is the class to encapsulate a filter for messages.	<code>hwc-api.js</code>

Class	Description	Defined in
<code>hwc.perf</code>	The performance library allows you to instrument your application code and collect performance counters when executing the application on the device. Results are reported in a log file on the SD-card (BlackBerry and Android), or in the sandbox (iOS). The results can also be read in the domain log by calling <code>GetTrace</code> for the application connection in SAP Control Center.	<code>hwc-api.js</code>
<code>hwc.SUPStorage</code>	Constructs a new storage area identified by a storage key.	<code>SUPStorage.js</code>
<code>Resources</code>	Access localized string resources.	<code>Resources.js</code>

Hybrid App Framework JavaScript APIs

The files where the Hybrid App framework JavaScript APIs are defined are located in `<SMP_HOME>\MobileSDK<version>\HybridApp\API\AppFramework`.

Class	Description	Defined in
<code>doOnlineRequest</code>	Allows an operation or object query to be invoked.	<code>API.js</code>
<code>MessageValue</code>	Message value object that stores a key-value pair from a message sent to or from the server and the Hybrid App.	<code>WorkflowMessage.js</code>
<code>MessageValueCollection</code>	Message value collection object that stores a container node from a message sent to or from the server and the Hybrid App.	<code>WorkflowMessage.js</code>
<code>WorkflowMessage</code>	Access the Hybrid App message data functions.	<code>WorkflowMessage.js</code>

anonymous namespace

Used to group anonymous objects and callback functions used as method parameters.

Methods and fields in this namespace cannot be instantiated. Used for API docs generation only.

Classes

Name	Description
<i>anonymous.AppLogErrorCallbackParameter</i> on page 70	Object used in <i>anonymous.getLogEntriesErrorCallback</i> on page 78 and <i>anonymous.startOrStopLogListenerErrorCallback</i> on page 83 functions.
<i>anonymous.sendRequestErrorCBParameter</i> on page 70	Object used in <i>anonymous.sendRequestErrorCB</i> on page 82 function.
<i>anonymous.sendRequestSuccessCBParameter</i> on page 71	Object used in <i>anonymous.sendRequestSuccessCB</i> on page 83 function.

Members

Name	Description
<i>options</i> on page 72	Options object used with the <i>getExternalResource</i> function.
<i>PictureOptions</i> on page 72	Options object that is used with the <i>hwc.getPicture</i> on page 192 method.

Methods

Name	Description
<i>abort()</i> on page 72	JavaScript function to abort the HTTP(S) request
<i>alertDialogCallbackFunction(message)</i> on page 73	A callback function invoked when <i>hwc.log</i> on page 208 is invoked with true for the <i>notifyUser</i> parameter.
<i>AppInstallationListener(event, moduleId, version, moduleName)</i> on page 73	Callback function that will be invoked on hybrid app installation events.
<i>AppInstallationListener(event, moduleId, version, moduleName)</i> on page 74	Callback function that will be invoked on hybrid app installation events.

<i>AppInstallationListener(notifications, event, moduleId, version, moduleName, designerVersion, containerVersion)</i> on page 74	Callback function that will be invoked when push notifications are available.
<i>ApplicationListener(event, moduleId, version)</i> on page 75	Callback function that will be invoked on hybrid app events.
<i>complete(resultXHR)</i> on page 76	Callback function used in the Options object.
<i>ConnectionStateListener(event, errorCode, errorMessage)</i> on page 77	Callback function that will be invoked when the connection state changes.
<i>errorCallbackFunction(errorMessage)</i> on page 77	A callback function invoked if there is an error.
<i>genericCallbackFunction()</i> on page 78	A generic callback function that takes no parameters.
<i>getLogEntriesErrorCallback(data)</i> on page 78	Callback function that will be invoked when <i>AppLog.getLogEntries</i> on page 91 fails.
<i>getLogEntriesSuccessCallback(data)</i> on page 78	Callback function that will be invoked with all the entries in the app log.
<i>logListener(date, event, message)</i> on page 79	Callback function that will be invoked when events are logged to the app log.
<i>LogListener(milliseconds, event, optional-String)</i> on page 79	Callback function that will be invoked when events are logged to the event log.
<i>MessageListener(flag, msgId)</i> on page 80	Callback function that will be invoked on message events.
<i>onGetPictureError(err)</i> on page 81	Camera
<i>onGetPictureSuccess(filename, response)</i> on page 82	User provided function that will be invoked when the <i>hwc.getPicture</i> on page 192 function is successful.
<i>sendRequestErrorCB(data)</i> on page 82	Callback function that will be invoked <i>HttpsConnection.get()/sendRequest()</i> failed.
<i>sendRequestSuccessCB(data)</i> on page 83	Callback function that will be invoked <i>HttpsConnection.get()/sendRequest()</i> succeeded.
<i>startOrStopLogListenerErrorCallback(data)</i> on page 83	Callback function that will be invoked upon failure to start a log listener via <i>AppLog.startLogListener</i> on page 92, or upon failure to removing a log listener via <i>AppLog.stopLogListener</i> on page 95.

<p><i>startOrStopLogListenerSuccessCallback()</i> on page 84</p>	<p>Callback function that will be invoked upon successfully starting a log listener via <i>AppLog.startLogListener</i> on page 92, or upon successfully removing a log listener via <i>AppLog.stopLogListener</i> on page 95.</p>
--	---

Source

Camera.js, line 266 on page 266.

anonymous.AppLogErrorCallbackParameter class

Object used in *anonymous.getLogEntriesErrorCallback* on page 78 and *anonymous.startOrStopLogListenerErrorCallback* on page 83 functions.

Syntax

`new AppLogErrorCallbackParameter()`

Properties

Name	Type	Description
<i>errorCode</i>	number	Predefined error code
<i>description</i>	string	The description of the error

Source

Plugins/AppLog/applog.js, line 479 on page 530.

anonymous.sendRequestErrorCBParameter class

Object used in *anonymous.sendRequestErrorCB* on page 82 function.

Syntax

`new sendRequestErrorCBParameter()`

Properties

Name	Type	Argument	Description
<i>errorCode</i>	number		Predefined error code
<i>description</i>	string		The description of the error

<i>nativeErrorCode</i>	number	<optional>	The native error code reported from Afaria, device, etc (optional)
------------------------	--------	------------	--

Source

Plugins/HttpsProxy/https-proxy.js, line 506 on page 558.

anonymous.sendRequestSuccessCBParameter class

Object used in *anonymous.sendRequestSuccessCB* on page 83 function.

Syntax

```
new sendRequestSuccessCBParameter()
```

Properties

Name	Type	Argument	Description
<i>status</i>	number		The HTTP status code
<i>headers</i>	object		An object that contains headerKey = value pairs.
<i>responseText</i>	string	<optional>	The text response. This parameter is present only if the response is a text response.
<i>responseBase64</i>	string	<optional>	Base64 encoded representation of the binary response. This parameter is included only if the response is a binary response.
<i>clientError</i>	object	<optional>	An optional object that contains the authentication error. It is an object of <i>anonymous.sendRequestErrorCBParameter</i> on page 70.

Source

Plugins/HttpsProxy/https-proxy.js, line 504 on page 558.

options member

Options object used with the `getExternalResource` function.

Supported options are:

- `method`: one of GET, PUT, DELETE, HEAD, OPTIONS, or POST. The default is GET.
- HTTP and HTTPS urls are supported.
- `async`: request should be sent asynchronously. The default is true.
- `headers`: request headers to be sent with request.
- `data`: data to be sent. If this is an array, it is converted to a query string. For a GET request, this is added to the end of the URL.
- *anonymous.complete* on page 76 is a callback function that will be invoked with the `resultXHR` when this method completes

Syntax

```
<static> options
```

Source

ExternalResource.js, line 270 on page 292.

PictureOptions member

Options object that is used with the `hwc.getPicture` on page 192 method.

Contains 2 fields that can be specified.

- `sourceType`: One of `hwc.Picture.SourceType` values
- `destinationType`: One of `hwc.Picture.DestinationType` values

Syntax

```
<static> PictureOptions
```

See

`hwc.getPicture` for an example.

Source

Camera.js, line 272 on page 266.

abort() method

JavaScript function to abort the HTTP(S) request

Syntax

```
<static> abort()
```

Source

Plugins/HttpsProxy/https-proxy.js, line 509 on page 558.

alertDialogCallbackFunction(message) method

A callback function invoked when *hwc.log* on page 208 is invoked with true for the `notifyUser` parameter.

This callback should notify the user of the log message in an appropriate manner.

Syntax

```
<static> alertDialogCallbackFunction( message )
```

Parameters

Name	Type	Description
<i>message</i>	string	The message that the user should be notified of.

Source

hwc-comms.js, line 1608 on page 489.

AppInstallationListener(event, moduleId, version, moduleName) method

Callback function that will be invoked on hybrid app installation events.

App installation listeners can be added with *hwc.addAppInstallationListener* on page 149.

Syntax

```
<static> AppInstallationListener( event, moduleId, version, moduleName )
```

Parameters

Name	Type	Description
<i>event</i>	number	A number indicating the event (will be either <i>hwc.INSTALLATION_BEGIN</i> on page 132 or <i>hwc.INSTALLATION_END</i> on page 132).
<i>moduleId</i>	string	The module ID of the hybrid app the event is about.
<i>version</i>	string	The version of the hybrid app the event is about.

<i>moduleName</i>	string	The display name of the hybrid app the event is about.
-------------------	--------	--

Source

hwc-api.js, line 3673 on page 428.

AppInstallationListener(event, moduleId, version, moduleName) method

Callback function that will be invoked on hybrid app installation events.

Syntax

`<static> AppInstallationListener(event, moduleId, version, moduleName)`

Parameters

Name	Type	Description
<i>event</i>	Integer	Installation flags including, BEGIN(1), END(2)
<i>moduleId</i>	String	Optional Module Id
<i>version</i>	String	Optional Module version
<i>moduleName</i>	String	Optional Module display name

Source

hwc-api.js, line 3676 on page 428.

AppInstallationListener(notifications, event, moduleId, version, moduleName, designerVersion, containerVersion) method

Callback function that will be invoked when push notifications are available.

Push notification listeners can be added with *hwc.addPushNotificationListener* on page 157.

Syntax

`<static> AppInstallationListener(notifications, event, moduleId, version, moduleName, designerVersion, containerVersion) {number}`

Parameters

Name	Type	Description
<i>notifications</i>	Array	An array of notifications.

<i>event</i>	Integer	Installation flags including, BEGIN(1), END(2), FAIL(3)
<i>moduleId</i>	String	Optional Module Id
<i>version</i>	String	Optional Module version
<i>moduleName</i>	String	Optional Module display name
<i>designerVersion</i>	String	Optional Version of designer used to create app
<i>containerVersion</i>	String	Optional Version of hybrid web container

Returns

A number indicating whether other push notification listeners should be called after this one. Must be either *hwc.NOTIFICATION_CANCEL* on page 137 (if no more listener callbacks should be called) or *hwc.NOTIFICATION_CONTINUE* on page 137 (if more listener callbacks should be called). Callback function that will be invoked on hybrid app installation events.

Type:

number

Source

hwc-api.js, line 3675 on page 428.

ApplicationListener(event, moduleId, version) method

Callback function that will be invoked on hybrid app events.

Application listeners can be added with *hwc.addAppListener* on page 150.

Syntax

<static> ApplicationListener(*event*, *moduleId*, *version*)

Parameters

Name	Type	Description
------	------	-------------

<i>event</i>	number	A number indicating what event has taken place (will be one of <i>hwc.APP_REFRESH</i> on page 126, <i>hwc.APP_ADDED</i> on page 126, <i>hwc.APP_UPDATED</i> on page 127, <i>hwc.APP_REMOVED</i> on page 127).
<i>moduleId</i>	number	The module id of the hybrid app the event is about.
<i>version</i>	number	module The version of the hybrid app the event is about.

Source

hwc-api.js, line 3678 on page 428.

complete(resultXHR) method

Callback function used in the Options object.

Syntax

<static> complete(*resultXHR*)

Parameters

Name	Type	Description
<i>resultXHR</i>	object	<p>the response object.</p> <hr/> <p>The fields/methods available on resultXHR are</p> <ol style="list-style-type: none"> 1. status 2. statusText 3. responseText 4. getReponseHeader(key) 5. getAllResponesHeaders() <p>These fields and methods are not supported for resultXHR:</p> <ul style="list-style-type: none"> • open()

Source

ExternalResource.js, line 272 on page 292.

ConnectionStateListener(event, errorCode, errorMessage) method

Callback function that will be invoked when the connection state changes.

Connection listeners can be added with *hwc.addConnectionListener* on page 151.

Syntax

```
<static> ConnectionStateListener( event, errorCode, errorMessage )
```

Parameters

Name	Type	Description
<i>event</i>	number	A number indicating the event that occurred (will be <i>hwc.CONNECTED</i> on page 127 or <i>hwc.DISCONNECTED</i> on page 130).
<i>errorCode</i>	number	An error code (0 indicating success).
<i>errorMessage</i>	string	Text of the error message. Will be empty if there is no error.

Source

hwc-api.js, line 3669 on page 428.

errorCallbackFunction(errorMessage) method

A callback function invoked if there is an error.

Syntax

```
<static> errorCallbackFunction( errorMessage )
```

Parameters

Name	Type	Description
<i>errorMessage</i>	string	The message describing the error.

Source

hwc-comms.js, line 1610 on page 489.

genericCallbackFunction() method

A generic callback function that takes no parameters.

Used to execute code when a certain event occurs.

Syntax

`<static> genericCallbackFunction()`

Source

hwc-comms.js, line 1612 on page 489.

getLogEntriesErrorCallback(data) method

Callback function that will be invoked when *AppLog.getLogEntries* on page 91 fails.

Syntax

`<static> getLogEntriesErrorCallback(data)`

Parameters

Name	Type	Description
<i>data</i>	<i>anonymous.AppLogErrorCallbackParameter</i> on page 70	The error object.

Source

Plugins/AppLog/applog.js, line 473 on page 530.

getLogEntriesSuccessCallback(data) method

Callback function that will be invoked with all the entries in the app log.

There will be one *AppLog.LogEntry* on page 92 object for each line in the app log. Log entries can be retrieved with *AppLog.getLogEntries* on page 91.

Syntax

`<static> getLogEntriesSuccessCallback(data)`

Parameters

Name	Type	Description
<i>data</i>	<i>AppLog.LogEntry[]</i>	An array of <i>AppLog.LogEntry</i> objects.

Source

Plugins/AppLog/applog.js, line 471 on page 530.

logListener(date, event, message) method

Callback function that will be invoked when events are logged to the app log.

Log listeners can be added with *AppLog.startLogListener* on page 92.

Syntax

```
<static> logListener( date, event, message )
```

Parameters

Name	Type	Description
<i>date</i>	Date	The date of the log entry.
<i>event</i>	number	The event ID of the log entry (will be one of the AppLog status events, or possibly a custom value).
<i>message</i>	string	The string carrying the message of the log entry.

Source

Plugins/AppLog/applog.js, line 481 on page 530.

LogListener(milliseconds, event, optionalString) method

Callback function that will be invoked when events are logged to the event log.

Log listeners can be added with *hwc.addLogListener* on page 153.

Syntax

```
<static> LogListener( milliseconds, event, optionalString )
```

Parameters

Name	Type	Description
<i>milliseconds</i>	number	The date of the log message represented in milliseconds.

<i>event</i>	number	A number that represents which category this event falls under (It will be one of <i>hwc.CONNECTION_ERROR</i> on page 128, <i>hwc.CONNECTION_OTHER</i> on page 129, <i>hwc.CONNECTION_CONNECTED</i> on page 128, <i>hwc.CONNECTION_DISCONNECTED</i> on page 128, <i>hwc.CONNECTION_RETRIEVED_ITEMS</i> on page 129).
<i>optionalString</i>	string	The string carrying the message of the log event.

Source

hwc-api.js, line 3671 on page 428.

MessageListener(flag, msgId) method

Callback function that will be invoked on message events.

Message listeners can be added with *hwc.addMessageListener* on page 155.

Syntax

<static> MessageListener(*flag*, *msgId*)

Parameters

Name	Type	Description
<i>flag</i>	number	A number indicating which message event occurred (will be one of <i>hwc.MSG_ADDED</i> on page 135, <i>hwc.MSG_REMOVED</i> on page 136, <i>hwc.MSG_UPDATED</i> on page 137, <i>hwc.MSG_REFRESH</i> on page 136).
<i>msgId</i>	number	The message id of the affected message.

Source

hwc-api.js, line 3680 on page 428.

onGetPictureError(err) method

Camera

Syntax`<static> onGetPictureError(err)`*Parameters*

Name	Type	Description
<i>err</i>	number	<p>the error code returned. Possible values are</p> <ol style="list-style-type: none"> 1. <code>PictureError.NO_ERROR = 0</code>; 2. <code>PictureError.NOT_SUPPORTED = -1</code>; <code>getPicture()</code> not implemented, camera not present, 3. <code>PictureError.IN_PROGRESS = -2</code>; <code>getPicture()</code> has already been requested but has not yet completed. 4. <code>PictureError.USER_REJECT = -3</code>; the user has canceled the request. 5. <code>PictureError.BAD_OPTIONS = -4</code>; supplied options were not recognized. 6. <code>PictureError.TOO_LARGE = -5</code>; the returned image size was too large to be handled by JavaScript 7. <code>PictureError.UNKNOWN = -6</code>; an unknown error occurred.

*Source**Camera.js, line 268* on page 266.

onGetPictureSuccess(filename, response) method

User provided function that will be invoked when the *hwc.getPicture* on page 192 function is successful.

Syntax

```
<static> onGetPictureSuccess( filename, response )
```

Parameters

Name	Type	Description
<i>filename</i>	string	file name of the image
<i>response</i>	string	<p>the response will be either a Base64-encoded JPG string or a URI depending on the options passed to the <i>hwc.getPicture</i> on page 192 function.</p> <ul style="list-style-type: none"> if options.destinationType == PictureOption.DestinationType.IMAGE_URI, response is an uniform reference identifier for the image. <i>onGetPictureSuccess(fileName, imageURI)</i> if options.destinationType == PictureOption.DestinationType.IMAGE_DATA, response is a Base64-encoded string. <i>onGetPictureSuccess(fileName, imageData)</i>

Source

Camera.js, line 270 on page 266.

sendRequestErrorCB(data) method

Callback function that will be invoked *HttpsConnection.get()/sendRequest()* failed.

Syntax

```
<static> sendRequestErrorCB( data )
```

Parameters

Name	Type	Description
<i>data</i>	<i>anonymous.sendRequestErrorCBParameter</i> on page 70	The error object.

Source

Plugins/HttpsProxy/https-proxy.js, line 502 on page 558.

sendRequestSuccessCB(data) method

Callback function that will be invoked `HttpsConnection.get()/sendRequest()` succeeded.

Syntax

```
<static> sendRequestSuccessCB( data )
```

Parameters

Name	Type	Description
<i>data</i>	<i>anonymous.sendRequestSuccessCBParameter</i> on page 71	The response data object.

Source

Plugins/HttpsProxy/https-proxy.js, line 500 on page 558.

startOrStopLogListenerErrorCallback(data) method

Callback function that will be invoked upon failure to start a log listener via `AppLog.startLogListener` on page 92, or upon failure to removing a log listener via `AppLog.stopLogListener` on page 95.

Syntax

```
<static> startOrStopLogListenerErrorCallback( data )
```

Parameters

Name	Type	Description
<i>data</i>	<i>anonymous.AppLogErrorCallbackParameter</i> on page 70	The error object.

Source

Plugins/AppLog/applog.js, line 477 on page 530.

startOrStopLogListenerSuccessCallback() method

Callback function that will be invoked upon successfully starting a log listener via *AppLog.startLogListener* on page 92, or upon successfully removing a log listener via *AppLog.stopLogListener* on page 95.

Syntax

```
<static> startOrStopLogListenerSuccessCallback()
```

Source

Plugins/AppLog/applog.js, line 475 on page 530.

AppLog namespace

The namespace for AppLog plugin

Members

Name	Description
<i>ERR_UNKNOWN</i> on page 86	Constant indicating the operation failed with unknown error.
<i>STATUS_EVENT_CONNECTED</i> on page 86	Constant indicating an app log entry is associated with the client successfully connecting to the SUP server.
<i>STATUS_EVENT_DISCONNECTED</i> on page 86	Constant indicating an app log entry is associated with the client losing connection to the SUP server.
<i>STATUS_EVENT_DISCONNECTED_LOW_STORAGE</i> on page 87	Constant indicating an app log entry is associated with the client losing connection to the SUP server due to low storage.
<i>STATUS_EVENT_DISCONNECTED_ROAMING</i> on page 87	Constant indicating an app log entry is associated with the client losing connection to the SUP server due to roaming.
<i>STATUS_EVENT_FLIGHT_MODE</i> on page 87	Constant indicating an app log entry is associated with the client going into flight mode.
<i>STATUS_EVENT_NOTIFICATION_RECEIVED</i> on page 88	Constant indicating an app log entry is associated with the client receiving a notification.
<i>STATUS_EVENT_OUT_OF_NETWORK</i> on page 88	Constant indicating an app log entry is associated with the client going out of network.

<i>STATUS_EVENT_REGISTRATION_STARTED</i> on page 88	Constant indicating an app log entry is associated with the client starting registration.
<i>STATUS_EVENT_RESTART</i> on page 89	Constant indicating an app log entry is associated with restarting the client connection to the SUP server.
<i>STATUS_EVENT_SET_DEFAULT_ITEM</i> on page 89	Constant indicating an app log entry is associated with a default app being set from the server.
<i>STATUS_EVENT_SHUTDOWN</i> on page 89	Constant indicating an app log entry is associated with shutting down the client connection to the SUP server.
<i>STATUS_EVENT_STARTUP</i> on page 90	Constant indicating an app log entry is associated with starting the client connection to the SUP server.
<i>STATUS_EVENT_UNKNOWN</i> on page 90	Constant indicating an app log entry is associated with an unknown event.
<i>STATUS_EVENT_UNSET_DEFAULT_ITEM</i> on page 90	Constant indicating an app log entry is associated with a default app being unset from the server.
<i>STATUS_EVENT_WAITING_TO_CONNECT</i> on page 91	Constant indicating an app log entry is associated with the client waiting to connect to the SUP server.

Methods

Name	Description
<i>getLogEntries(successCB, errorCallback)</i> on page 91	Call this function to get an array of <i>AppLog.LogEntry</i> on page 92 objects.
<i>LogEntry(logDate, event, msg)</i> on page 92	This object represents a log entry.
<i>startLogListener(successCB, errorCallback, logListener, [containingObject])</i> on page 92	Registers a log listener.
<i>stopLogListener(successCB, errorCallback, logListener, [containingObject])</i> on page 95	Removes a log listener.

Source

Plugins/AppLog/applog.js, line 16 on page 513.

ERR_UNKNOWN member

Constant indicating the operation failed with unknown error.

Used in *anonymous.AppLogErrorCallbackParameter* on page 70.

Syntax

<static> ERR_UNKNOWN : number

Type

number

Source

Plugins/AppLog/applog.js, line 23 on page 513.

STATUS_EVENT_CONNECTED member

Constant indicating an app log entry is associated with the client successfully connecting to the SUP server.

Used in *AppLog.LogEntry* on page 92.

Syntax

<static> STATUS_EVENT_CONNECTED : number

Type

number

Source

Plugins/AppLog/applog.js, line 53 on page 514.

STATUS_EVENT_DISCONNECTED member

Constant indicating an app log entry is associated with the client losing connection to the SUP server.

Used in *AppLog.LogEntry* on page 92.

Syntax

<static> STATUS_EVENT_DISCONNECTED : number

Type

number

Source

Plugins/AppLog/applog.js, line 59 on page 515.

STATUS_EVENT_DISCONNECTED_LOW_STORAGE member

Constant indicating an app log entry is associated with the client losing connection to the SUP server due to low storage.

Used in *AppLog.LogEntry* on page 92.

Syntax

```
<static> STATUS_EVENT_DISCONNECTED_LOW_STORAGE : number
```

Type

number

Source

Plugins/AppLog/applog.js, line 89 on page 516.

STATUS_EVENT_DISCONNECTED_ROAMING member

Constant indicating an app log entry is associated with the client losing connection to the SUP server due to roaming.

Used in *AppLog.LogEntry* on page 92.

Syntax

```
<static> STATUS_EVENT_DISCONNECTED_ROAMING : number
```

Type

number

Source

Plugins/AppLog/applog.js, line 83 on page 516.

STATUS_EVENT_FLIGHT_MODE member

Constant indicating an app log entry is associated with the client going into flight mode.

Used in *AppLog.LogEntry* on page 92.

Syntax

```
<static> STATUS_EVENT_FLIGHT_MODE : number
```

Type

number

Source

Plugins/AppLog/applog.js, line 65 on page 515.

STATUS_EVENT_NOTIFICATION_RECEIVED member

Constant indicating an app log entry is associated with the client receiving a notification.

Used in *AppLog.LogEntry* on page 92.

Syntax

<static> STATUS_EVENT_NOTIFICATION_RECEIVED : number

Type

number

Source

Plugins/AppLog/applog.js, line 101 on page 516.

STATUS_EVENT_OUT_OF_NETWORK member

Constant indicating an app log entry is associated with the client going out of network.

Used in *AppLog.LogEntry* on page 92.

Syntax

<static> STATUS_EVENT_OUT_OF_NETWORK : number

Type

number

Source

Plugins/AppLog/applog.js, line 71 on page 515.

STATUS_EVENT_REGISTRATION_STARTED member

Constant indicating an app log entry is associated with the client starting registration.

Used in *AppLog.LogEntry* on page 92.

Syntax

<static> STATUS_EVENT_REGISTRATION_STARTED : number

Type

number

Source

Plugins/AppLog/applog.js, line 95 on page 516.

STATUS_EVENT_RESTART member

Constant indicating an app log entry is associated with restarting the client connection to the SUP server.

Used in *AppLog.LogEntry* on page 92.

Syntax

```
<static> STATUS_EVENT_RESTART : number
```

Type

number

Source

Plugins/AppLog/applog.js, line 47 on page 514.

STATUS_EVENT_SET_DEFAULT_ITEM member

Constant indicating an app log entry is associated with a default app being set from the server.

Used in *AppLog.LogEntry* on page 92.

Syntax

```
<static> STATUS_EVENT_SET_DEFAULT_ITEM : number
```

Type

number

Source

Plugins/AppLog/applog.js, line 107 on page 516.

STATUS_EVENT_SHUTDOWN member

Constant indicating an app log entry is associated with shutting down the client connection to the SUP server.

Used in *AppLog.LogEntry* on page 92.

Syntax

```
<static> STATUS_EVENT_SHUTDOWN : number
```

Type

number

Source

Plugins/AppLog/applog.js, line 41 on page 514.

STATUS_EVENT_STARTUP member

Constant indicating an app log entry is associated with starting the client connection to the SUP server.

Used in *AppLog.LogEntry* on page 92.

Syntax

```
<static> STATUS_EVENT_STARTUP : number
```

Type

number

Source

Plugins/AppLog/applog.js, line 35 on page 514.

STATUS_EVENT_UNKNOWN member

Constant indicating an app log entry is associated with an unknown event.

Used in *AppLog.LogEntry* on page 92.

Syntax

```
<static> STATUS_EVENT_UNKNOWN : number
```

Type

number

Source

Plugins/AppLog/applog.js, line 29 on page 514.

STATUS_EVENT_UNSET_DEFAULT_ITEM member

Constant indicating an app log entry is associated with a default app being unset from the server.

Used in *AppLog.LogEntry* on page 92.

Syntax

```
<static> STATUS_EVENT_UNSET_DEFAULT_ITEM : number
```

Type

number

Source

Plugins/AppLog/applog.js, line 113 on page 517.

STATUS_EVENT_WAITING_TO_CONNECT member

Constant indicating an app log entry is associated with the client waiting to connect to the SUP server.

Used in *AppLog.LogEntry* on page 92.

Syntax

```
<static> STATUS_EVENT_WAITING_TO_CONNECT : number
```

Type

number

Source

Plugins/AppLog/applog.js, line 77 on page 515.

getLogEntries(successCB, errorCB) method

Call this function to get an array of *AppLog.LogEntry* on page 92 objects.

There will be one *AppLog.LogEntry* on page 92 object for each line in the app log.

Syntax

```
<static> getLogEntries( successCB, errorCB )
```

Parameters

Name	Type	Description
<i>successCB</i>	<i>anonymous.getLogEntriesSuccessCallback</i> on page 78	The callback function that will receive the asynchronous callback with the log entries.
<i>errorCB</i>	<i>anonymous.getLogEntriesErrorCallback</i> on page 78	The callback function that will be invoked on errors.

Example

```
// A global function called with the log entries.
function onLogEntriesSuccessCallback(data) {
  for ( var i = 0; i < data.length; i++ )
  {
    var logEntry = data[ i ];
    alert('Log entry ' + ( i + 1 ) + ':\n
' +
      'Date (ms): ' + logEntry.date + '\n
' +
      'Status code: ' + logEntry.statusCode + '\n
' +
      'Message: ' + logEntry.message
```

```
    );  
  }  
}  
  
// A global function called if there is an error retrieving log  
entries.  
function onLogEntriesFailureCallback(error) {  
  alert('Error retrieving log entries: ' + error);  
}  
  
// Get the log entries  
AppLog.getLogEntries(onLogEntriesSuccessCallback,  
onLogEntriesFailureCallback);
```

Source

Plugins/AppLog/applog.js, line 163 on page 518.

LogEntry(logDate, event, msg) method

This object represents a log entry.

Syntax

<static> LogEntry(*logDate*, *event*, *msg*)

Parameters

Name	Type	Description
<i>logDate</i>	number	The date the log entry was recorded, in milliseconds since January 1, 1970, 00:00:00 GMT.
<i>event</i>	number	The event ID of the log entry (will be one of the AppLog status events, or possibly a custom value).
<i>msg</i>	string	The message of the log entry.

Source

Plugins/AppLog/applog.js, line 125 on page 517.

startLogListener(successCB, errorCallback, logListener, [containingObject]) method

Registers a log listener.

Syntax

```
<static> startLogListener( successCB, errorCB, logListener, [containingObject] )
```

Parameters

Name	Type	Argument	Description
<i>successCB</i>	<i>anonymous.startOr-StopLogListenerSuccessCallback</i> on page 84		A callback function that will be invoked if the log listener is successfully registered.
<i>errorCB</i>	<i>anonymous.startOr-StopLogListenerError-Callback</i> on page 83		A callback function that will be invoked if there is an error registering the log listener.
<i>logListener</i>	<i>anonymous.logListener</i> on page 79		The callback to register. This will be invoked when new entries are added to the log.
<i>containingObject</i>	Object	(optional)	Object containing the definition for logListener. If a log listener callback function references variables in its containing object, then the containing object should be passed to this function.

Example

```
// This example shows how to use this function with a globally-scoped
logListener.
// A global function called by the log listener.
var doSomething = function()
{
    alert("this gets displayed when there is a new log entry.");
}

// The log listener callback function that will be passed to
AppLog.startLogListener.
// This function will be invoked whenever there is a new log entry.
var logListener = function( date, statusCode, message )
{
    doSomething();
}
```

```
function onStartLogListenerSuccessCallback() {
    // Do something here after listener has been added
}

function onStartLogListenerFailureCallback(error) {
    // React to error here
}

// Add the log listener.
AppLog.startLogListener( onStartLogListenerSuccessCallback,
                        onStartLogListenerFailureCallback,
                        logListener );

// This example shows how to use this function with a logListener
contained in an object.
// logListenerManager is an object that will contain the listener
callback as well
// as a function that will be invoked from the listener callback
function.
var logListenerManager = {};

// This is a function that is called from the listener callback.
logListenerManager.doSomething = function()
{
    alert("this gets displayed when there is a new log entry.");
}

// This is the listener callback that will be passed to
AppLog.startLogListener.
// Since a variable is referenced from the containing object, the
containing object
// will need to be passed to AppLog.startLogListener.
logListenerManager.listener = function( date, statusCode, message )
{
    this.doSomething();
}

function onStartLogListenerSuccessCallback() {
    // Do something here after listener has been added
}

function onStartLogListenerFailureCallback(error) {
    // React to error here
}

// Pass both the listener callback and the containing object.
AppLog.startLogListener( onStartLogListenerSuccessCallback,
                        onStartLogListenerFailureCallback,
                        logListenerManager.listener,
                        logListenerManager );
```

Source

Plugins/AppLog/applog.js, line 249 on page 522.

stopLogListener(successCB, errorCallback, logListener, [containingObject])
method

Removes a log listener.

This function should be called with identical parameters that were used when adding the log listener with *AppLog.startLogListener* on page 92.

Syntax

```
<static> stopLogListener( successCB, errorCallback, logListener, [containingObject] )
```

Parameters

Name	Type	Argument	Description
<i>successCB</i>	<i>anonymous.startOr-StopLogListenerSuccessCallback</i> on page 84		A callback function that will be invoked if the log listener is successfully removed.
<i>errorCB</i>	<i>anonymous.startOr-StopLogListenerError-Callback</i> on page 83		A callback function that will be invoked if there is an error removing the log listener.
<i>logListener</i>	<i>anonymous.logListener</i> on page 79		The callback that was added with <i>AppLog.startLogListener</i> on page 92.
<i>containingObject</i>	Object	(optional)	Object containing the definition for logListener.

Example

```
// This example shows how to use this function with a globally-scoped
logListener.
// A global function called by the log listener.
var doSomething = function()
{
    alert("this gets displayed when there is a new log entry.");
}

// The log listener callback function that will be passed to
AppLog.startLogListener.
// This function will be invoked whenever there is a new log entry.
var logListener = function( date, statusCode, message )
{
    doSomething();
}
```

```
}

function onStartLogListenerSuccessCallback() {
    // Do something here after listener has been added
}

function onStartLogListenerFailureCallback(error) {
    // React to error here
}

function onStopLogListenerSuccessCallback() {
    // Do something here after listener has been removed
}

function onStopLogListenerFailureCallback(error) {
    // React to error here
}

// Add the log listener.
AppLog.startLogListener( onStartLogListenerSuccessCallback,
                        onStartLogListenerFailureCallback,
                        logListener );

// At some other point if we want to remove the listener, we use the
following line.
AppLog.stopLogListener( onStopLogListenerSuccessCallback,
                       onStopLogListenerFailureCallback,
                       logListener );

// This example shows how to use this function with a logListener
contained in an object.
// logListenerManager is an object that will contain the listener
callback as well
// as a function that will be invoked from the listener callback
function.
var logListenerManager = {};

// This is a function that is called from the listener callback.
logListenerManager.doSomething = function()
{
    alert("this gets displayed when there is a new log entry.");
}

// This is the listener callback that will be passed to
AppLog.startLogListener.
// Since a variable is referenced from the containing object, the
containing object
// will need to be passed to AppLog.startLogListener.
logListenerManager.listener = function( date, statusCode, message )
{
    this.doSomething();
}

function onStartLogListenerSuccessCallback() {
    // Do something here after listener has been added
}
```

```
function onStartLogListenerFailureCallback(error) {
    // React to error here
}

function onStopLogListenerSuccessCallback() {
    // Do something here after listener has been removed
}

function onStopLogListenerFailureCallback(error) {
    // React to error here
}

// Pass both the listener callback and the containing object.
AppLog.startLogListener( onStartLogListenerSuccessCallback,
                        onStartLogListenerFailureCallback,
                        logListenerManager.listener,
                        logListenerManager );

// At some other point if we want to remove the listener, we use the
following line.
AppLog.stopLogListener( onStopLogListenerSuccessCallback,
                       onStopLogListenerFailureCallback,
                       logListenerManager.listener,
                       logListenerManager );
```

Source

Plugins/AppLog/applog.js, line 376 on page 526.

HttpsConnection namespace

The namespace for HTTP(S) proxy

Classes

Name	Description
<i>HttpsConnection.CertificateFromAfaria</i> on page 98	Create certificate source description object for certificates from Afaria.
<i>HttpsConnection.CertificateFromFile</i> on page 99	Create certificate source description object for certificates from a keystore file.
<i>HttpsConnection.CertificateFromStore</i> on page 99	Create certificate source description object for certificates from system keystore (Keystore in BB, Keychain in iOS and Android).

Methods

Name	Description
------	-------------

<i>deleteCertificateFromStore(successCB, [errorCB], certificateKey)</i> on page 100	Delete cached certificate from keychain.
<i>generateODataHttpClient()</i> on page 101	Generate an OData HttpClient object over https proxy of native platform.
<i>get(url, header, successCB, [errorCB], [user], [password], [timeout], [certSource])</i> on page 102	Send a HTTP(S) GET request to a remote server.
<i>sendRequest(method, url, header, requestBody, successCB, errorCB, [user], [password], [timeout], [certSource])</i> on page 104	Send a HTTP(S) request to a remote server.

Source

Plugins/HttpsProxy/dataajs-https-proxy.js, line 18 on page 533.

HttpsConnection.CertificateFromAfaria class

Create certificate source description object for certificates from Afaria.

Syntax

`new CertificateFromAfaria(CN, [ChallengeCode])`

Parameters

Name	Type	Argument	Description
<i>CN</i>	string		Common Name (CN) for CA/SCEP protocol. For iOS, the retrieved certificate is stored in the key store with the common name as the certificate key, the following requests for the same common name will just load the saved certificate from key store, instead of sending a new request to Afaria server.
<i>ChallengeCode</i>	string	(optional)	Challenge code for CA/SCEP protocol.

Source

Plugins/HttpsProxy/https-proxy.js, line 137 on page 543.

HttpsConnection.CertificateFromFile class

Create certificate source description object for certificates from a keystore file.

Not supported on Blackberry platform

Syntax

```
new CertificateFromFile( Path, Password, CertificateKey )
```

Parameters

Name	Type	Description
<i>Path</i>	string	Path of the keystore file. For iOS client, it first tries to load the relative file path from application's Documents folder; if it fails, then tries to load the file path from application's main bundle. In addition, before trying to load the certificate from file system, iOS client first checks whether the specified certificate key already exists in the key store, if so, it just loads the existing certificate from key store, instead of loading the certificate from file system.
<i>Password</i>	string	Password of the keystore.
<i>CertificateKey</i>	string	An unique key that will be used to locate the certificate.

Source

Plugins/HttpsProxy/https-proxy.js, line 119 on page 543.

HttpsConnection.CertificateFromStore class

Create certificate source description object for certificates from system keystore (Keystore in BB, Keychain in iOS and Android).

The certificateKey is not used on the BB platform. BB will prompt the user to select a certificate if a certificate was not already used for the server connection.

Syntax

```
new CertificateFromStore( CertificateKey )
```

Parameters

Name	Type	Description
<i>CertificateKey</i>	string	An unique key that will be used to locate the certificate. Not used in BB platform.

Source

Plugins/HttpsProxy/https-proxy.js, line 152 on page 544.

deleteCertificateFromStore(successCB, [errorCB], certificateKey) method

Delete cached certificate from keychain.

iOS client will always try the cached certificate first if it is available before requesting the certificate from afaria server or loading the certificate from file system. In case the cached certificate is no longer valid, use this method to delete it from keychain **Only supported by iOS platform**

Syntax

```
<static> deleteCertificateFromStore( successCB, [errorCB], certificateKey )
```

Parameters

Name	Type	Argument	Description
<i>successCB</i>	<i>anonymous.sendRequestSuccessCB</i> on page 83		Callback method upon success.
<i>errorCB</i>	<i>anonymous.sendRequestErrorCB</i> on page 82	(optional)	Callback method upon failure.
<i>certificateKey</i>	string		The key of the certificate to be deleted.

Source

Plugins/HttpsProxy/https-proxy.js, line 409 on page 555.

generateODataHttpClient() method

Generate an OData HttpClient object over https proxy of native platform.

This object will re-direct all odata request to the http proxy because even with HTTP connection, there are some known issue by default setting since the application in device is cross server accessing the odata service. See: <http://datajs.codeplex.com/discussions/396112> for details of the issue. Call this method normally on HTML page load event to replace the default odata HTTP client.

Syntax

```
<static> generateODataHttpClient()
```

Example

```
// Call datajs api without certificate, users could call just as
normal by passing
// URL as first argument
var length = 0;
var updateUri = server + "/example.svc/Categories(1)";

OData.read(server + "/example.svc/Categories",
function (data, response) {
    alert("length " + data.results.length);
    length = data.results.length;
    if ( length > 0 )
{
    var updateRequest = {
        requestUri: updateUri,
        method: "PUT",
        data:
{
        Picture: new Date().getTime(),
        Description: "Update Record",
        CategoryName: "Updated Category",
        CategoryID: 1
        }
    };

    OData.request(updateRequest,
        function (data, response) {
            alert("Response " + JSON.stringify(response));
        },
        function (err) {
            alert("Error occurred " + err.message);
        }
    );
};
},
function (err) {
    alert("Error occurred " + err.message);
});
```

```
// However, to specify certificate source in the method call, users
need to pass in
// the request object instead of URL,
// and add the field "certificateSource" to the request object.
var length = 0;
var updateUri = server + "/example.svc/Categories(1)";

OData.read({ requestUri: server + "/example.svc/Categories",
certificateSource : cert},
function (data, response) {
    alert("length " + data.results.length);
    length = data.results.length;
    if ( length > 0 )
    {
        var updateRequest = {
            requestUri: updateUri,
            certificateSource : cert,
            method: "PUT",
            data:
            {
                Picture: new Date().getTime(),
                Description: "Update Record",
                CategoryName: "Updated Category",
                CategoryID: 1
            }
        };

        OData.request(updateRequest,
            function (data, response) {
                alert("Response " + JSON.stringify(response));
            },
            function (err) {
                alert("Error occurred " + err.message);
            }
        );
    }
},
function (err) {
    alert("Error occurred " + err.message);
});
```

Source

Plugins/HttpsProxy/dataajs-https-proxy.js, line 109 on page 536.

get(url, header, successCB, [errorCB], [user], [password], [timeout], [certSource]) method

Send a HTTP(S) GET request to a remote server.

Syntax

```
<static> get( url, header, successCB, [errorCB], [user], [password], [timeout], [certSource] )
{anonymous.abort}
```


Parameters

Name	Type	Argument	Description
<i>url</i>	string		The http url with format http(s)://[user:password]@host-name[:port]/path.
<i>header</i>	Object		HTTP header to be sent to server. This is an Object. Can be null.
<i>successCB</i>	<i>anonymous.sendRequestSuccessCB</i> on page 83		Callback method upon success.
<i>errorCB</i>	<i>anonymous.sendRequestErrorCB</i> on page 82	(optional)	Callback method upon failure.
<i>user</i>	string	(optional)	User ID for basic authentication.
<i>password</i>	string	(optional)	User password for basic authentication.
<i>timeout</i>	number	(optional)	Timeout setting in seconds.
<i>certSource</i>	Object	(optional)	Certificate description object. It can be one of <i>HttpsConnection.CertificateFromFile</i> on page 99, <i>HttpsConnection.CertificateFromStore</i> on page 99, or <i>HttpsConnection.CertificateFromAfaria</i> on page 98.

Returns

A JavaScript function object to cancel the operation.

Type:

anonymous.abort on page 72

Example

```
// To send a get request to server, call the method
HttpsConnection.get("http://www.google.com", null, function (data) {
    alert("Status: " + JSON.stringify(data.status));
    alert("Headers: " + JSON.stringify(data.headers));
    if (data.responseText){
        alert("Response: " +
JSON.stringify(data.responseText));
    }
},
function (error) {
    alert("Failed: " + JSON.stringify(error));
});
// To send a get request to server with headers, call the method
HttpsConnection.get(url, {HeaderName : "Header value"}, successCB,
errorCB);
// To send a get request to server with basic authentication, call
the method
HttpsConnection.get(url, headers, successCB, errorCB, "username",
"password");
// To send a get request to server with mutual authentication, call
the method
HttpsConnection.get("https://hostname", headers, successCB, errorCB,
null, null, 0,
new CertificateFromFile("/mnt/sdcard/my.p12", "password",
"mykey"));
```

Source

Plugins/HttpsProxy/https-proxy.js, line 395 on page 554.

sendRequest(method, url, header, requestBody, successCB, errorCallback, [user], [password], [timeout], [certSource]) method

Send a HTTP(S) request to a remote server.

Syntax

<static> sendRequest(*method*, *url*, *header*, *requestBody*, *successCB*, *errorCB*, [*user*], [*password*], [*timeout*], [*certSource*]) {anonymous.abort}

Parameters

Name	Type	Argument	Description
<i>method</i>	string		Standard HTTP request method name.

<i>url</i>	string		The http url with format http(s)://[user:password]@host-name[:port]/path.
<i>header</i>	Object		HTTP header to be sent to server.This is an Object. Can be null.
<i>requestBody</i>	string		Data to be sent to server with the request.It's a string value. Can be null.
<i>successCB</i>	<i>anonymous.sendRequestSuccessCB</i> on page 83		Callback method upon success.
<i>errorCB</i>	<i>anonymous.sendRequestErrorCB</i> on page 82		Callback method upon failure.
<i>user</i>	string	(optional)	User ID for basic authentication.
<i>password</i>	string	(optional)	User password for basic authentication.
<i>timeout</i>	number	(optional)	Timeout setting in seconds.
<i>certSource</i>	Object	(optional)	Certificate description object.It can be one of <i>HttpsConnection.CertificateFromFile</i> on page 99, <i>HttpsConnection.CertificateFromStore</i> on page 99, or <i>HttpsConnection.CertificateFromAfaria</i> on page 98.

Returns

A JavaScript function object to cancel the operation.

Type:

anonymous.abort on page 72

Example

```
// To send a post request to server, call the method
HttpsConnection.sendRequest("POST", "http://www.google.com", null,
"THIS IS THE BODY", function (data) {
    alert("Status: " + JSON.stringify(data.status));
    alert("Headers: " + JSON.stringify(data.headers));
    alert("Response: " + JSON.stringify(data.response));
}, function (data) {
    alert("Failed: " + JSON.stringify(data));});
// To send a post request to server with headers, call the method
HttpsConnection.sendRequest("POST", url, {HeaderName : "Header
value"}, "THIS IS THE BODY", successCB, errorCB);
// To send a post request to server with basic authentication, call
the method
HttpsConnection.sendRequest("POST", url, headers, "THIS IS THE
BODY", successCB, errorCB, "username", "password");
// To send a post request to server with mutual authentication, call
the method
HttpsConnection.sendRequest("POST", "https://hostname", headers,
"THIS IS THE BODY", successCB, errorCB, null,
    null, 0, new CertificateFromFile("/mnt/sdcard/my.keystore",
"password", "mykey"));
```

Source

Plugins/HttpsProxy/https-proxy.js, line 289 on page 550.

hwc namespace

The namespace for the Hybrid Web Container javascript

Classes

Name	Description
<i>hwc.SUPStorage</i> on page 116	Storage
<i>hwc.SUPStorageException</i> on page 120	Storage

Namespaces

Name	Description
<i>NativeErrorCodes</i> on page 121	This object contains constants representing the different types of public native error codes.

Members

Name	Description
------	-------------

<i>APP_ADDED</i> on page 126	A constant indicating that a hybrid app has been added.
<i>APP_REFRESH</i> on page 126	A constant indicating that the application list requires a refresh.
<i>APP_REMOVED</i> on page 127	A constant indicating that a hybrid app was removed.
<i>APP_UPDATED</i> on page 127	A constant indicating that a hybrid app was updated.
<i>CONNECTED</i> on page 127	Constant indicating that the hwc is connected.
<i>CONNECTION_CONNECTED</i> on page 128	A constant indicating that the log message is about the connection being established.
<i>CONNECTION_DISCONNECTED</i> on page 128	A constant indicating that the log message is about the connection being disconnected.
<i>CONNECTION_ERROR</i> on page 128	A constant indicating that the log message is about a connection error.
<i>CONNECTION_OTHER</i> on page 129	A constant indicating that the log message is not about the connection.
<i>CONNECTION_RETRIEVED_ITEMS</i> on page 129	a constant indicating that the log message is about retrieved items.
<i>DEFAULT_CUSTOM_ICON_INDEX</i> on page 129	A constant indicating the custom icon index.
<i>DISCONNECTED</i> on page 130	Constant indicating that the hwc is disconnected.
<i>e2eTrace</i> on page 130	Represents an E2E Trace.
<i>INSTALLATION_BEGIN</i> on page 132	A constant indicating that the application is starting to be installed.
<i>INSTALLATION_END</i> on page 132	A constant indicating that the application has finished being installed.
<i>MediaCache</i> on page 133	Represents a Media Cache.
<i>MSG_ADDED</i> on page 135	A constant indicating that a message has been added.
<i>MSG_PRIORITY_HIGH</i> on page 135	A constant indicating a message has high priority.
<i>MSG_PRIORITY_NORMAL</i> on page 136	A constant indicating a message has normal priority.

<i>MSG_REFRESH</i> on page 136	A constant indicating that a message needs to be refreshed.
<i>MSG_REMOVED</i> on page 136	A constant indicating that a message has been removed.
<i>MSG_UPDATED</i> on page 137	A constant indicating that a message has been updated.
<i>NOTIFICATION_CANCEL</i> on page 137	A constant indicating that no more push notification listeners should be called.
<i>NOTIFICATION_CONTINUE</i> on page 137	A constant indicating that other push notification listeners should continue to be called.
<i>OPEN_APP_NOT_EXIST</i> on page 138	A constant indicating that <i>hwc.openApp</i> on page 221 failed because the specified app does not exist.
<i>OPEN_APP_OTHER</i> on page 138	A constant indicating that <i>hwc.openApp</i> on page 221 failed for an unspecified reason.
<i>OPEN_APP_SUCCESS</i> on page 138	A constant indicating that <i>hwc.openApp</i> on page 221 completed successfully.
<i>OPEN_MSG_APP_NOT_EXIST</i> on page 139	A constant indicating that a message could not be opened because there was no associated hybrid app.
<i>OPEN_MSG_NOT_EXIST</i> on page 139	A constant indicating that a message could not be opened because no message with the given ID exists.
<i>OPEN_MSG_OTHER</i> on page 139	A constant indicating that a message could not be opened due to an unspecified error.
<i>OPEN_MSG_SUCCESS</i> on page 140	A constant indicating that a message was successfully opened.
<i>perf</i> on page 140	Represents the Performance Manager.
<i>PictureError</i> on page 142	An array that holds all possible error codes
<i>REG_ERR_AUTO_REG_NOT_ENABLED</i> on page 144	Constant indicating that auto registration was not enabled in the template.
<i>REG_ERR_AUTO_REG_TEMPLATE_NOT_FOUND</i> on page 145	Constant indicating that no MBS template was found for given AppId and/or Security configuration.

<i>REG_ERR_AU-TO_REG_USER_NAME_TOO_LONG</i> on page 145	Constant indicating that the user name is longer than the legal limit.
<i>REG_ERR_AU-TO_REG_WRONG_USER_FOR_DEVICE</i> on page 145	Constant indicating that the given device id is already registered for another user.
<i>REG_ERR_COULD_NOT_REACH_MMS_SERVER</i> on page 146	Constant indicating that the connection to the MMS service failed.
<i>REG_ERR_INVALID_USER_NAME</i> on page 146	Constant indicating that the user name contains invalid characters.
<i>REG_ERR_MMS_AUTHENTICATION_FAILED</i> on page 146	Constant indicating that MMS Authentication failed.
<i>REGISTRATION_METHOD_AFARIA</i> on page 147	Constant indicating that automatic registration using a certificate from Afaria is the preferred method.
<i>REGISTRATION_METHOD_AUTOMATIC</i> on page 147	Constant indicating that automatic registration using password is the preferred method.
<i>REGISTRATION_METHOD_CERTIFICATE</i> on page 147	Constant indicating that automatic registration using a local certificate is the preferred method.
<i>REGISTRATION_METHOD_MANUAL</i> on page 148	Constant indicating that manual registration is the preferred method.
<i>REGISTRATION_METHOD_NO_PREFERENCE</i> on page 148	Constant indicating no registration method preference.
<i>SETTING_SUCCESS</i> on page 148	Constant indicating <i>hwc.saveSettings</i> on page 237 completed successfully.
<i>STATUS</i> on page 149	This object contains constants representing the status of the hybrid app.

Methods

Name	Description
<i>activationRequired()</i> on page 149	This function sets the activation required state of this hybrid app to true.
<i>addAppInstallationListener(AppInstallationListener)</i> on page 149	Register the application installation listener.

<i>addAppListener(ApplicationListener, [containingObject])</i> on page 150	Register the application listener.
<i>addConnectionListener(ConnectionStateListener, [containingObject])</i> on page 151	Register the connection state listener.
<i>addLogListener(LogListener, [containingObject])</i> on page 153	Register the log listener.
<i>addMenuItemCollection(collection)</i> on page 154	This function adds a menu item collection to the menu items for the screen.
<i>addMessageListener(filters, MessageListener, [containingObject])</i> on page 155	Registers a message listener.
<i>addPushNotificationListener(PushNotificationListener, [containingObject])</i> on page 157	Register a push notification listener.
<i>CertificateStore()</i> on page 158	Use these functions for X.509 credential handling.
<i>clearCache()</i> on page 164	This function clears the contents of the on-device request result cache for the current hybrid app.
<i>clearCacheItem(cachekey)</i> on page 164	This function clears an item from the contents of the on-device request result cache for the current hybrid app.
<i>ClientVariables(clientVariablesVersion, clientVariableItems)</i> on page 165	Represents a ClientVariables object.
<i>ClientVariablesException(errCode, errMsg)</i> on page 168	This exception is thrown when hwc.ClientVariables#getVariableValueByName is called with a variable name that does not exist.
<i>close()</i> on page 168	This function closes the hybrid app.
<i>ConnectionSettings(regmethod, server, port, server, user, activationcode, protocol, password, urlsuffix)</i> on page 168	Represents the connection settings for connecting to the SUP Server.
<i>connectToServer([onNotification])</i> on page 170	Resumes the connection to the SUP server.
<i>convertLocalTimeToUtc(date)</i> on page 171	Timezone
<i>convertUtcToLocalTime(date)</i> on page 171	Timezone

<i>CustomIcon(width, height, type, name, path, processedPath, moduleId, moduleVersion, index)</i> on page 172	Represents a CustomIcon.
<i>disconnectFromServer()</i> on page 175	Suspends the connection to the SUP server.
<i>expireCredentials()</i> on page 175	Allows the user to set the credentials to the expired state for the current hybrid app.
<i>getAllMessages([messageFilter], [completeList])</i> on page 175	Gets received messages based on a filter and the existence of a default hybrid app.
<i>getAppById(moduleId, version)</i> on page 177	Gets a <i>hwc.HybridApp</i> on page 197 object with the given module id and version.
<i>getAppIconUrl(app, processed)</i> on page 177	This function gets the URL of the icon for a hybrid app depending on whether custom icons are defined.
<i>getApplicationConnectionID()</i> on page 178	Gets the Hybrid Web Container application connection ID.
<i>getBuiltInIconUrl(iconIndex, processed)</i> on page 179	Gets the icon URL for the built-in icon.
<i>getCallbackFromNativeError(errString)</i> on page 180	Extract the error call back method name from a URL string.
<i>getClientVariables(moduleId, version)</i> on page 180	Gets the client variables of the hybrid app with given module id and version.
<i>getCodeFromNativeError(errString)</i> on page 181	Extract an error code from a URL string.
<i>getCurrentApp()</i> on page 182	Gets the hybrid app that is currently open.
<i>getCurrentLocale()</i> on page 182	Timezone
<i>getCustomIconUrl(moduleId, moduleVersion, iconIndex, processed)</i> on page 183	Gets the URL to the custom icon.
<i>getDstOffsetAtGivenTimeInMinutes(date)</i> on page 183	Timezone
<i>getExternalResource(url, options)</i> on page 184	Makes an external cross domain request.
<i>getInstalledApps([completeList])</i> on page 185	Returns an array of <i>hwc.HybridApp</i> on page 197 objects.
<i>getLocalizedDate(date)</i> on page 186	Timezone

<i>getLocalizedDateTime(date)</i> on page 186	Timezone
<i>getLocalizedTime(date)</i> on page 187	Timezone
<i>getLogEntries()</i> on page 187	Call this function to get an array of <i>hwc.LogEntry</i> on page 209 objects.
<i>getLoggingAlertDialog()</i> on page 188	This function gets the callback used by <i>hwc.log</i> when it is required to notify the user of a log item.
<i>getLoggingCurrentLevel()</i> on page 188	This function gets the logging level.
<i>getMessageByID(msgId)</i> on page 189	Gets a <i>hwc.Message</i> on page 215 object with the given message ID.
<i>getMsgIconUrl(msg)</i> on page 189	This function gets the URL of the icon for a message object depending on its processed status and whether there are custom icons defined.
<i>getNativeMessageFromNativeError(errString)</i> on page 190	Extract a native message from a URL string.
<i>getOffsetFromUTC(date)</i> on page 191	Timezone
<i>getOnErrorMessageFromNativeError(errString)</i> on page 191	Extract the error message from a URL string.
<i>getPicture(onGetPictureError, onGetPictureSuccess, options)</i> on page 192	Camera
<i>getQueryVariable(variable)</i> on page 193	This function looks in the query string on the URL for the value corresponding to the given name.
<i>getServerInitiatedApps()</i> on page 193	Returns an array of <i>hwc.HybridApp</i> on page 197 objects that are server initiated.
<i>getSharedStorageKey()</i> on page 194	Storage
<i>getTimezoneId()</i> on page 194	Timezone
<i>getTransformData()</i> on page 195	Returns the transform data for the hybridapp.
<i>getURLParamFromNativeError(paramName, url)</i> on page 195	Extract a parameter value from a URL string with a given parameter name.
<i>getUsesDST()</i> on page 195	Timezone
<i>getXMLHttpRequest()</i> on page 196	Reliably returns an XMLHttpRequest object regardless of what platform this code is being executed on.

<i>guid()</i> on page 196	This function generates a GUID (globally unique identifier).
<i>hideProgressDialog()</i> on page 197	This function hides the progress dialog displaying the spinner.
<i>HybridApp(moduleId, version, displayName, iconIndex, defaultCustomIcon, customIconList)</i> on page 197	This object represents a hybrid app.
<i>isAndroid()</i> on page 201	Platform
<i>isAndroid3()</i> on page 201	Platform
<i>isBlackBerry()</i> on page 201	Platform
<i>isBlackBerry5()</i> on page 202	Platform
<i>isBlackBerry5WithTouchScreen()</i> on page 202	Platform
<i>isBlackBerry6NonTouchScreen()</i> on page 202	Platform
<i>isBlackBerry7()</i> on page 203	Platform
<i>isClosed()</i> on page 203	This function checks if the hybrid app has been closed.
<i>isDstActiveAtGivenTime(date)</i> on page 204	Timezone
<i>isIOS()</i> on page 204	Platform
<i>isIOS4()</i> on page 205	Returns true if the hybrid app application is being run on iOS4
<i>isIOS5()</i> on page 205	Platform
<i>isIOS6()</i> on page 205	Returns true if the hybrid app application is being run on iOS6
<i>isIOS7()</i> on page 206	Returns true if the hybrid app application is being run on iOS7
<i>isIPad()</i> on page 206	Platform
<i>isSharedStorageEnabled()</i> on page 206	Storage
<i>isWindows()</i> on page 207	Platform
<i>isWindowsMobile()</i> on page 207	Platform
<i>loadSettings()</i> on page 207	Loads the current connection settings from the native application storage.

<i>log(sMsg, eLevel, notifyUser)</i> on page 208	Allows the user to log a message to the device trace log which can be remotely retrieved from the server.
<i>LogEntry(date, event, msg)</i> on page 209	This object represents a log entry.
<i>markAsActivated()</i> on page 211	This function sets the activation required state for the current hybrid app to false.
<i>markAsProcessed()</i> on page 211	Allows the user to set the processed state to true for the current message.
<i>MenuItemCollection()</i> on page 212	This class represents a collection of menu items.
<i>Message(msgId, date, icon, sender, isRead, processed, priority, subject, module, version)</i> on page 215	Represents a message received by the HWC.
<i>MessageFilter([sender], [subject], [moduleId], [version], [isread], [processed])</i> on page 221	Represents a filter used to filter messages.
<i>openApp(moduleId, version)</i> on page 221	Launch the hybrid app with the given module ID and version.
<i>openMessage(msgId)</i> on page 222	Launch the server initiated hybrid app associated with a message.
<i>removeAllMenuItems()</i> on page 223	This function removes all menu items that were added by the hybrid app.
<i>removeAppInstallationListener(AppInstallationListener)</i> on page 223	Remove the application installation listener.
<i>removeAppListener(ApplicationListener, [containingObject])</i> on page 224	Remove the application listener.
<i>removeConnectionListener(ConnectionStateListener, [containingObject])</i> on page 226	Remove the connection state listener.
<i>removeLogListener(LogListener, [containingObject])</i> on page 227	Remove the log listener.
<i>removeMessage(msgId)</i> on page 229	Removes (deletes) a message.
<i>removeMessageListener(MessageListener, [containingObject])</i> on page 229	Removes the message listener.
<i>removePushNotificationListener(PushNotificationListener, [containingObject])</i> on page 231	Remove the push notification listener.

<i>sample_AppListener(event, moduleId, version)</i> on page 232	A sample <i>anonymous.ApplicationListener</i> on page 75 callback function.
<i>sample_ConnectionListener(event, errorCode, errorMessage)</i> on page 233	A sample <i>anonymous.ConnectionStateListener</i> on page 77 callback function.
<i>sample_InstallationAppListener(event, moduleId, version, moduleName, designerVersion, containerVersion)</i> on page 234	Sample application listener callback function
<i>sample_LogListener(milliseconds, event, optionalString)</i> on page 234	Sample <i>anonymous.LogListener</i> on page 79 callback function.
<i>sample_MessageListener(flag, msgId)</i> on page 235	A sample <i>anonymous.MessageListener</i> on page 80 callback function.
<i>sample_PushNotificationListener(notifications)</i> on page 235	A sample implementation of a <i>anonymous.PushNotificationListener</i> callback function.
<i>saveLoginCertificate(certificate)</i> on page 236	This function saves login credentials from a certificate to the credential cache.
<i>saveLoginCredentials(userName, password)</i> on page 236	This function saves login credentials to the credential cache.
<i>saveSettings(settings)</i> on page 237	Save the connection settings to native application storage.
<i>setLoggingAlertDialog(newAlertDialogCallback)</i> on page 238	This function sets the callback used by <i>hwc.log</i> when it is required to notify the user of a log item.
<i>setLoggingCurrentLevel(newLoggingLevel)</i> on page 239	This function sets the logging level.
<i>setReportErrorFromNativeCallback(callbackToSet)</i> on page 239	This function sets the callback function called when there is a native error reported.
<i>setScreenTitle_CONT(screenTitle)</i> on page 240	Sets the title of the screen.
<i>SharedStorage()</i> on page 240	Storage
<i>showAlertDialog(message, [title])</i> on page 240	Displays an alert dialog to the user.
<i>showAttachmentContents_CONT(contents, mimeType, fileName, waitDialogCallbackString)</i> on page 241	Shows the given file contents in a content-appropriate way.

<i>showAttachmentFromCache_CONT(uniqueKey, mimeType, fileName, waitDialogCallbackString)</i> on page 242	Shows the given file contents in a content-appropriate way.
<i>showCertificatePicker()</i> on page 243	This function opens a form on the device that allows the user to specify the credentials for the use of certificate-based authentication.
<i>showConfirmDialog(message, [title])</i> on page 243	Shows a confirm dialog to the user.
<i>showLocalAttachment(key)</i> on page 244	Shows a local attachment.
<i>showProgressDialog([message])</i> on page 244	This function shows a progress dialog with spinner.
<i>showUrlInBrowser(url)</i> on page 245	This function opens the supplied URL in a browser.
<i>shutdown()</i> on page 246	Shutdown the client connection to the SUP server.
<i>startClient([onNotification])</i> on page 246	Start the client connection to the SUP server.
<i>this.getIconUrl(processed)</i> on page 247	Gets the URL of this custom icon.
<i>updateMessageProcessed(msgId, status)</i> on page 248	Updates the message processed status.
<i>updateMessageRead(msgId, status)</i> on page 248	Updates the message read status.

Source

Callbacks.js, line 15 on page 254.

hwc.SUPStorage class

Storage

Syntax

`new SUPStorage(store)`

Parameters

Name	Type	Description
<i>store</i>	string	the store name

Example

```
var store1 = new hwc.SUPStorage("one");
```

Members

Name	Description
<i>BB7_MAX_STRING_STORAGE_LENGTH</i> on page 117	A constant for the maximum length for a string being stored on BB7 BB7 cannot handle strings with length longer than 524000 This restriction applies to real devices as well as simulators.

Methods

Name	Description
<i>clear()</i> on page 117	Storage
<i>getItem(key)</i> on page 118	Storage
<i>key(index)</i> on page 118	Storage
<i>length()</i> on page 119	Storage
<i>removeItem(key)</i> on page 119	Storage
<i>setItem(key, value)</i> on page 120	Storage

Source

SUPStorage.js, line 40 on page 561.

BB7_MAX_STRING_STORAGE_LENGTH member

A constant for the maximum length for a string being stored on BB7 BB7 cannot handle strings with length longer than 524000 This restriction applies to real devices as well as simulators.

Syntax

```
<static> BB7_MAX_STRING_STORAGE_LENGTH
```

Source

SUPStorage.js, line 218 on page 567.

clear() method

Storage

Syntax

clear()

Source

SUPStorage.js, line 302 on page 570.

getItem(key) method

Storage

Syntax

getItem(*key*) {string}

Parameters

Name	Type	Description
<i>key</i>	string	String key corresponding to the requested value.

Returns

A String value corresponding to the key, or null if either the key is not known, or if the key exists but its value was set to null.

Type:

string

Example

```
// Create the SUP Storage
var store = new hwc.SUPStorage ("one");
store.setItem ("foo", "bar"); // add an item.
result = store.getItem ("foo"); // will returns "bar".
result = store.getItem ("fool"); // fool does not exists; will return null.
```

Source

SUPStorage.js, line 170 on page 565.

key(index) method

Storage

Syntax

key(*index*) {string}

Parameters

Name	Type	Description
<i>index</i>	Integer	0-based index to the key. Must be less than the value retrieved by <code>.length</code> .

Returns

The key, or null if the index is invalid.

Type:

string

Example

```
// Create the SUP Storage
var store = new hwc.SUPStorage ("one");
store.setItem ("foo", "bar"); // add an item.
var result = store.key (0); // will returns "foo".
```

Source

SUPStorage.js, line 97 on page 563.

length() method

Storage

Syntax

length()

Example

```
// Create the SUP Storage
var store = new hwc.SUPStorage ("one");
store.setItem ("foo", "bar"); // add an item.
store.setItem ("foo1", "bar"); // add an item.
store.setItem ("foo2", "bar"); // add an item.
var result = store.length; // result = 3
```

Source

SUPStorage.js, line 59 on page 562.

removeItem(key) method

Storage

Syntax

```
removeItem( key )
```

Parameters

Name	Type	Description
<i>key</i>	string	String key to remove.

Example

```
// Create the SUP Storage
var store = new hwc.SUPStorage ("one");
store.setItem ("foo", "bar"); // add an item.
store.removeItem ("foo");
result = store.getItem ("food"); // will be null.
```

Source

SUPStorage.js, line 276 on page 569.

setItem(key, value) method

Storage

Syntax

```
setItem( key, value )
```

Parameters

Name	Type	Description
<i>key</i>	string	String key corresponding to the value.
<i>value</i>	string	String value to store.

Example

```
// Create the SUP Storage
var store = new hwc.SUPStorage ("one");
store.setItem ("foo", "bar"); // add an item.
```

Source

SUPStorage.js, line 233 on page 568.

hwc.SUPStorageException class

Storage

Syntax

```
new SUPStorageException( code, message )
```

Parameters

Name	Type	Description
<i>code</i>	Integer	the error code
<i>message</i>	string	the error message.

Source

SUPStorage.js, line 330 on page 571.

NativeErrorCodes namespace

This object contains constants representing the different types of public native error codes.

Error codes larger than 500 are reserved for server communication errors which may occur as the result of online requests and/or attachment downloads

Members

Name	Description
<i>ATTACHMENT_NOT_DOWNLOADED</i> on page 122	A constant indicating the attachment has not been downloaded.
<i>CERTIFICATE_NOT_SELECTED</i> on page 122	A constant indicating there was no certificate selected by the user.
<i>DEVICE_NOT_CONNECTED</i> on page 123	A constant indicating the device is not connected.
<i>FAIL_TO_SAVE_CERTIFICATE</i> on page 123	A constant indicating a failure to save a certificate.
<i>FAIL_TO_SAVE_CREDENTIAL</i> on page 123	A constant indicating a failure to save a credential.
<i>FILENAME_NO_EXTENSION</i> on page 123	A constant indicating there was a filename without an extension.
<i>INVALID_COMMON_NAME</i> on page 124	A constant indicating an invalid common name was passed while requesting a certificate from Afaria.
<i>NAVIGATION_ERROR</i> on page 124	A constant indicating that opening the URL failed.

<i>REQUIRED_PARAMETER_NOT_AVAILABLE</i> on page 124	A constant indicating a required parameter was not available.
<i>RESPONSE_TOO_LARGE</i> on page 125	A constant indicating the response is too large for a javascript variable.
<i>SSOCERT_EXCEPTION</i> on page 125	A constant indicating there was an SSO certificate manager exception.
<i>UNKNOWN_ERROR</i> on page 125	A constant indicating there was an unknown error.
<i>UNKNOWN_MIME_TYPE</i> on page 125	A constant indicating there was an unknown MIME type.
<i>UNSUPPORTED_ATTACHMENT_TYPE</i> on page 126	A constant indicating the attachment type is not supported.

Source

hwc-comms.js, line 320 on page 444.

ATTACHMENT_NOT_DOWNLOADED member

A constant indicating the attachment has not been downloaded.

Syntax

```
<static> ATTACHMENT_NOT_DOWNLOADED : number
```

Type

number

Source

hwc-comms.js, line 338 on page 444.

CERTIFICATE_NOT_SELECTED member

A constant indicating there was no certificate selected by the user.

Syntax

```
<static> CERTIFICATE_NOT_SELECTED : number
```

Type

number

Source

hwc-comms.js, line 370 on page 445.

DEVICE_NOT_CONNECTED member

A constant indicating the device is not connected.

Syntax

```
<static> DEVICE_NOT_CONNECTED : number
```

Type

number

Source

hwc-comms.js, line 410 on page 447.

FAIL_TO_SAVE_CERTIFICATE member

A constant indicating a failure to save a certificate.

Syntax

```
<static> FAIL_TO_SAVE_CERTIFICATE : number
```

Type

number

Source

hwc-comms.js, line 402 on page 446.

FAIL_TO_SAVE_CREDENTIAL member

A constant indicating a failure to save a credential.

Syntax

```
<static> FAIL_TO_SAVE_CREDENTIAL : number
```

Type

number

Source

hwc-comms.js, line 394 on page 446.

FILENAME_NO_EXTENSION member

A constant indicating there was a filename without an extension.

Syntax

```
<static> FILENAME_NO_EXTENSION : number
```

Type

number

Source

hwc-comms.js, line 354 on page 445.

INVALID_COMMON_NAME member

A constant indicating an invalid common name was passed while requesting a certificate from Afaria.

Syntax

```
<static> INVALID_COMMON_NAME : number
```

Type

number

Source

hwc-comms.js, line 434 on page 447.

NAVIGATION_ERROR member

A constant indicating that opening the URL failed.

Syntax

```
<static> NAVIGATION_ERROR : number
```

Type

number

Source

hwc-comms.js, line 426 on page 447.

REQUIRED_PARAMETER_NOT_AVAILABLE member

A constant indicating a required parameter was not available.

Syntax

```
<static> REQUIRED_PARAMETER_NOT_AVAILABLE : number
```

Type

number

Source

hwc-comms.js, line 362 on page 445.

RESPONSE_TOO_LARGE member

A constant indicating the response is too large for a javascript variable.

Syntax

<static> RESPONSE_TOO_LARGE : number

Type

number

Source

hwc-comms.js, line 418 on page 447.

SSOCERT_EXCEPTION member

A constant indicating there was an SSO certificate manager exception.

Syntax

<static> SSOCERT_EXCEPTION : number

Type

number

Source

hwc-comms.js, line 386 on page 446.

UNKNOWN_ERROR member

A constant indicating there was an unknown error.

Syntax

<static, constant> UNKNOWN_ERROR : number

Source

hwc-comms.js, line 330 on page 444.

UNKNOWN_MIME_TYPE member

A constant indicating there was an unknown MIME type.

Syntax

<static> UNKNOWN_MIME_TYPE : number

Type

number

Source

hwc-comms.js, line 346 on page 444.

UNSUPPORTED_ATTACHMENT_TYPE member

A constant indicating the attachment type is not supported.

Syntax

```
<static> UNSUPPORTED_ATTACHMENT_TYPE : number
```

Type

number

Source

hwc-comms.js, line 378 on page 445.

APP_ADDED member

A constant indicating that a hybrid app has been added.

Used in *anonymous.ApplicationListener* on page 75 callback functions as a possible value for event.

Syntax

```
<static> APP_ADDED : number
```

Type

number

Source

hwc-api.js, line 1640 on page 355.

APP_REFRESH member

A constant indicating that the application list requires a refresh.

Used in *anonymous.ApplicationListener* on page 75 callback functions as a possible value for event.

Syntax

```
<static> APP_REFRESH : number
```


Type

number

Source

hwc-api.js, line 1634 on page 354.

APP_REMOVED member

A constant indicating that a hybrid app was removed.

Used in *anonymous.ApplicationListener* on page 75 callback functions as a possible value for event.

Syntax

<static> APP_REMOVED : number

Type

number

Source

hwc-api.js, line 1652 on page 355.

APP_UPDATED member

A constant indicating that a hybrid app was updated.

Used in *anonymous.ApplicationListener* on page 75 callback functions as a possible value for event.

Syntax

<static> APP_UPDATED : number

Type

number

Source

hwc-api.js, line 1646 on page 355.

CONNECTED member

Constant indicating that the hwc is connected.

Used in *anonymous.ConnectionStateListener* on page 77 callback functions.

Syntax

<static> CONNECTED : number

Type

number

Source

hwc-api.js, line 503 on page 313.

CONNECTION_CONNECTED member

A constant indicating that the log message is about the connection being established.

Used in *anonymous.LogListener* on page 79 callback functions.

Syntax

```
<static> CONNECTION_CONNECTED : number
```

Type

number

Source

hwc-api.js, line 873 on page 326.

CONNECTION_DISCONNECTED member

A constant indicating that the log message is about the connection being disconnected.

Used in *anonymous.LogListener* on page 79 callback functions.

Syntax

```
<static> CONNECTION_DISCONNECTED : number
```

Type

number

Source

hwc-api.js, line 878 on page 326.

CONNECTION_ERROR member

A constant indicating that the log message is about a connection error.

Used in *anonymous.LogListener* on page 79 callback functions.

Syntax

```
<static> CONNECTION_ERROR : number
```

Type

number

Source

hwc-api.js, line 863 on page 326.

CONNECTION_OTHER member

A constant indicating that the log message is not about the connection.

Used in *anonymous.LogListener* on page 79 callback functions.

Syntax

```
<static> CONNECTION_OTHER : number
```

Type

number

Source

hwc-api.js, line 868 on page 326.

CONNECTION_RETRIEVED_ITEMS member

a constant indicating that the log message is about retrieved items.

Used in *anonymous.LogListener* on page 79 callback functions.

Syntax

```
<static> CONNECTION_RETRIEVED_ITEMS : number
```

Type

number

Source

hwc-api.js, line 883 on page 327.

DEFAULT_CUSTOM_ICON_INDEX member

A constant indicating the custom icon index.

Syntax

```
<static> DEFAULT_CUSTOM_ICON_INDEX : number
```

Type

number

Source

hwc-api.js, line 1901 on page 364.

DISCONNECTED member

Constant indicating that the hwc is disconnected.

Used in *anonymous.ConnectionStateListener* on page 77 callback functions.

Syntax

```
<static> DISCONNECTED : number
```

Type

number

Source

hwc-api.js, line 508 on page 313.

e2eTrace member

Represents an E2E Trace.

This object is used for debugging and analysis.

Syntax

```
<static> e2eTrace
```

Source

hwc-api.js, line 3457 on page 420.

isTraceEnabled() method

Gets whether the e2e tracing has been requested to be started.

This function returns true between calls to `hwc.e2eTrace#startTrace` and `hwc.e2eTrace#stopTrace`.

Syntax

```
<static> isTraceEnabled() {boolean}
```

Returns

True if trace is enabled, false otherwise.

Type:

boolean

Source

hwc-api.js, line 3491 on page 422.

setTraceLevel(The) method

Sets the passport e2eTrace level.

This function must be called before hwc.e2eTrace#startTrace.

Syntax

```
<static> setTraceLevel( The )
```

Parameters

Name	Type	Description
<i>The</i>	string	trace level. Must be one of hwc.e2eTrace.TraceLevel.LOW, hwc.e2eTrace.TraceLevel.MEDIUM, or hwc.e2eTrace.TraceLevel.HIGH.

Source

hwc-api.js, line 3507 on page 422.

startTrace() method

Starts tracing user actions and requests.

Before this function is called, the trace level must be set with hwc.e2eTrace#setTracelevel.

Syntax

```
<static> startTrace()
```

Source

hwc-api.js, line 3520 on page 423.

stopTrace() method

Stops tracing user actions and requests.

Syntax

```
<static> stopTrace()
```

Source

hwc-api.js, line 3533 on page 423.

uploadTrace() method

Upload the Business Transaction XML (BTX) to the server.

To upload, the SAP Solution Manager URL must be set in SAP Control Center configuration.

Syntax

```
<static> uploadTrace() {boolean}
```

Returns

True if the upload is successful, false otherwise.

Type:

boolean

Source

hwc-api.js, line 3548 on page 424.

INSTALLATION_BEGIN member

A constant indicating that the application is starting to be installed.

Used in *anonymous.AppInstallationListener* on page 73 callback functions.

Syntax

```
<static> INSTALLATION_BEGIN : number
```

Type

number

Source

hwc-api.js, line 1029 on page 332.

INSTALLATION_END member

A constant indicating that the application has finished being installed.

Used in *anonymous.AppInstallationListener* on page 73 callback functions.

Syntax

```
<static> INSTALLATION_END : number
```

Type

number

Source

hwc-api.js, line 1034 on page 332.

MediaCache member

Represents a Media Cache.

This object gives the option to use the cache when accessing .

Syntax

<static> MediaCache

Source

hwc-api.js, line 3379 on page 417.

Policy member

hwc.MediaCache.Policy An object containing constants representing the different caching policies.

Syntax

<static> Policy

Source

hwc-api.js, line 3385 on page 418.

CACHE_FIRST member

hwc.MediaCache.Policy.CACHE_FIRST Use cache first policy: requests will be served from the cache if possible.

Syntax

<static> CACHE_FIRST : string

Type

string

Source

hwc-api.js, line 3398 on page 418.

SERVER_FIRST member

hwc.MediaCache.Policy.SERVER_FIRST Use server first policy: requests will only be served from the cache if the server is unavailable.

Syntax

<static> SERVER_FIRST : string

Type

string

Source

hwc-api.js, line 3392 on page 418.

getUrl(resourceUrl, [policy]) method

Creates a media cache URL for the resource.

The cache first policy will be used if no policy is specified.

Syntax

<static> getUrl(resourceUrl, [policy]) {string}

Parameters

Name	Type	Argument	Description
<i>resourceUrl</i>	string		The URL to the resource
<i>policy</i>	<i>hwc.MediaCache.Policy</i> on page 133	(optional)	The optional cache policy to use. If set, it must be either <i>hwc.MediaCache.Policy.SERV-ER_FIRST</i> on page 133 or <i>hwc.MediaCache.Policy.CACHE_FIRST</i> on page 133. Default policy is cache first.

Returns

The URL that can be used to access the resource with the specified caching policy.

Type:

string

Example

```
// This line creates a url that can be used to retrieve the picture
// from the cache if possible, and from the server otherwise.
var mediaCacheURL = hwc.MediaCache.getUrl( "http://yourserver.com/
Pictures/pentagon.jpg", hwc.MediaCache.Policy.CACHE_FIRST );
// The following function adds a picture to the page. Since the
// mediaCacheURL variable is used for the url, the picture will be
// retrieved from the cache if possible.
```



```

var addPicFromMediaCache = function()
{
    // Create the image element.
    var image = document.createElement( "img" );
    // Set the source of the image to the media cache URL.
    image.setAttribute( 'src', mediaCacheURL );
    // Add the image element to the page.
    document.body.appendChild( image );
}

// This line creates a url that can be used to retrieve the picture
// from the server if it is available, or the cache otherwise.
var mediaCacheURL_serverFirst = hwc.MediaCache.getUrl( "http://
yourserver.com/Pictures/pentagon.jpg",
hwc.MediaCache.Policy.SERVER_FIRST );
// The following function adds a picture to the page. Since the
// mediaCacheURL_serverFirst variable is used for the url, the picture
// will be gotten
// from the server if the server is available, and from the cache
// otherwise.
var addPicFromMediaCache_ServerFirst = function()
{
    // Create the image element.
    var image = document.createElement( "img" );
    // Set the source of the image to the media cache URL.
    image.setAttribute( 'src', mediaCacheURL_serverFirst );
    // Add the image element to the page.
    document.body.appendChild( image );
}

```

Source

hwc-api.js, line 3442 on page 420.

MSG_ADDED member

A constant indicating that a message has been added.

Used in *anonymous.MessageListener* on page 80 callback functions.

Syntax

<static> MSG_ADDED : number

Type

number

Source

hwc-api.js, line 2888 on page 399.

MSG_PRIORITY_HIGH member

A constant indicating a message has high priority.

Used in *hwc.Message* on page 215.

Syntax

<static> MSG_PRIORITY_HIGH : number

Type

number

Source

hwc-api.js, line 2908 on page 400.

MSG_PRIORITY_NORMAL member

A constant indicating a message has normal priority.

Used in *hwc.Message* on page 215.

Syntax

<static> MSG_PRIORITY_NORMAL : number

Type

number

Source

hwc-api.js, line 2903 on page 400.

MSG_REFRESH member

A constant indicating that a message needs to be refreshed.

Used in *anonymous.MessageListener* on page 80 callback functions.

Syntax

<static> MSG_REFRESH : number

Type

number

Source

hwc-api.js, line 2883 on page 399.

MSG_REMOVED member

A constant indicating that a message has been removed.

Used in *anonymous.MessageListener* on page 80 callback functions.

Syntax

<static> MSG_REMOVED : number

Type

number

Source

hwc-api.js, line 2898 on page 400.

MSG_UPDATED member

A constant indicating that a message has been updated.

Used in *anonymous.MessageListener* on page 80 callback functions.

Syntax

<static> MSG_UPDATED : number

Type

number

Source

hwc-api.js, line 2893 on page 399.

NOTIFICATION_CANCEL member

A constant indicating that no more push notification listeners should be called.

Used as a return value for *anonymous.PushNotificationListener* functions.

Syntax

<static> NOTIFICATION_CANCEL : number

Type

number

Source

hwc-api.js, line 1338 on page 344.

NOTIFICATION_CONTINUE member

A constant indicating that other push notification listeners should continue to be called.

Used as a return value for *anonymous.PushNotificationListener* functions.

Syntax

<static> NOTIFICATION_CONTINUE : number

Type

number

Source

hwc-api.js, line 1332 on page 344.

OPEN_APP_NOT_EXIST member

A constant indicating that *hwc.openApp* on page 221 failed because the specified app does not exist.

This is a possible return value for *hwc.openApp* on page 221.

Syntax

```
<static> OPEN_APP_NOT_EXIST : number
```

Type

number

Source

hwc-api.js, line 1848 on page 362.

OPEN_APP_OTHER member

A constant indicating that *hwc.openApp* on page 221 failed for an unspecified reason.

This is a possible return value for *hwc.openApp* on page 221.

Syntax

```
<static> OPEN_APP_OTHER : number
```

Type

number

Source

hwc-api.js, line 1854 on page 363.

OPEN_APP_SUCCESS member

A constant indicating that *hwc.openApp* on page 221 completed successfully.

This is a possible return value for *hwc.openApp* on page 221.

Syntax

```
<static> OPEN_APP_SUCCESS : number
```

Type

number

Source

hwc-api.js, line 1842 on page 362.

OPEN_MSG_APP_NOT_EXIST member

A constant indicating that a message could not be opened because there was no associated hybrid app.

This is a possible return value for *hwc.openMessage* on page 222.

Syntax

```
<static> OPEN_MSG_APP_NOT_EXIST : number
```

Type

number

Source

hwc-api.js, line 3156 on page 409.

OPEN_MSG_NOT_EXIST member

A constant indicating that a message could not be opened because no message with the given ID exists.

This is a possible return value for *hwc.openMessage* on page 222.

Syntax

```
<static> OPEN_MSG_NOT_EXIST : number
```

Type

number

Source

hwc-api.js, line 3150 on page 409.

OPEN_MSG_OTHER member

A constant indicating that a message could not be opened due to an unspecified error.

This is a possible return value for *hwc.openMessage* on page 222.

Syntax

```
<static> OPEN_MSG_OTHER : number
```

Type

number

Source

hwc-api.js, line 3162 on page 409.

OPEN_MSG_SUCCESS member

A constant indicating that a message was successfully opened.

This is a possible return value for *hwc.openMessage* on page 222.

Syntax

```
<static> OPEN_MSG_SUCCESS : number
```

Type

number

Source

hwc-api.js, line 3144 on page 409.

perf member

Represents the Performance Manager.

Syntax

```
<static> perf
```

Example

```
// Start performance collection.
if (hwc.perf.isEnabled())
{
    hwc.perf.stopInteraction();
}

hwc.perf.startInteraction('someinteraction');

hwc.perf.startInterval('IntervalName', 'CustomType'); // Start an
optional interval.

// Stop performance collection. Logs will be written.
if (hwc.perf.isEnabled())
{
    hwc.perf.stopInterval('IntervalName'); // Stop an optional
interval.
    hwc.perf.stopInteraction();
}
```

Source

hwc-api.js, line 3579 on page 425.

isEnabled() method

Gets whether the performance agent is enabled.

Syntax

```
<static> isEnabled() {boolean}
```

Returns

True if the performance agent is enabled, false otherwise.

Type:

boolean

Source

hwc-api.js, line 3586 on page 425.

startInteraction(interactionName) method

Starts the interaction.

Syntax

```
<static> startInteraction( interactionName )
```

Parameters

Name	Type	Description
<i>interactionName</i>	string	The name of the interaction.

Source

hwc-api.js, line 3600 on page 426.

startInterval(intervalName, intervalType) method

Starts an interval.

Syntax

```
<static> startInterval( intervalName, intervalType )
```

Parameters

Name	Type	Description
<i>intervalName</i>	string	The name of the interval.
<i>intervalType</i>	string	The type of the interval.\

Source

hwc-api.js, line 3628 on page 427.

stopInteraction() method

Stops the interaction.

Syntax

```
<static> stopInteraction()
```

Source

hwc-api.js, line 3613 on page 426.

stopInterval(intervalName) method

Stops the interval.

Syntax

```
<static> stopInterval( intervalName )
```

Parameters

Name	Type	Description
<i>intervalName</i>	string	The name of the interval.

Source

hwc-api.js, line 3642 on page 427.

PictureError member

An array that holds all possible error codes

Syntax

```
<static> PictureError
```

Source

Camera.js, line 83 on page 260.

BAD_OPTIONS member

Constant indicating that the supplied options were not recognized by the *hwc.getPicture* on page 192 method

Syntax

```
<static> BAD_OPTIONS
```

Source

Camera.js, line 113 on page 261.

IN_PROGRESS member

Constant indicating that the *hwc.getPicture* on page 192 method has been invoked, but has not completed yet.

Syntax

```
<static> IN_PROGRESS
```

Source

Camera.js, line 101 on page 260.

NO_ERROR member

Constant indicating that the *hwc.getPicture* on page 192 method was successful.

Syntax

```
<static> NO_ERROR
```

Source

Camera.js, line 89 on page 260.

NOT_SUPPORTED member

Constant indicating that the *hwc.getPicture* on page 192 method is not implemented, camera not present, etc.

Syntax

```
<static> NOT_SUPPORTED
```

Source

Camera.js, line 95 on page 260.

TOO_LARGE member

Constant indicating that the returned image size was too large to be handled by JavaScript.

Syntax

<static> TOO_LARGE

Source

Camera.js, line 119 on page 261.

UNKNOWN member

Constant indicating that an unknown error occurred during the execution of *hwc.getPicture* on page 192 method.

Syntax

<static> UNKNOWN

Source

Camera.js, line 125 on page 261.

USER_REJECT member

Constant indicating that the user has cancelled the *hwc.getPicture* on page 192 invocation.

Syntax

<static> USER_REJECT

Source

Camera.js, line 107 on page 260.

REG_ERR_AUTO_REG_NOT_ENABLED member

Constant indicating that auto registration was not enabled in the template.

Possible return value for *hwc.saveSettings* on page 237.

Syntax

<static> REG_ERR_AUTO_REG_NOT_ENABLED : number

Type

number

Source

hwc-api.js, line 144 on page 299.

REG_ERR_AUTO_REG_TEMPLATE_NOT_FOUND member

Constant indicating that no MBS template was found for given AppId and/or Security configuration.

Possible return value for *hwc.saveSettings* on page 237.

Syntax

<static> REG_ERR_AUTO_REG_TEMPLATE_NOT_FOUND : number

Type

number

Source

hwc-api.js, line 140 on page 299.

REG_ERR_AUTO_REG_USER_NAME_TOO_LONG member

Constant indicating that the user name is longer than the legal limit.

Possible return value for *hwc.saveSettings* on page 237.

Syntax

<static> REG_ERR_AUTO_REG_USER_NAME_TOO_LONG : number

Type

number

Source

hwc-api.js, line 152 on page 299.

REG_ERR_AUTO_REG_WRONG_USER_FOR_DEVICE member

Constant indicating that the given device id is already registered for another user.

Possible return value for *hwc.saveSettings* on page 237.

Syntax

<static> REG_ERR_AUTO_REG_WRONG_USER_FOR_DEVICE : number

Type

number

Source

hwc-api.js, line 148 on page 299.

REG_ERR_COULD_NOT_REACH_MMS_SERVER member

Constant indicating that the connection to the MMS service failed.

Possible return value for *hwc.saveSettings* on page 237.

Syntax

```
<static> REG_ERR_COULD_NOT_REACH_MMS_SERVER : number
```

Type

number

Source

hwc-api.js, line 136 on page 299.

REG_ERR_INVALID_USER_NAME member

Constant indicating that the user name contains invalid characters.

Possible return value for *hwc.saveSettings* on page 237.

Syntax

```
<static> REG_ERR_INVALID_USER_NAME : number
```

Type

number

Source

hwc-api.js, line 156 on page 300.

REG_ERR_MMS_AUTHENTICATION_FAILED member

Constant indicating that MMS Authentication failed.

Possible return value for *hwc.saveSettings* on page 237.

Syntax

```
<static> REG_ERR_MMS_AUTHENTICATION_FAILED : number
```

Type

number

Source

hwc-api.js, line 132 on page 298.

REGISTRATION_METHOD_AFARIA member

Constant indicating that automatic registration using a certificate from Afaria is the preferred method.

Used in *hwc.ConnectionSettings* on page 168.

Syntax

```
<static> REGISTRATION_METHOD_AFARIA : number
```

Type

number

Source

hwc-api.js, line 42 on page 295.

REGISTRATION_METHOD_AUTOMATIC member

Constant indicating that automatic registration using password is the preferred method.

Used in *hwc.ConnectionSettings* on page 168.

Syntax

```
<static> REGISTRATION_METHOD_AUTOMATIC : number
```

Type

number

Source

hwc-api.js, line 34 on page 295.

REGISTRATION_METHOD_CERTIFICATE member

Constant indicating that automatic registration using a local certificate is the preferred method.

Used in *hwc.ConnectionSettings* on page 168.

Syntax

```
<static> REGISTRATION_METHOD_CERTIFICATE : number
```

Type

number

Source

hwc-api.js, line 46 on page 295.

REGISTRATION_METHOD_MANUAL member

Constant indicating that manual registration is the preferred method.

Used in *hwc.ConnectionSettings* on page 168.

Syntax

```
<static> REGISTRATION_METHOD_MANUAL : number
```

Type

number

Source

hwc-api.js, line 38 on page 295.

REGISTRATION_METHOD_NO_PREFERENCE member

Constant indicating no registration method preference.

The application implementation decides the default method to use. This is handled as Manual registration by the HWC. Used in *hwc.ConnectionSettings* on page 168.

Syntax

```
<static> REGISTRATION_METHOD_NO_PREFERENCE : number
```

Type

number

Source

hwc-api.js, line 30 on page 295.

SETTING_SUCCESS member

Constant indicating *hwc.saveSettings* on page 237 completed successfully.

Possible return value for *hwc.saveSettings* on page 237.

Syntax

```
<static> SETTING_SUCCESS : number
```

Type

number

Source

hwc-api.js, line 160 on page 300.

STATUS member

This object contains constants representing the status of the hybrid app.

Syntax

```
<static> STATUS
```

Source

hwc-comms.js, line 166 on page 438.

activationRequired() method

This function sets the activation required state of this hybrid app to true.

After calling this function, the current hybrid app will need to be activated.

Syntax

```
<static> activationRequired()
```

Example

```
hwc.activationRequired();
```

Source

hwc-comms.js, line 773 on page 459.

addAppInstallationListener(AppInstallationListener) method

Register the application installation listener.

Syntax

```
<static> addAppInstallationListener( AppInstallationListener )
```

Parameters

Name	Type	Description
<i>AppInstallationListener</i>	<i>anonymous.AppInstallationListener</i> on page 73	A callback for application installation changes.

Example

```
// appInstallListener is the callback function that will be passed to
hwc.addAppInstallationListener.
var appInstallListener = function( event, moduleId, version,
moduleName )
{
    if( event == hwc.INSTALLATION_BEGIN )
    {
```

```
    alert(moduleName + " has just started the installation
process.");
  }
  else if( event == hwc.INSTALLATION_END )
  {
    alert(moduleName + " has just finished the installation
process.");
  }
}
hwc.addAppInstallationListener( appInstallListener );
```

Source

hwc-api.js, line 946 on page 329.

addAppListener(ApplicationListener, [containingObject]) method

Register the application listener.

Syntax

<static> addAppListener(*ApplicationListener*, [*containingObject*])

Parameters

Name	Type	Argument	Description
<i>ApplicationListener</i>	<i>anonymous.ApplicationListener</i> on page 75		The callback function for application changes.
<i>containingObject</i>	Object	(optional)	The containing object of the listener method. This parameter is only required if the <i>ApplicationListener</i> references the containing object.

Example

```
// This is the callback function that will be passed to
hwc.addAppListener.
var appListener = function( event, moduleId, version )
{
  if( event == hwc.APP_ADDED )
  {
    alert("A hybrid app has been added.");
  }
}
hwc.addAppListener( appListener );
```



```

// appListenerManager is an object that will contain the callback
function as well as variables
// the callback function references.
var appListenerManager = {};
// doSomething is a function that is called from inside the callback
function.
appListenerManager.doSomething = function( event )
{
    if( event == hwc.APP_REMOVED )
    {
        alert("A hybrid app has been removed.");
    }
}
// This is the callback function that will be passed to
hwc.addAppListener. It calls doSomething,
// the definition of which is in the containing function.
appListenerManager.listener = function( event, moduleId, version )
{
    this.doSomething( event );
}
// Since the listener callback function references a variable from
its containing object,
// the containing object must be passed to hwc.addAppListener.
hwc.addAppListener( appListenerManager.listener,
appListenerManager );

```

Source

hwc-api.js, line 1539 on page 351.

addConnectionListener(ConnectionStateListener, [containingObject]) **method**

Register the connection state listener.

Syntax

<static> addConnectionListener(*ConnectionStateListener*, [*containingObject*])

Parameters

Name	Type	Argument	Description
<i>ConnectionStateListener</i>	<i>anonymous.ConnectionStateListener</i> on page 77		Callback for connection state changes.

<i>containingObject</i>	Object	(optional)	Object containing definition for ConnectionStateListener. If a connection state callback function references variables in its containing object, then the containing object should be passed to this function.
-------------------------	--------	------------	--

Example

```
// doSomething is a global function that gets called from the
connection listener.
var doSomething = function()
{
    alert("sample function that gets executed when the hwc becomes
connected");
}
// connectionListener is the callback function that is given to
addConnectionListener.
// When there is a connection event, connectionListener will be
invoked with the details.
var connectionListener = function( event, errorCode, errorMessage )
{
    if( event == hwc.CONNECTED )
    {
        doSomething();
    }
}
hwc.addConnectionListener( connectionListener );

// connectionStateManager is an object that will contain the
connection listener callback as well as
// a variable used by the callback.
var connectionStateManager = {};
// The connectionStateManager keeps track of whether the HWC is
connected or not.
connectionStateManager.connected = false;
// A function called by the listener.
connectionStateManager.doSomething = function()
{
    if( this.connected )
    {
        alert("this alert gets displayed if the hwc is connected");
    }
}
// This is the callback function that will be passed to
addConnectionListener. This callback references variables
// from the containing object (this.connected and this.doSomething),
so when we call addConnectionListener we have
// to give the containing object as the second parameter.
```

```

connectionStateManager.listener = function( event, errorCode,
errorMessage )
{
    if( event == hwc.CONNECTED )
    {
        this.connected = true;
    }
    else
    {
        this.connected = false;
    }
    this.doSomething();
}
// Pass both the listener and the containing object. This enables
the listener to refer to variables in the containing object when it
is invoked.
hwc.addConnectionListener( connectionStateManager.listener,
connectionStateManager );

```

Source

hwc-api.js, line 369 on page 308.

addLogListener(LogListener, [containingObject]) method

Register the log listener.

Syntax

<static> addLogListener(*LogListener*, [*containingObject*])

Parameters

Name	Type	Argument	Description
<i>LogListener</i>	<i>anonymous.LogListener</i> on page 79		Callback for changes to the log.
<i>containingObject</i>	Object	(optional)	Object containing definition for LogListener. If a log listener callback function references variables in its containing object, then the containing object should be passed to this function.

Example

```

// A global function called by the log listener.
var doSomething = function()
{

```

```
    alert("this gets displays when there is a log event.");
}
// The log listener callback function that will be passed to
hwc.addLogListener.
// This function will be invoked whenever there is a log event.
var logListener = function( event, errorCode, errorMessage )
{
    doSomething();
}
// Add the log listener.
hwc.addLogListener( logListener );

// logListenerManager is an object that will contain the listener
callback as well
// as a function that will be invoked from the listener callback
function.
var logListenerManager = {};
// This is a function that is called from the listener callback.
logListenerManager.doSomething = function()
{
    alert("this gets displays when there is a log event.");
}
// This is the listener callback that will be passed to
hwc.addLogListener.
// Since a variable is referenced from the containing object, the
containing object
// will need to be passed to hwc.addLogListener.
logListenerManager.listener = function( event, errorCode,
errorMessage )
{
    this.doSomething();
}
// Pass both the listener callback and the containing object.
hwc.addLogListener( logListenerManager.listener,
logListenerManager );
```

Source

hwc-api.js, line 757 on page 322.

addMenuItemCollection(collection) method

This function adds a menu item collection to the menu items for the screen.

Syntax

```
<static> addMenuItemCollection( collection )
```

Parameters

Name	Type	Description
<i>collection</i>	<i>hwc.MenuItemCollection</i> on page 212	The collection of menu items to add to the screen.

Example

```
var callbackFunctionName = function()
{
    alert( "Menu item clicked!" );
}
var menuItemCollection = new hwc.MenuItemCollection();
menuItemCollection.addMenuItem("menu item name",
"callbackFunctionName()");
hwc.addMenuItemCollection( menuItemCollection );
```

Source

hwc-comms.js, line 669 on page 455.

addMessageListener(filters, MessageListener, [containingObject]) method

Registers a message listener.

Syntax

<static> addMessageListener(*filters*, *MessageListener*, [*containingObject*])

Parameters

Name	Type	Argument	Description
<i>filters</i>	<i>hwc.MessageFilter</i> on page 221		The message filter that message events must pass to get passed to the <i>anonymous.MessageListener</i> on page 80. If no filter is desired, then null can be used for this parameter.
<i>MessageListener</i>	<i>anonymous.MessageListener</i> on page 80		The callback function for message changes.
<i>containingObject</i>	Object	(optional)	The containing object of the message listener. If a message listener callback function references variables in its containing object, then the containing object should be passed to this function.

Example

```
// soSomething is a global function called by the listener callback.
var doSomething = function()
{
    alert("New message!");
}
// messageListener is the callback function passed to
hwc.addMessageListener.
var messageListener = function( flag, messageId )
{
    if( flag == hwc.MSG_ADDED )
    {
        doSomething();
    }
}
// We do not want to filter the message events the listener will get
invoked for, so pass null for the first parameter.
hwc.addMessageListener( null, messageListener );

// someObject is an object that will contain the listener callback as
well as a variable referenced by the callback.
var someObject = {};
// doSomething is a function referenced by the callback function.
someObject.doSomething = function()
{
    alert("New message!");
}
// messageListener is the callback that will be passed to
hwc.addMessageListener.
someObject.messageListener = function( flag, messageId )
{
    if( flag == hwc.MSG_ADDED )
    {
        this.doSomething();
    }
}
// Create a filter so that not all message events will invoke our
callback function.
// Only events about messages with a subject of "Subject" will
trigger our callback function.
var filter = new hwc.MessageFilter( null, "Subject", null, null,
null, null);
// The callback function references a variable in its containing
object, so we need to pass in the containing object
// in addition to the filter and the callback function.
hwc.addMessageListener( filter, someObject.messageListener,
someObject );
```

Source

hwc-api.js, line 2781 on page 395.

addPushNotificationListener(PushNotificationListener, [containingObject])
method

Register a push notification listener.

Syntax

```
<static> addPushNotificationListener( PushNotificationListener,  
[containingObject] )
```

Parameters

Name	Type	Argument	Description
<i>PushNotificationListener</i>	function		The callback for push notifications.
<i>containingObject</i>	Object	(optional)	Object containing definition for PushNotificationListener. If the listener callback function references variables in its containing object, then the containing object should be passed to this function.

Example

```
// pushListener is the callback function that will be passed to  
hwc.addPushNotificationListener.  
var pushListener = function( notifications )  
{  
    alert( "push notification:\n"  
+ JSON.stringify(notifications) );  
    return hwc.NOTIFICATION_CONTINUE;  
}  
hwc.addPushNotificationListener( pushListener );
```

```
// pushListenerManager is an object that will contain the listener  
callback as well as a variable  
// referenced from the callback.  
var pushListenerManager = {};  
// doSomething is a function that is called from inside the callback.  
pushListenerManager.doSomething = function( notifications )  
{  
    alert( "push notification:\n"  
+ JSON.stringify(notifications) );  
    return hwc.NOTIFICATION_CONTINUE;  
}
```

```
// This is the callback function.
pushListenerManager.listener = function( notifications )
{
    return this.doSomething( notifications );
}
// Since the callback function references variables in its containing
object, the containing object
// must be passed to hwc.addPushNotificationListener as well.
hwc.addPushNotificationListener( pushListenerManager.listener,
pushListenerManager );
```

Source

hwc-api.js, line 1242 on page 340.

CertificateStore() method

Use these functions for X.509 credential handling.

Use these functions to create a user interface in HTML and JavaScript, that uses X.509 certificates as the Workflow credentials.

This file contains the functions that allow parsing a certificate date, creating a certificate from a JSON string value, retrieving a certificate from a file (Android), retrieving a certificate from the server (iOS), and so on.

Syntax

```
<static> CertificateStore()
```

Source

Certificate.js, line 45 on page 270.

certificateLabels(filterSubject, filterIssuer) method

Certificate

Syntax

```
certificateLabels( filterSubject, filterIssuer ) {String[]}
```

Parameters

Name	Type	Description
<i>filterSubject</i>	String	filter of subject
<i>filterIssuer</i>	String	filter of issuer

Returns

Only filtered certificate labels

Type:

String[]

Example

```
// The following script gets all the labels for certificates
// with the provided subject and issuer
var certStore = CertificateStore.getDefault();
var labels = certStore.certificateLabels("MyUser", "mydomain.com");
```

Source

Certificate.js, line 112 on page 272.

getDefault() method

Certificate

Syntax

<static> getDefault() {hwc.CertificateStore}

Returns

a certificate without the signedCertificate part set

Type:

hwc.CertificateStore on page 158

Source

Certificate.js, line 144 on page 273.

getPublicCertificate(label) method

Certificate

Syntax

getPublicCertificate(*label*)

Parameters

Name	Type	Description
<i>label</i>	String	label of the desired certificate

Returns

certificate object

Example

```
// The following script gets the certificate data for the first
// certificate to match the provided subject and issuer
var certStore = CertificateStore.getDefault();
var labels = certStore.certificateLabels("MyUser", "mydomain.com");
var cert = certStore.getPublicCertificate(labels[0]);
```

Source

Certificate.js, line 164 on page 274.

getSignedCertificate(label, password) method

Certificate

Syntax

getSignedCertificate(*label*, *password*)

Parameters

Name	Type	Description
<i>label</i>	String	label of the desired certificate
<i>password</i>	String	Access password for the private key of the certificate. Pass null unless the platform requires a password.

Returns

Certificate object

Example

```
// The following script gets the signed certificate data for the
// first
// certificate to match the provided subject and issuer
var certStore = CertificateStore.getDefault();
var labels = certStore.certificateLabels("MyUser", mydomain.com");
var cert = certStore.getSignedCertificate(labels[0]);

var username = cert.subjectCN;
var password = cert.signedCertificate;
```

Source

Certificate.js, line 209 on page 276.

getSignedCertificateFromAfaria(commonName, challengeCode) method
Certificate**Syntax**`getSignedCertificateFromAfaria(commonName, challengeCode)`**Parameters**

Name	Type	Description
<i>commonName</i>	String	Common name used to generate the certificate by Afaria
<i>challengeCode</i>	String	Challenge code for the user so that CA can verify and sign it

Throws

- If called on a platform that is not supported.

Returns

JSON object with CertBlob in Base64 encoded format and other information about certificate

Example

```
// The following script gets a signed certificate from the Afaria
server.
var certStore = CertificateStore.getDefault();
cert = certStore.getSignedCertificateFromAfaria("Your_CN",
"CA_challenge_code");
```

Source*Certificate.js*, line 362 on page 282.*getSignedCertificateFromFile(filePath, password) method*
Certificate**Syntax**`getSignedCertificateFromFile(filePath, password)`**Parameters**

Name	Type	Description
<i>filePath</i>	String	The absolute path to the file.

<i>password</i>	String	The password needed to access the certificate's private data.
-----------------	--------	---

Example

```
// The following script gets the signed certificate data for the first
// p12 file found on the sdcard
var certStore = CertificateStore.getDefault();
var certPaths =
certStore.listAvailableCertificatesFromFileSystem("/sdcard/",
"p12");
var cert = certStore.getSignedCertificateFromFile(certPaths[0],
"password");
```

Source

Certificate.js, line 287 on page 279.

getSignedCertificateFromServer(username, serverPassword, certPassword)

method

Certificate

Syntax

getSignedCertificateFromServer(*username*, *serverPassword*, *certPassword*)

Parameters

Name	Type	Description
<i>username</i>	String	The username for the Windows user (in the form "DOMAIN\username")
<i>serverPassword</i>	String	The password for the Windows user
<i>certPassword</i>	String	The password needed to access the certificate (may be the same or different from the Windows password)

Example

```
// The following script gets the signed certificate data for the
// user MYDOMAIN\MYUSERNAME from the server
var certStore = CertificateStore.getDefault();
cert = certStore.getSignedCertificateFromServer("MYDOMAIN\\
\MYUSERNAME", "myserverpassword", "mycertpassword");
```

Source

Certificate.js, line 325 on page 280.

***listAvailableCertificatesFromFileSystem(sFolder, sFileExtension)* method**

Certificate

Syntax

```
listAvailableCertificatesFromFileSystem( sFolder, sFileExtension )
{String[]}
```

Parameters

Name	Type	Description
<i>sFolder</i>	String	Folder in which to search for files. This should be a full absolute path, based on the root of the device file system. The separator may be either "/" or "\". For example, "\sdcard\mycerts" or "/sdcard/mycerts" is acceptable. Do not include any http prefixes, such as "file:".
<i>sFileExtension</i>	String	File extension to which the list should be restricted. Pass the string expected after the "." in the file name. For example, to match *.p12, pass "p12" as the argument. Pass null to return all files in the folder.

Returns

A list of Strings, each String being the full path name of a matched file in the given folder.

Type:

String[]

Example

```
// The following script gets an array of file paths for files on
// the sdcard with the extension p12
var certStore = CertificateStore.getDefault();
var certPaths = certStore.listAvailableCertificatesFromFileSystem("/sdcard/", "p12");
```

Source

Certificate.js, line 254 on page 277.

clearCache() method

This function clears the contents of the on-device request result cache for the current hybrid app.

Syntax

<static> clearCache()

Example

```
hwc.clearCache();
```

Source

hwc-comms.js, line 838 on page 461.

clearCacheItem(cachekey) method

This function clears an item from the contents of the on-device request result cache for the current hybrid app.

Syntax

<static> clearCacheItem(*cachekey*)

Parameters

Name	Type	Description
<i>cachekey</i>	string	The key for the cache item to be removed. This is the same key that was passed to hwc.doOnlineRequest.

Example

```
// The cache key is set when calling hwc.doOnlineRequest_CONT  
hwc.doOnlineRequest( . . . , cacheKey, . . . );  
// At some later point if we want to clear the cache for the above  
request, we use the following code:  
hwc.clearCacheItem( cacheKey );
```

Source

hwc-comms.js, line 860 on page 462.

ClientVariables(clientVariablesVersion, clientVariableItems) method

Represents a ClientVariables object.

Syntax

```
<static> ClientVariables( clientVariablesVersion, clientVariableItems )
```

Parameters

Name	Type	Description
<i>clientVariablesVersion</i>	number	The version of client variables.
<i>clientVariableItems</i>	Object	The json object that contains key/value pairs of client variable items.

Source

hwc-api.js, line 2000 on page 368.

ITEM_NOT_FOUND member

A constant indicating that a variable does not exist in a *hwc.ClientVariables* on page 165 object.

Syntax

```
<static> ITEM_NOT_FOUND : number
```

Type

number

Source

hwc-api.js, line 2120 on page 372.

this.containsName(variableName) method

Check if this *hwc.ClientVariables* on page 165 has a variable by the given name.

Syntax

```
<static> this.containsName( variableName ) {boolean}
```

Parameters

Name	Type	Description
------	------	-------------

<i>variableName</i>	string	The name of variable to check for.
---------------------	--------	------------------------------------

Returns

True if this *hwc.Client Variables* on page 165 has a variable by the given name, false otherwise.

Type:

boolean

Source

hwc-api.js, line 2070 on page 370.

this.getAllVariableNames() method

Gets an array containing the names of all variables in this *hwc.Client Variables* on page 165.

Syntax

```
<static> this.getAllVariableNames() {string[]}
```

Returns

The array holding the names of all variables contained in this *hwc.Client Variables* on page 165.

Type:

string[]

Source

hwc-api.js, line 2036 on page 369.

this.getCount() method

Gets the number of variables this *hwc.Client Variables* on page 165 contains.

Syntax

```
<static> this.getCount() {number}
```

Returns

The number of variables.

Type:

number

Source

hwc-api.js, line 2022 on page 369.

this.getVariableValueByName(variableName) method

Gets the value of the variable with the given name.

If this *hwc.ClientVariables* on page 165 does not have a variable by the given name, a *hwc.ClientVariablesException* on page 168 will be thrown.

Syntax

```
<static> this.getVariableValueByName( variableName ) {string}
```

Parameters

Name	Type	Description
<i>variableName</i>	string	The name of the variable to get the value of.

Throws

- This exception is thrown when there is no variable by the given name in this *hwc.ClientVariables* on page 165.
- Type:
hwc.ClientVariableException

Returns

The value of the variable.

Type:

string

Source

hwc-api.js, line 2092 on page 371.

this.getVersion() method

Gets the version of the client variables.

Syntax

```
<static> this.getVersion() {number}
```

Returns

The version of the client variables.

Type:

number

Source

hwc-api.js, line 2011 on page 368.

ClientVariablesException(errorCode, errMsg) method

This exception is thrown when `hwc.ClientVariables#getVariableValueByName` is called with a variable name that does not exist.

Syntax

```
<static> ClientVariablesException( errorCode, errMsg )
```

Parameters

Name	Type	Description
<i>errorCode</i>	number	The error code (will be <i>hwc.ClientVariables.ITEM_NOT_FOUND</i> on page 165).
<i>errMsg</i>	string	A message describing the error.

Source

hwc-api.js, line 2111 on page 372.

close() method

This function closes the hybrid app.

Syntax

```
<static> close()
```

Example

```
hwc.close();
```

Source

hwc-comms.js, line 1564 on page 488.

ConnectionSettings(regmethod, server, port, server, user, activationcode, protocol, password, urlsuffix) method

Represents the connection settings for connecting to the SUP Server.

Used in *hwc.loadSettings* on page 207 and *hwc.saveSettings* on page 237.

Syntax

```
<static> ConnectionSettings( regmethod, server, port, server, user, activationcode,  
protocol, password, urlsuffix )
```

Parameters

Name	Type	Description
<i>regmethod</i>	number	A number representing the registration method (must be one of <i>hwc.REGISTRATION_METHOD_NO_PREFERENCE</i> on page 148, <i>hwc.REGISTRATION_METHOD_MANUAL</i> on page 148, <i>hwc.REGISTRATION_METHOD_AUTOMATIC</i> on page 147, <i>hwc.REGISTRATION_METHOD_AFARIA</i> on page 147, <i>hwc.REGISTRATION_METHOD_CERTIFICATE</i> on page 147).
<i>server</i>	string	The SUP/Relay server name.
<i>port</i>	number	The SUP/Relay server port number.
<i>server</i>	string	The farm id.
<i>user</i>	string	The user name.
<i>activationcode</i>	string	The activation code.
<i>protocol</i>	string	The protocol to use. Must be "HTTP" or "HTTPS".
<i>password</i>	string	The password for automatic registration.
<i>urlsuffix</i>	string	The url suffix (used only when connecting to a relay server).

Example

```
// Create a new ConnectionSettings object.  
var connectionSettings = new  
hwc.ConnectionSettings( hwc.REGISTRATION_METHOD_MANUAL,  
                        "999.999.999.999",
```

```

5001,
0,
"sampleUsername",
123,
"HTTP",
"samplePassword",
"/" );

// Use the ConnectionSettings object we just created to set the
connection settings.
hwc.saveSettings( connectionSettings );

```

Source

hwc-api.js, line 79 on page 297.

connectToServer([onNotification]) method

Resumes the connection to the SUP server.

Companion function to *hwc.disconnectFromServer* on page 175. This function should only be called after the connection to the SUP server has been suspended with a call to *hwc.disconnectFromServer* on page 175.

Syntax

<static> connectToServer([onNotification])

Parameters

Name	Type	Argument	Description
<i>onNotification</i>	<i>anonymous.LogListener</i> on page 79	(optional)	A log listener callback function.If you are interested in the connection state it is recommended that you call <i>hwc.addConnectionListener</i> on page 151 before calling <i>hwc.connectToServer</i> .

Example

```

hwc.connectToServer();

// Add a log listener while calling hwc.connectToServer.
var logListener = function( time, event, message )
{
    alert(message);
}
hwc.connectToServer( logListener );

```

*Source**hwc-api.js, line 601 on page 316.***convertLocalTimeToUtc(date) method**

Timezone

Syntax`<static> convertLocalTimeToUtc(date) {Date}`*Parameters*

Name	Type	Description
<i>date</i>	Date	Date to be converted, initialized to some valid local time.

Returns

Returns the converted Date object.

Type:

Date

Example

```
var utcDate = hwc.convertLocalTimeToUtc( date );
```

*Source**Timezone.js, line 238 on page 582.***convertUtcToLocalTime(date) method**

Timezone

Syntax`<static> convertUtcToLocalTime(date) {Date}`*Parameters*

Name	Type	Description
<i>date</i>	Date	Date to be converted, initialized to some valid UTC time.

Returns

Returns the converted Date object.

Type:

Date

Example

```
var localDate = hwc.convertUtcToLocalTime( date );
```

Source

Timezone.js, line 210 on page 581.

CustomIcon(width, height, type, name, path, processedPath, moduleId, moduleVersion, index) method

Represents a CustomIcon.

Used with the *hwc.HybridApp* on page 197 object.

Syntax

```
<static> CustomIcon( width, height, type, name, path, processedPath, moduleId, moduleVersion, index )
```

Parameters

Name	Type	Description
<i>width</i>	number	The width of this custom icon.
<i>height</i>	number	The height of this custom icon.
<i>type</i>	string	The image type of this custom icon.
<i>name</i>	string	The name of this custom icon.
<i>path</i>	string	The file path of the unprocessed icon.
<i>processedPath</i>	string	The file path of the processed icon.
<i>moduleId</i>	number	The module ID of the hybrid app this icon is for.
<i>moduleVersion</i>	number	The module version of the hybrid app this icon is for.
<i>index</i>	number	The index of this custom icon.

Source

hwc-api.js, line 2137 on page 373.

this.getHeight() method

Gets the height of this custom icon.

Syntax

```
<static> this.getHeight() {number}
```

Returns

The height of this custom icon.

Type:

number

Source

hwc-api.js, line 2166 on page 374.

this.getImagePath() method

Gets the file path of the unprocessed icon.

Syntax

```
<static> this.getImagePath() {string}
```

Returns

The file path of the unprocessed icon.

Type:

string

Source

hwc-api.js, line 2199 on page 375.

this.getName() method

Gets the name of this custom icon.

Syntax

```
<static> this.getName() {string}
```

Returns

The name of this custom icon.

Type:

string

Source

hwc-api.js, line 2188 on page 375.

this.getProcessedImagePath() method

Gets the file path of the processed icon.

Syntax

```
<static> this.getProcessedImagePath() {string}
```

Returns

The file path of the processed icon.

Type:

string

Source

hwc-api.js, line 2210 on page 375.

this.getType() method

Gets the image type of this custom icon.

Syntax

```
<static> this.getType() {string}
```

Returns

The file type of the image.

Type:

string

Source

hwc-api.js, line 2177 on page 374.

this.getWidth() method

Gets the width of this custom icon.

Syntax

```
<static> this.getWidth() {number}
```

Returns

The width of this custom icon.

Type:

number

Source

hwc-api.js, line 2155 on page 373.

disconnectFromServer() method

Suspends the connection to the SUP server.

Companion function to *hwc.connectToServer* on page 170.

Syntax

```
<static> disconnectFromServer()
```

Example

```
hwc.disconnectFromServer();
```

Source

hwc-api.js, line 640 on page 318.

expireCredentials() method

Allows the user to set the credentials to the expired state for the current hybrid app.

Syntax

```
<static> expireCredentials()
```

Example

```
hwc.expireCredentials();
```

Source

hwc-comms.js, line 822 on page 461.

getAllMessages([messageFilter], [completeList]) method

Gets received messages based on a filter and the existence of a default hybrid app.

Syntax

```
<static> getAllMessages( [messageFilter], [completeList] ) {hwc.Message[]}
```

Parameters

Name	Type	Argument	Description
------	------	----------	-------------

<i>messageFilter</i>	<i>hwc.MessageFilter</i> on page 221	(optional)	A filter that all returned messages will pass.If you do not want to filter based on a certain attribute, use null for that attribute when creating the filter. If you do not want to filter at all, pass in null for this parameter or do not pass in this parameter at all.
<i>completeList</i>	boolean	(optional)	If this parameter is set to true, then all messages will be returned.If this parameter is set to false or if it is not set, then if there is a default hybrid app only the messages belonging to the default hybrid app will be returned (and if there is no default hybrid app all messages will be returned).

Returns

An array of *hwc.Message* on page 215 objects - the received messages.

Type:

`hwc.Message[]`

Example

```
// get all messages that have the subject "a subject".
var filter = new hwc.MessageFilter( null, "a subject", null, null,
null, null );
var messages = hwc.getAllMessages( filter );
```

```
// Get all messages without filtering, but if there is a default
hybrid app only return its messages.
var messages = hwc.getAllMessages();
```

```
// Get all messages (without filtering) for all hybrid apps, even if
there is a default hybrid app.
var messages = hwc.getAllMessages( null, true );
```

Source

hwc-api.js, line 2945 on page 401.

getAppByID(moduleID, version) method

Gets a *hwc.HybridApp* on page 197 object with the given module id and version.

Syntax

```
<static> getAppByID( moduleID, version ) {hwc.HybridApp}
```

Parameters

Name	Type	Description
<i>moduleID</i>	number	The module ID of the hybrid app.
<i>version</i>	number	The version of the hybrid app.

Returns

The hybrid app object, or null if there is no hybrid app with the given ID and version.

Type:

hwc.HybridApp on page 197

Example

```
// Messages do not have a direct link to the hybrid app they belong
// to. Instead they have
// the module ID and version of the hybrid app they belong to. If you
// have a message and
// need to access its hybrid app, first you must call hwc.getAppByID.
var messages = hwc.getAllMessages();
if( messages.length > 0 )
{
  var app = hwc.getAppByID( messages[0].getModuleId(),
messages[0].getModuleVersion() );
}
```

Source

hwc-api.js, line 1810 on page 361.

getAppIconUrl(app, processed) method

This function gets the URL of the icon for a hybrid app depending on whether custom icons are defined.

Syntax

```
<static> getAppIconUrl( app, processed ) {string}
```

Parameters

Name	Type	Description
<i>app</i>	<i>hwc.HybridApp</i> on page 197	The hybrid app for which the icon URL is desired.
<i>processed</i>	boolean	Whether to get the URL of the processed icon (true) or the URL of the unprocessed icon (false).

Returns

The URL of the icon.

Type:

string

Example

```
var apps = hwc.getInstalledApps();
if( apps.length > 0 )
{
    var hybridApp = apps[0];
    // Create the image element.
    var hybridAppIcon = document.createElement("img");
    // Set the source of the image to the icon URL.
    hybridAppIcon.setAttribute( 'src', hwc.getAppIconUrl( hybridApp,
false ) );
    // Add the image element to the page.
    document.body.appendChild( hybridAppIcon );
}
```

Source

hwc-api.js, line 2406 on page 382.

getApplicationConnectionID() method

Gets the Hybrid Web Container application connection ID.

Syntax

```
<static> getApplicationConnectionID() {string}
```

Returns

Application connection ID

Type:

string

Example

```
var appConnectionID = hwc.getApplicationConnectionID();
```

Source

hwc-api.js, line 1912 on page 365.

getBuiltInIconUrl(iconIndex, processed) method

Gets the icon URL for the built-in icon.

This function is used by *hwc.getMsgIconUrl* on page 189 and *hwc.getAppIconUrl* on page 177. It is possible to call this function directly, but generally it is easier simply to call *hwc.getAppIconUrl* on page 177 or *hwc.getMsgIconUrl* on page 189 instead. Those functions handle both cases where there is and isn't a custom icon for the hybrid app or message.

Syntax

```
<static> getBuiltInIconUrl( iconIndex, processed ) {string}
```

Parameters

Name	Type	Description
<i>iconIndex</i>	number	The index of the built-in icon.
<i>processed</i>	boolean	Whether or not to get the URL of the processed icon (true) or the unprocessed icon (false).

Returns

The URL to the icon.

Type:

string

Example

```
// Create the image element.
var builtInIcon = document.createElement( "img" );
// Set the source of the image to the icon URL.
builtInIcon.setAttribute( 'src', hwc.getBuiltInIconUrl(56, false) );
// Add the image element to the page.
document.body.appendChild( builtInIcon );
```

Source

hwc-api.js, line 2342 on page 380.

getCallbackFromNativeError(errString) method

Extract the error call back method name from a URL string.

The parameter name of the error call back method should be "onErrorCallback".

Syntax

```
<static> getCallbackFromNativeError( errString ) {String}
```

Parameters

Name	Type	Description
<i>errString</i>	String	The error string URL

Returns

the error callback method name

Type:

String

Source

hwc-utils.js, line 161 on page 496.

getClientVariables(moduleID, version) method

Gets the client variables of the hybrid app with given module id and version.

Syntax

```
<static> getClientVariables( moduleID, version ) {hwc.ClientVariables}
```

Parameters

Name	Type	Description
<i>moduleID</i>	number	The module ID of the hybrid app.
<i>version</i>	number	The version of the hybrid app.

Returns

A *hwc.ClientVariables* on page 165 object, or null if there are no ClientVariables for the hybrid app with the given module id and version.

Type:

hwc.ClientVariables on page 165

Example

```
var apps = hwc.getInstalledApps();
// Loop through the apps, showing the client variables for each one.
for( var i = 0; i < apps.length; i++ )
{
    var app = apps[i];
    // Get the client variables.
    var clientVariables = hwc.getClientVariables( app.getModuleID(),
app.getVersion() );
    if( clientVariables.getCount() > 0 )
    {
        // Get all the names of the variables for this app.
        var keys = clientVariables.getAllVariableNames();
        // Loop through all the variable for this app.
        for( var index = 0; index < keys.length; index++ )
        {
            // Get a specific variable by name.
            var variable =
clientVariables.getVariableValueByName( keys[index] );
            alert( "variable name: " + keys[index] + "\
variable value: " + variable );
        }
    }
}
```

Source

hwc-api.js, line 1961 on page 367.

getCodeFromNativeError(errString) method

Extract an error code from a URL string.

The parameter name of the error code should be "errCode".

Syntax

```
<static> getCodeFromNativeError( errString ) {String}
```

Parameters

Name	Type	Description
<i>errString</i>	String	The error string URL

Returns

error code

Type:

String

Example

```
TODO: CONFIRM THE RETURN DATATYPE
```

Source

hwc-utils.js, line 179 on page 496.

getCurrentApp() method

Gets the hybrid app that is currently open.

Syntax

```
<static> getCurrentApp() {hwc.HybridApp}
```

Returns

The hybrid app that is currently open.

Type:

hwc.HybridApp on page 197

Example

```
var openHybridApp = hwc.getCurrentApp();
```

Source

hwc-api.js, line 1674 on page 356.

getCurrentLocale() method

Timezone

Syntax

```
<static> getCurrentLocale() {string}
```

Returns

Returns a string containing the current locale, or null if it is not available.

Type:

string

Example

```
var sLocale = hwc.getCurrentLocale();
```

Source

Timezone.js, line 30 on page 574.

getCustomIconUrl(moduleId, moduleVersion, iconIndex, processed) method

Gets the URL to the custom icon.

This function is used by `hwc.CustomIcon#getIconUrl`.

Syntax

```
<static> getCustomIconUrl( moduleId, moduleVersion, iconIndex, processed ) {string}
```

Parameters

Name	Type	Description
<i>moduleId</i>	number	The module Id of the hybrid app the custom icon belongs to.
<i>moduleVersion</i>	number	The version of the hybrid app the custom icon belongs to.
<i>iconIndex</i>	number	The index of the custom icon.
<i>processed</i>	boolean	Whether to get the processed icon (true), or the unprocessed icon (false).

Returns

The URL to the target icon.

Type:

string

Source

hwc-api.js, line 2318 on page 379.

getDstOffsetAtGivenTimeInMinutes(date) method

Timezone

Syntax

```
<static> getDstOffsetAtGivenTimeInMinutes( date ) {int}
```

Parameters

Name	Type	Description
<i>date</i>	Date	Date at which to determine day-light savings offset.

Returns

Returns the number of minutes offset for daylight savings for the current timezone and at the given Date, or 0 if the current timezone doesn't practice daylight savings.

Type:

int

Example

```
var idstOffsetAtTime = hwc.getDstOffsetAtGivenTimeInMinutes (date);
```

Source

Timezone.js, line 438 on page 589.

getExternalResource(url, options) method

Makes an external cross domain request.

Syntax

```
<static> getExternalResource( url, options )
```

Parameters

Name	Type	Description
<i>url</i>	String	The url to make request to
<i>options</i>	<i>anonymous.options</i> on page 72	a set of key/value pairs that configure the underlying request.

Example

```
var options = {
  method: "GET",
  data: "data",
  async: true,
  headers: {
    "Content-Type": "text/plain;charset=UTF-8"
  },
  complete: function(response) {
    // invoked when the request completes (asynchronous mode)
    if (response.status === 200)
      alert("Update successful");
    else
      alert("Update Failed");
  }
};

getExternalResource(url, options);
```

Source

ExternalResource.js, line 49 on page 284.

getInstalledApps([completeList]) method

Returns an array of *hwc.HybridApp* on page 197 objects.

Syntax

```
<static> getInstalledApps( [completeList] ) {hwc.HybridApp[]}
```

Parameters

Name	Type	Argument	Description
<i>completeList</i>	boolean	(optional)	If this parameter is set to true, then all apps that are user invocable or require activation will be returned. If set to false or if it is not set, then if there is a default hybrid app only the default hybrid app will be returned (and if there is no default hybrid app it will return all hybrid apps that are user invocable or require activation).

Returns

An array of hybrid app objects.

Type:

`hwc.HybridApp[]`

Example

```
var apps = hwc.getInstalledApps();
```

```
var apps = hwc.getInstalledApps( true );
```

Source

hwc-api.js, line 1717 on page 357.

getLocalizedDate(date) method

Timezone

Syntax

```
<static> getLocalizedDate( date ) {string}
```

Parameters

Name	Type	Description
<i>date</i>	Date	Date to be localized, initialized to some valid time.

Returns

Returns a localized date string, or undefined if platform is unsupported.

Type:

string

Example

```
var sD = hwc.getLocalizedDate( date );
```

Source

Timezone.js, line 119 on page 577.

getLocalizedDateTime(date) method

Timezone

Syntax

```
<static> getLocalizedDateTime( date ) {string}
```

Parameters

Name	Type	Description
<i>date</i>	Date	Date to be localized, initialized to some valid time.

Returns

Returns a localized date/time string, or undefined if platform is unsupported.

Type:

string

Example

```
var sDT = hwc.getLocalizedDateTime( date );
```

Source

Timezone.js, line 75 on page 576.

getLocalizedTime(date) method

Timezone

Syntax

```
<static> getLocalizedTime( date ) {string}
```

Parameters

Name	Type	Description
<i>date</i>	Date	Date to be localized, initialized to some valid time.

Returns

Returns a localized time string, or undefined if platform is unsupported.

Type:

string

Example

```
var sT = hwc.getLocalizedTime( date );
```

Source

Timezone.js, line 163 on page 579.

getLogEntries() method

Call this function to get an array of *hwc.LogEntry* on page 209 objects.

There will be one *hwc.LogEntry* on page 209 object for each line in the HWC log.

Syntax

```
<static> getLogEntries() {hwc.LogEntry[]}
```

Returns

An array of *hwc.LogEntry* objects.

Type:

`hwc.LogEntry[]`

Example

```
var log = hwc.getLogEntries();
```

Source

hwc-api.js, line 1047 on page 333.

getLoggingAlertDialog() method

This function gets the callback used by `hwc.log` when it is required to notify the user of a log item.

Syntax

```
<static> getLoggingAlertDialog() {anonymous.alertDialogCallbackFunction}
```

Returns

The alert dialog callback function.

Type:

anonymous.alertDialogCallbackFunction on page 73

Source

hwc-comms.js, line 219 on page 440.

getLoggingCurrentLevel() method

This function gets the logging level.

Syntax

```
<static> getLoggingCurrentLevel() {number}
```

Returns

A number representing the logging level. Will be an integer in the range [1..4]. The higher numbers represent more verbose logging levels from 1 for ERROR level logging up to 4 for DEBUG level logging.

Type:

number

Example

```
// Get the logging level  
var loggingLevel = hwc.getLoggingCurrentLevel();
```

Source

hwc-comms.js, line 252 on page 441.

getMessageByID(msgId) method

Gets a *hwc.Message* on page 215 object with the given message ID.

Syntax

```
<static> getMessageByID( msgId ) {hwc.Message}
```

Parameters

Name	Type	Description
<i>msgId</i>	number	The message ID of the message to get.

Returns

A message object, or null if no message with given ID.

Type:

hwc.Message on page 215

Example

```
// A message listener is one place that would likely need to call
hwc.getMessageByID.
var messageListener = function( flag, messageID )
{
  // Since the callback function only gets the messageID, not the
  message itself, if we want
  // more information about the message we must call
  hwc.getMessageByID.
  var message = hwc.getMessageByID( messageID );
  if( message.getSubject() == "a special subject" )
  {
    alert( "An event occurred for a special message!" );
  }
}
hwc.addMessageListener( null, messageListener );
```

Source

hwc-api.js, line 3034 on page 405.

getMsgIconUrl(msg) method

This function gets the URL of the icon for a message object depending on its processed status and whether there are custom icons defined.

Syntax

```
<static> getMsgIconUrl( msg ) {string}
```

Parameters

Name	Type	Description
<i>msg</i>	<i>hwc.Message</i> on page 215	The message object

Returns

The url to access the icon.

Type:

string

Example

```
var messages = hwc.getAllMessages();
if( messages.length > 0 )
{
    // Create the image element.
    var messageIcon = document.createElement("img");
    // Set the source of the image to the icon URL.
    messageIcon.setAttribute( 'src',
hwc.getMsgIconUrl( messages[0] ) );
    // Add the image element to the page.
    document.body.appendChild( messageIcon );
}
```

Source

hwc-api.js, line 2368 on page 381.

getNativeMessageFromNativeError(errString) method

Extract a native message from a URL string.

The parameter name of the native message should be "nativeErrMsg".

Syntax

```
<static> getNativeMessageFromNativeError( errString ) {String}
```

Parameters

Name	Type	Description
<i>errString</i>	String	The error string URL

Returns

the native message

Type:

String

Source

hwc-utils.js, line 195 on page 497.

getOffsetFromUTC(date) method

Timezone

Syntax

```
<static> getOffsetFromUTC( date ) {int}
```

Parameters

Name	Type	Description
<i>date</i>	Date	Date at which time to determine offset, initialized to some valid time.

Returns

Returns the GMT offset in minutes.

Type:

int

Example

```
var totalOffset = hwc.getOffsetFromUTC( date );
```

Source

Timezone.js, line 269 on page 583.

getOnErrorMessageFromNativeError(errString) method

Extract the error message from a URL string.

The parameter name of the error message should be "onErrorMsg".

Syntax

```
<static> getOnErrorMessageFromNativeError( errString ) {String}
```

Parameters

Name	Type	Description
<i>errString</i>	String	The error string URL

Returns

error message

Type:

String

Source

hwc-utils.js, line 138 on page 495.

getPicture(onGetPictureError, onGetPictureSuccess, options) method

Camera

Syntax

<static> getPicture(*onGetPictureError*, *onGetPictureSuccess*, *options*)

Parameters

Name	Type	Description
<i>onGetPictureError</i>	<i>anonymous.onGetPictureError</i> on page 81	Function to be invoked if the attempt to get a picture fails.err will be one of the PictureError codes.
<i>onGetPictureSuccess</i>	<i>anonymous.onGetPictureSuccess</i> on page 82	Function to be invoked if a picture is successfully retrieved.response will either be a Base64-encoded JPG string or a URI.
<i>options</i>	<i>anonymous.PictureOptions</i> on page 72	the options to control the sourceType and destinationType.

Example

```
// Error handler. will be invoked asynchronously.
fail = function(errorCode){
    // handle error code and take appropriate action.
}
// Success handler. will be invoked asynchronously.
success = function(fileName, content){
    // handle the content. content may be a location or base64
    encoded string that is
    // determined by the options passed to the destinationType
    argument.
}

getPicture(fail,
           success,
```

```
{ sourceType: PictureOption.SourceType.CAMERA,
  destinationType: PictureOption.DestinationType.IMAGE_URI
});
```

Source

Camera.js, line 161 on page 262.

getQueryVariable(variable) method

This function looks in the query string on the URL for the value corresponding to the given name.

Syntax

```
<static> getQueryVariable( variable ) {string}
```

Parameters

Name	Type	Description
<i>variable</i>	string	The name of the variable in the URL to retrieve the value for.

Returns

The value corresponding to the given name.

Type:

string

Example

```
// Get the pageToShow variable from the URL query string
var pageToShow = hwc.getQueryVariable( "pageToShow" );
```

Source

hwc-comms.js, line 301 on page 443.

getServerInitiatedApps() method

Returns an array of *hwc.HybridApp* on page 197 objects that are server initiated.

Syntax

```
<static> getServerInitiatedApps() {hwc.HybridApp[]}
```

Returns

An array of server initiated hybrid apps.

Type:

`hwc.HybridApp[]`

Example

```
var serverInitiatedApps = hwc.getServerInitiatedApps();
```

Source

hwc-api.js, line 1763 on page 359.

getSharedStorageKey() method

Storage

Syntax

```
<static> getSharedStorageKey() {string}
```

Returns

the shared storage key.

Type:

string

Source

SUPStorage.js, line 348 on page 572.

getTimezoneId() method

Timezone

Syntax

```
<static> getTimezoneId() {string}
```

Returns

Returns a string containing the current Timezone's standard name.

Type:

string

Example

```
var sTzId = hwc.getTimezoneId();
```

Source

Timezone.js, line 523 on page 592.

getTransformData() method

Returns the transform data for the hybridapp.

Only a server-initiated app will have this data.

Syntax

```
<static> getTransformData()
```

Returns

the transform data.

Example

TODO: Add an example

Source

hwc-utils.js, line 59 on page 492.

getURLParamFromNativeError(paramName, url) method

Extract a parameter value from a URL string with a given parameter name.

Syntax

```
<static> getURLParamFromNativeError( paramName, url ) {String}
```

Parameters

Name	Type	Description
<i>paramName</i>	String	The parameter name
<i>url</i>	String	The containing URL of the parameter

Returns

The parameter value

Type:

String

Source

hwc-utils.js, line 212 on page 497.

getUsesDST() method

Timezone

Syntax

```
<static> getUsesDST() {boolean}
```

Returns

Returns true iff the device's current timezone practices daylight savings, irrespective of whether daylight savings is currently in effect.

Type:

boolean

Example

```
var isDstAware = hwc.getUsesDST();
```

Source

Timezone.js, line 568 on page 593.

getXMLHttpRequest() method

Reliably returns an XMLHttpRequest object regardless of what platform this code is being executed on.

Syntax

```
<static> getXMLHttpRequest() {object}
```

Returns

An XMLHttpRequest object.

Type:

object

Example

```
var request = hwc.getXMLHttpRequest();
```

Source

hwc-comms.js, line 470 on page 448.

guid() method

This function generates a GUID (globally unique identifier).

Syntax

```
<static> guid() {string}
```

Returns

The generated GUID.

Type:

string

Example

```
var globallyUniqueName = hwc.guid();
```

Source

hwc-comms.js, line 457 on page 448.

hideProgressDialog() method

This function hides the progress dialog displaying the spinner.

This function should be used to hide the progress dialog after a call to *hwc.showProgressDialog* on page 244. If this function is called while there is no progress dialog, then nothing will happen.

Syntax

```
<static> hideProgressDialog()
```

Example

```
var showProgress = function()
{
    hwc.showProgressDialog( "a message" );
    setTimeout( hideProgress, 10000 );
}

var hideProgress = function()
{
    hwc.hideProgressDialog();
}
```

Source

hwc-comms.js, line 1502 on page 486.

HybridApp(moduleId, version, displayName, iconIndex, defaultCustomIcon, customIconList) method

This object represents a hybrid app.

Syntax

```
<static> HybridApp( moduleId, version, displayName, iconIndex, defaultCustomIcon,
customIconList )
```

Parameters

Name	Type	Description
<i>moduleId</i>	number	The module id of this hybrid app.
<i>version</i>	number	The version of this hybrid app.
<i>displayName</i>	string	The display name of this hybrid app.
<i>iconIndex</i>	number	The index specifying the icon representing this Hybrid App.
<i>defaultCustomIcon</i>	<i>hwc.CustomIcon</i> on page 172	The default custom icon for this hybrid app.
<i>customIconList</i>	<i>hwc.CustomIcon</i> []	An array of custom icon objects.

Source

hwc-api.js, line 1363 on page 345.

this.getClientVariables() method

Return a *hwc.ClientVariables* on page 165 object for the given module id and version.

Syntax

```
<static> this.getClientVariables() {hwc.ClientVariables}
```

Returns

The *hwc.ClientVariables* on page 165 object for this hybrid app.

Type:

hwc.ClientVariables on page 165

Source

hwc-api.js, line 1444 on page 347.

this.getCustomIconList() method

Gets the list of custom icons associated with this hybrid app.

Syntax

```
<static> this.getCustomIconList() {hwc.CustomIcon[]}
```


Returns

The array of custom icon objects. Null if this hybrid app has no custom icons.

Type:

`hwc.CustomIcon[]`

Source

hwc-api.js, line 1433 on page 347.

this.getDefaultCustomIcon() method

Gets the default custom icon object of this hybrid app.

Syntax

```
<static> this.getDefaultCustomIcon() {hwc.CustomIcon}
```

Returns

The default custom icon of this hybrid app. Null if this hybrid app does not have a custom icon.

Type:

hwc.CustomIcon on page 172

Source

hwc-api.js, line 1422 on page 347.

this.getDisplayName() method

Gets the display name for this hybrid app.

Syntax

```
<static> this.getDisplayName() {string}
```

Returns

The display name.

Type:

string

Source

hwc-api.js, line 1400 on page 346.

this.getIconIndex() method

Gets the icon index used in the list of built-in icons.

Syntax

```
<static> this.getIconIndex() {number}
```

Returns

The icon index

Type:

number

Source

hwc-api.js, line 1411 on page 346.

this.getModuleID() method

Gets the module ID for this hybrid app.

Syntax

```
<static> this.getModuleID() {number}
```

Returns

The module ID.

Type:

number

Source

hwc-api.js, line 1378 on page 345.

this.getVersion() method

Gets the version number for this hybrid app.

Syntax

```
<static> this.getVersion() {number}
```

Returns

The version.

Type:

number

Source

hwc-api.js, line 1389 on page 346.

isAndroid() method

Platform

Syntax

```
<static> isAndroid() {boolean}
```

Returns

True if the hybrid app application is being run on an Android platform.

Type:

boolean

Source

PlatformIdentification.js, line 149 on page 506.

isAndroid3() method

Platform

Syntax

```
<static> isAndroid3() {boolean}
```

Returns

True if the hybrid app application is being run on an Android 3.0 OS

Type:

boolean

Source

PlatformIdentification.js, line 141 on page 506.

isBlackBerry() method

Platform

Syntax

```
<static> isBlackBerry() {boolean}
```

Returns

True if the hybrid app application is being run on a BlackBerry platform.

Type:

boolean

Source

PlatformIdentification.js, line 83 on page 503.

isBlackBerry5() method

Platform

Syntax

```
<static> isBlackBerry5() {boolean}
```

Returns

True if the hybrid app application is being run on a BlackBerry 5.0 OS

Type:

boolean

Source

PlatformIdentification.js, line 91 on page 504.

isBlackBerry5WithTouchScreen() method

Platform

Syntax

```
<static> isBlackBerry5WithTouchScreen() {boolean}
```

Returns

True if the hybrid app application is being run on a BlackBerry 5.0 OS with a touch screen

Type:

boolean

Source

PlatformIdentification.js, line 107 on page 504.

isBlackBerry6NonTouchScreen() method

Platform

Syntax

```
<static> isBlackBerry6NonTouchScreen() {boolean}
```

Returns

True if the hybrid app application is being run on a BlackBerry 6.0 OS without a touch screen

Type:

boolean

Source

PlatformIdentification.js, line 115 on page 505.

isBlackBerry7() method

Platform

Syntax

```
<static> isBlackBerry7() {boolean}
```

Returns

True if the hybrid app application is being run on a BlackBerry 7.x OS

Type:

boolean

Source

PlatformIdentification.js, line 99 on page 504.

isClosed() method

This function checks if the hybrid app has been closed.

Syntax

```
<static> isClosed() {boolean}
```

Returns

true if hybrid app is closed, otherwise false.

Type:

boolean

Example

```
hwc.isClosed();
```

Source

hwc-comms.js, line 1602 on page 489.

isDstActiveAtGivenTime(date) method

Timezone

Syntax

```
<static> isDstActiveAtGivenTime( date ) {boolean}
```

Parameters

Name	Type	Description
<i>date</i>	Date	Date at which to determine whether daylight savings is in effect.

Returns

Returns true iff daylight savings rules are in effect at the given time in the current timezone.

Type:

boolean

Example

```
var isAwareAtTime = hwc.isDstActiveAtGivenTime(date);
```

Source

Timezone.js, line 356 on page 586.

isIOS() method

Platform

Syntax

```
<static> isIOS() {boolean}
```

Returns

True if the hybrid app application is being run on an iOS (e.g. iPhone, iPad) platform.

Type:

boolean

Source

PlatformIdentification.js, line 34 on page 502.

isIOS4() method

Returns true if the hybrid app application is being run on iOS4

Syntax

```
<static> isIOS4() {Boolean}
```

Returns

True if the hybrid app application is being run on iOS4

Type:

Boolean

Source

PlatformIdentification.js, line 74 on page 503.

isIOS5() method

Platform

Syntax

```
<static> isIOS5() {boolean}
```

Returns

True if the hybrid app application is being run on iOS5

Type:

boolean

Source

PlatformIdentification.js, line 50 on page 502.

isIOS6() method

Returns true if the hybrid app application is being run on iOS6

Syntax

```
<static> isIOS6() {boolean}
```

Returns

True if the hybrid app application is being run on iOS6

Type:

boolean

Source

PlatformIdentification.js, line 57 on page 502.

isIOS7() method

Returns true if the hybrid app application is being run on iOS7

Syntax

```
<static> isIOS7() {boolean}
```

Returns

True if the hybrid app application is being run on iOS7

Type:

boolean

Source

PlatformIdentification.js, line 65 on page 503.

isIPad() method

Platform

Syntax

```
<static> isIPad() {boolean}
```

Returns

True if the hybrid app application is being run on an iPad.

Type:

boolean

Source

PlatformIdentification.js, line 42 on page 502.

isSharedStorageEnabled() method

Storage

Syntax

```
<static> isSharedStorageEnabled() {boolean}
```

Returns

true if the shared storage is enabled; false otherwise.

Type:

boolean

Source

SUPStorage.js, line 362 on page 572.

isWindows() method

Platform

Syntax

```
<static> isWindows() {boolean}
```

Returns

True if the hybrid app application is being run on a Windows platform.

Type:

boolean

Source

PlatformIdentification.js, line 132 on page 505.

isWindowsMobile() method

Platform

Syntax

```
<static> isWindowsMobile() {boolean}
```

Returns

True if the hybrid app application is being run on a Windows Mobile platform.

Type:

boolean

Source

PlatformIdentification.js, line 124 on page 505.

loadSettings() method

Loads the current connection settings from the native application storage.

Syntax

```
<static> loadSettings() {hwc.ConnectionSettings}
```

Returns

The connection settings or null if there are no cached settings.

Type:

hwc.ConnectionSettings on page 168

Example

```
// Load the connection settings.
var connectionSettings = hwc.loadSettings();
```

Source

hwc-api.js, line 101 on page 297.

log(sMsg, eLevel, notifyUser) method

Allows the user to log a message to the device trace log which can be remotely retrieved from the server.

Whether the message actually gets logged will depend on how the log level that the administrator has selected for this device user compares with the log level of this message. The logging level and alert dialog callback can be set with *hwc.setLoggingCurrentLevel* on page 239 and *setLoggingAlertDialog*.

Syntax

<static> log(*sMsg*, *eLevel*, *notifyUser*)

Parameters

Name	Type	Description
<i>sMsg</i>	string	The message to be logged.
<i>eLevel</i>	string	The error level for this message. This parameter must be one of: "ERROR", "WARN", "INFO" or "DEBUG".
<i>notifyUser</i>	boolean	Whether the logging alert callback will be invoked. This parameter is independent of the logging level (the logging alert callback will always be invoked if this is true, and never if this is false).

Example

```

var logAlert = function( message )
{
    alert( "New log message: " + message );
}
hwc.setLoggingAlertDialog( logAlert );
hwc.setLoggingCurrentLevel( 3 );
// The following will be logged, and the logging alert dialog will be
invoked.
hwc.log( "info message notify", "INFO", true );
// The following will be logged, but the logging alert dialog will
not be invoked.
hwc.log( "info message", "INFO", false );
// The following will not be logged, but the logging alert dialog
will be invoked.
hwc.log( "debug message notify", "DEBUG", true );
// The following will not be logged, and the logging alert dialog
will not be invoked.
hwc.log( "debug message", "DEBUG", false );

```

Source

hwc-comms.js, line 902 on page 464.

LogEntry(date, event, msg) method

This object represents a log entry.

Syntax

<static> LogEntry(*date, event, msg*)

Parameters

Name	Type	Description
<i>date</i>	number	The date the log entry was recorded, in milliseconds since January 1, 1970, 00:00:00 GMT

<i>event</i>	number	The event ID of the log entry (will be one of <i>hwc.CONNECTION_ERROR</i> on page 128, <i>hwc.CONNECTION_OTHER</i> on page 129, <i>hwc.CONNECTION_CONNECTED</i> on page 128, <i>hwc.CONNECTION_DISCONNECTED</i> on page 128, <i>hwc.CONNECTION_RETRIEVED_ITEMS</i> on page 129)
<i>msg</i>	string	The message of the log entry.

Source

hwc-api.js, line 1084 on page 334.

this.getDate() method

Gets the date of the log entry.

Syntax

```
<static> this.getDate() {number}
```

Returns

The date the log entry was created in the HWC, in milliseconds.

Type:

number

Source

hwc-api.js, line 1096 on page 335.

this.getEventID() method

Gets the event ID of the log entry to see what this log entry is about.

Syntax

```
<static> this.getEventID() {number}
```

Returns

A constant indication what this log entry is about (will be one of *hwc.CONNECTION_ERROR* on page 128, *hwc.CONNECTION_OTHER* on page 129, *hwc.CONNECTION_CONNECTED* on page 128,

hwc.CONNECTION_DISCONNECTED on page 128,
hwc.CONNECTION_RETRIEVED_ITEMS on page 129).

Type:

number

Source

hwc-api.js, line 1108 on page 335.

this.getMessage() method

Gets the message text of the log entry.

Syntax

```
<static> this.getMessage() {string}
```

Returns

The message text of the log entry.

Type:

string

Source

hwc-api.js, line 1119 on page 335.

markAsActivated() method

This function sets the activation required state for the current hybrid app to false.

After calling this function, the current hybrid app will not need to be activated.

Syntax

```
<static> markAsActivated()
```

Example

```
hwc.markAsActivated();
```

Source

hwc-comms.js, line 790 on page 460.

markAsProcessed() method

Allows the user to set the processed state to true for the current message.

Syntax

```
<static> markAsProcessed()
```

Example

```
hwc.markAsProcessed()
```

Source

hwc-comms.js, line 806 on page 460.

MenuItemCollection() method

This class represents a collection of menu items.

Syntax

```
<static> MenuItemCollection()
```

Example

```
// This is the function we'll use as a callback for the first menu
item.
var callback = function()
{
    alert( "You clicked the first menu item!" );
}

// This is the function we'll use as a callback for the second menu
item.
var callback2 = function()
{
    alert( "You clicked the second menu item!" );
}

// This function creates and adds a menu item collection.
var addMenuItems = function()
{
    var menuItemCollection = new hwc.MenuItemCollection();
    menuItemCollection.addItem("menu item 1", "callback()");
    menuItemCollection.addItem("menu item 2", "callback2()");
    hwc.addItemCollection( menuItemCollection );
}
```

Source

hwc-comms.js, line 554 on page 451.

addItem(title, callback, [isDefault]) method

This function adds a menu item to the collection.

Syntax

```
addItem( title, callback, [isDefault] )
```

Parameters

Name	Type	Argument	Description
<i>title</i>	string		The display text for the menu item.
<i>callback</i>	<i>anonymous.genericCallbackFunction</i> on page 78		The function to call when the menu item is clicked.
<i>isDefault</i>	boolean	(optional)	Determines if the menu item is selected by default on BlackBerry. If more than one menu item is added to the same collection with true for this parameter, the last menu item added with true for this parameter will be selected by default on BlackBerry.

Example

```
var callbackFunctionName = function()
{
    alert( "Menu item clicked!" );
}
var menuItemCollection = new hwc.MenuItemCollection();
menuItemCollection.addMenuItem("menu item name",
"callbackFunctionName()", true);
```

Source

hwc-comms.js, line 579 on page 452.

setOKAction(callback) method

This function sets the OK action to use on WM.

Syntax

```
setOKAction( callback )
```

Parameters

Name	Type	Description
------	------	-------------

<i>callback</i>	<i>anonymous.genericCallback-Function</i> on page 78	The function to call when the OK button is pressed.
-----------------	--	---

Example

```
var callbackFunctionName = function()
{
    alert( "Menu item clicked!" );
}
var okActionFunction = function()
{
    alert( "A OKAY!" );
}
var menuItemCollection = new hwc.MenuItemCollection();
menuItemCollection.setOKAction( "okActionFunction()" );
menuItemCollection.addItem("menu item name",
"callbackFunctionName()");
```

Source

hwc-comms.js, line 626 on page 454.

setSubMenuName(name) method

This function sets the sub menu name to use on Windows Mobile.

Syntax

setSubMenuName(*name*)

Parameters

Name	Type	Description
<i>name</i>	string	The sub menu name to use.

Example

```
var callbackFunctionName = function()
{
    alert( "Menu item clicked!" );
}
var menuItemCollection = new hwc.MenuItemCollection();
menuItemCollection.setSubMenuName( "Custom Menu" );
menuItemCollection.addItem("menu item name",
"callbackFunctionName()");
```

Source

hwc-comms.js, line 603 on page 453.

stringify() method

This function converts the menu item collection to a JSON string.

This function is used as a helper for *hwc.addItemCollection* on page 154.

Syntax

```
stringify() {string}
```

Returns

The JSON string representing this menu item collection.

Type:

string

Example

```
var callbackFunctionName = function()
{
    alert( "Menu item clicked!" );
}
var menuItemCollection = new hwc.MenuItemCollection();
var jsonMenuItemCollection = menuItemCollection.stringify();
```

Source

hwc-comms.js, line 645 on page 455.

Message(msgId, date, icon, sender, isRead, processed, priority, subject, module, version) method

Represents a message received by the HWC.

Syntax

```
<static> Message( msgId, date, icon, sender, isRead, processed, priority, subject, module, version )
```

Parameters

Name	Type	Description
<i>msgId</i>	number	The message ID of this message.
<i>date</i>	Date	The date this message was received.
<i>icon</i>	number	The icon index for this message.
<i>sender</i>	string	The sender of this message.

<i>isRead</i>	boolean	Whether this message has been read or not.
<i>processed</i>	boolean	Whether this message has been processed or not.
<i>priority</i>	number	The priority of this message (must be either <i>hwc.MSG_PRIORITY_HIGH</i> on page 135 or <i>hwc.MSG_PRIORITY_NORMAL</i> on page 136).
<i>subject</i>	string	The subject of this message.
<i>module</i>	number	The module ID of the hybrid app associated with this message.
<i>version</i>	number	The version of the hybrid app associated with this message.

Source

hwc-api.js, line 2441 on page 383.

this.getIconIndex() method

Gets the icon index of this message.

Syntax

```
<static> this.getIconIndex() {number}
```

Returns

The icon index of this message.

Type:

number

Source

hwc-api.js, line 2482 on page 385.

this.getMessageId() method

Gets the message ID of this message.

Syntax

```
<static> this.getMessageId() {number}
```

Returns

The message ID of this message.\

Type:

number

Source

hwc-api.js, line 2460 on page 384.

this.getModuleId() method

Gets the module ID of the hybrid app this message belongs to.

Syntax

```
<static> this.getModuleId() {number}
```

Returns

The module ID of the hybrid app this message belongs to.

Type:

number

Source

hwc-api.js, line 2526 on page 386.

this.getModuleVersion() method

Gets the version of the hybrid app this message belongs to.

Syntax

```
<static> this.getModuleVersion() {number}
```

Returns

The version of the hybrid app this message belongs to.

Type:

number

Source

hwc-api.js, line 2537 on page 387.

this.getPriority() method

Gets the priority of the message.

Syntax

```
<static> this.getPriority() {number}
```

Returns

A constant indicating the priority of the message. Will be either *hwc.MSG_PRIORITY_NORMAL* on page 136 or *hwc.MSG_PRIORITY_HIGH* on page 135.

Type:

number

Source

hwc-api.js, line 2563 on page 388.

this.getReceivedDate() method

Gets the date this message was received.

Syntax

```
<static> this.getReceivedDate() {Date}
```

Returns

The date this message was received.

Type:

Date

Source

hwc-api.js, line 2471 on page 384.

this.getSender() method

Gets the sender of this message.

Syntax

```
<static> this.getSender() {string}
```

Returns

The sender of this message.

Type:

string

Source

hwc-api.js, line 2493 on page 385.

this.getSubject() method

Gets the subject of this message.

Syntax

```
<static> this.getSubject() {string}
```

Returns

The subject of this message.

Type:

string

Source

hwc-api.js, line 2515 on page 386.

this.isProcessed() method

Gets whether this message has been processed or not.

A message is generally marked as processed once the user submits changes from the hybrid app that was launched from the message.

Syntax

```
<static> this.isProcessed() {boolean}
```

Returns

True if this message has been processed, false otherwise.

Type:

boolean

Source

hwc-api.js, line 2550 on page 387.

this.isRead() method

Gets whether this message has been read or not.

Syntax

```
<static> this.isRead() {boolean}
```

Returns

Whether this message has been read (true) or not (false).

Type:

boolean

Source

hwc-api.js, line 2504 on page 386.

this.updateProcessed(status) method

Updates the processed status of the message.

Syntax

```
<static> this.updateProcessed( status )
```

Parameters

Name	Type	Description
<i>status</i>	boolean	The new processed status.

Source

hwc-api.js, line 2592 on page 388.

this.updateRead(status) method

Updates the read status of the message.

Syntax

```
<static> this.updateRead( status )
```

Parameters

Name	Type	Description
<i>status</i>	boolean	The new read status.

Source

hwc-api.js, line 2580 on page 388.

MessageFilter([sender], [subject], [moduleId], [version], [isread], [processed]) method

Represents a filter used to filter messages.

Pass in null for any parameter you do not wish to filter (or do not pass in such parameters at all).

Syntax

```
<static> MessageFilter( [sender], [subject], [moduleId], [version], [isread], [processed] )
```

Parameters

Name	Type	Argument	Description
<i>sender</i>	string	(optional)	The sender of the message.
<i>subject</i>	string	(optional)	The subject of the message.
<i>moduleId</i>	number	(optional)	The associated application module ID.
<i>version</i>	number	(optional)	The associated application module versions.
<i>isread</i>	boolean	(optional)	The read status.
<i>processed</i>	boolean	(optional)	The processed status.

Source

hwc-api.js, line 2614 on page 389.

openApp(moduleId, version) method

Launch the hybrid app with the given module ID and version.

The hybrid app will be opened on top of the hybrid app that is open when `hwc.openApp` is called. When the hybrid app that was opened with `hwc.openApp` exits, it will exit to the hybrid app that was open when `hwc.openApp` was called. It is possible to nest open hybrid apps, but it is best not to have too many nested hybrid apps (eg: recursively opening hybrid apps) because each open hybrid app takes up device memory.

Syntax

```
<static> openApp( moduleId, version ) {number}
```

Parameters

Name	Type	Description
<i>moduleId</i>	number	Module id of the hybrid app.
<i>version</i>	number	Version of the hybrid app.

Returns

A constant indicating the result of opening the hybrid app (will be one of *hwc.OPEN_APP_SUCCESS* on page 138, *hwc.OPEN_APP_NOT_EXIST* on page 138, *hwc.OPEN_APP_OTHER* on page 138).

Type:

number

Example

```
var apps = hwc.getInstalledApps();
if( apps.length > 0 )
{
    // Check to make sure the first app is not this app (the app that
    // is currently running),
    // since we don't want to recursively open this app until memory
    // runs out.
    if( hwc.getCurrentHybridApp.getDisplayName() !=
    apps[0].getDisplayName() )
    {
        hwc.openApp( apps[0].getModuleID(), apps[0].getVersion() );
    }
}
```

Source

hwc-api.js, line 1883 on page 364.

openMessage(msgId) method

Launch the server initiated hybrid app associated with a message.

The hybrid app will be opened on top of the hybrid app that is open when *hwc.openMessage* is called. When the hybrid app that was opened with *hwc.openMessage* exits, it will exit to the hybrid app that was open when *hwc.openMessage* was called. It is possible to nest open hybrid apps, but it is best not to have too many nested hybrid apps (eg: recursively opening hybrid apps) because each open hybrid app takes up device memory.

Syntax

```
<static> openMessage( msgId ) {number}
```


Parameters

Name	Type	Description
<i>msgId</i>	number	The id of message to open.

Returns

A number indicating the success or failure of opening the message (will be one of *hwc.OPEN_MSG_SUCCESS* on page 140, *hwc.OPEN_MSG_NOT_EXIST* on page 139, *hwc.OPEN_MSG_APP_NOT_EXIST* on page 139, *hwc.OPEN_MSG_OTHER* on page 139).

Type:

number

Example

```
// get all messages, then open the first one
var messages = hwc.getAllMessages();
if( messages.length > 0 )
{
    hwc.openMessage( messages[0].getMessageId() );
}
```

Source

hwc-api.js, line 3185 on page 410.

removeAllMenuItems() method

This function removes all menu items that were added by the hybrid app.

Note: This API does not support on iOS platform.

Syntax

```
<static> removeAllMenuItems()
```

Example

```
hwc.removeAllMenuItems();
```

Source

hwc-comms.js, line 754 on page 458.

removeAppInstallationListener(AppInstallationListener) method

Remove the application installation listener.

This function should be called with identical parameters that were used to add the application installation listener with *hwc.addAppInstallationListener* on page 149.

Syntax

```
<static> removeAppInstallationListener( AppInstallationListener )
```

Parameters

Name	Type	Description
<i>AppInstallationListener</i>	<i>anonymous.AppInstallation-Listener</i> on page 73	The callback for application installation changes.

Example

```
// appInstallListener is the callback function that will be passed to  
hwc.addAppInstallationListener.  
var appInstallListener = function( event, moduleId, version,  
moduleName )  
{  
  if( event == hwc.INSTALLATION_BEGIN )  
  {  
    alert(moduleName + " has just started the installation  
process.");  
  }  
  else if( event == hwc.INSTALLATION_END )  
  {  
    alert(moduleName + " has just finished the installation  
process.");  
  }  
}  
hwc.addAppInstallationListener( appInstallListener );  
// when we want to remove this listener, we call the following line:  
hwc.removeAppInstallationListener( appInstallListener );
```

Source

hwc-api.js, line 984 on page 330.

removeAppListener(ApplicationListener, [containingObject]) method

Remove the application listener.

This function should be called with identical parameters that were used to add the application listener with *hwc.addAppListener* on page 150.

Syntax

```
<static> removeAppListener( ApplicationListener, [containingObject] )
```

Parameters

Name	Type	Argument	Description
------	------	----------	-------------

<i>ApplicationListener</i>	<i>anonymous.ApplicationListener</i> on page 75		The callback for application changes.
<i>containingObject</i>	Object	(optional)	The containing object of the application listener function.

Example

```
// This is the callback function that will be passed to
hwc.addAppListener.
var appListener = function( event, moduleId, version )
{
    if( event == hwc.APP_ADDED )
    {
        alert("A hybrid app has been added.");
    }
}
hwc.addAppListener( appListener );
// At some other point, if we want to remove the listener we use the
following line of code:
hwc.removeAppListener( appListener );

// appListenerManager is an object that will contain the callback
function as well as variables
// the callback function references.
var appListenerManager = {};
// doSomething is a function that is called from inside the callback
function.
appListenerManager.doSomething = function( event )
{
    if( event == hwc.APP_REMOVED )
    {
        alert("A hybrid app has been removed.");
    }
}
// This is the callback function that will be passed to
hwc.addAppListener. It calls doSomething,
// the definition of which is in the containing function.
appListenerManager.listener = function( event, moduleId, version )
{
    this.doSomething( event );
}
// Since the listener callback function references a variable from
its containing object,
// the containing object must be passed to hwc.addAppListener.
hwc.addAppListener( appListenerManager.listener,
appListenerManager );
// At some other point, if we want to remove the listener we use the
following line of code:
hwc.removeAppListener( appListenerManager.listener,
appListenerManager );
```

Source

hwc-api.js, line 1601 on page 353.

removeConnectionListener(ConnectionStateListener, [containingObject])
method

Remove the connection state listener.

This function should be called with identical parameters that were used when adding the connection state listener with *hwc.addConnectionListener* on page 151.

Syntax

```
<static> removeConnectionListener( ConnectionStateListener,  
[containingObject] )
```

Parameters

Name	Type	Argument	Description
<i>ConnectionStateListener</i>	<i>anonymous.ConnectionStateListener</i> on page 77		Callback function with connection state changes
<i>containingObject</i>	Object	(optional)	Optional Object containing definition of <i>ConnectionStateListener</i>

Example

```
// doSomething is a global function that gets called from the  
connection listener.  
var doSomething = function()  
{  
    alert("sample function that gets executed when the hwc becomes  
connected");  
}  
// connectionListener is the callback function that is given to  
addConnectionListener.  
// When there is a connection event, connectionListener will be  
invoked with the details.  
var connectionListener = function( event, errorCode, errorMessage )  
{  
    if( event == hwc.CONNECTED )  
    {  
        doSomething();  
    }  
}  
hwc.addConnectionListener( connectionListener );  
// At some other point if we want to remove the listener, we use the
```

```

following line:
hwc.removeConnectionListener( connectionListener );

// connectionStateManager is an object that will contain the
// connection listener callback as well as
// a variable used by the callback.
var connectionStateManager = {};
// The connectionStateManager keeps track of whether the HWC is
// connected or not.
connectionStateManager.connected = false;
// A function called by the listener.
connectionStateManager.doSomething = function()
{
    if( this.connected )
    {
        alert("this alert gets displayed if the hwc is connected");
    }
}
// This is the callback function that will be passed to
// addConnectionListener. This callback references variables
// from the containing object (this.connected and this.doSomething),
// so when we call addConnectionListener we have
// to give the containing object as the second parameter.
connectionStateManager.listener = function( event, errorCode,
errorMessage )
{
    if( event == hwc.CONNECTED )
    {
        this.connected = true;
    }
    else
    {
        this.connected = false;
    }
    this.doSomething();
}
// Pass both the listener and the containing object. This enables
// the listener to refer to variables in the containing object when it
// is invoked.
hwc.addConnectionListener( connectionStateManager.listener,
connectionStateManager );
// At some other point if we want to remove the listener, we use the
// following line:
hwc.removeConnectionListener( connectionStateManager.listener,
connectionStateManager );

```

Source

hwc-api.js, line 445 on page 311.

removeLogListener(LogListener, [containingObject]) method

Remove the log listener.

This function should be called with identical parameters that were used when adding the log listener with *hwc.addLogListener* on page 153.

Syntax

```
<static> removeLogListener( LogListener, [containingObject] )
```

Parameters

Name	Type	Argument	Description
<i>LogListener</i>	<i>anonymous.LogListener</i> on page 79		The callback function for log events.
<i>containingObject</i>	Object	(optional)	Object containing definition of Connection-StateListener

Example

```
// A global function called by the log listener.
var doSomething = function()
{
    alert("this gets displays when there is a log event.");
}
// The log listener callback function that will be passed to
hwc.addLogListener.
// This function will be invoked whenever there is a log event.
var logListener = function( event, errorCode, errorMessage )
{
    doSomething();
}
// Add the log listener.
hwc.addLogListener( logListener );
// at some other point if we want to remove the listener, we use the
following line
hwc.removeLogListener( logListener );

// logListenerManager is an object that will contain the listener
callback as well
// as a function that will be invoked from the listener callback
function.
var logListenerManager = {};
// This is a function that is called from the listener callback.
logListenerManager.doSomething = function()
{
    alert("this gets displays when there is a log event.");
}
// This is the listener callback that will be passed to
hwc.addLogListener.
// Since a variable is referenced from the containing object, the
containing object
// will need to be passed to hwc.addLogListener.
logListenerManager.listener = function( event, errorCode,
errorMessage )
{
    this.doSomething();
}
```

```

}
// Pass both the listener callback and the containing object.
hwc.addLogListener( logListenerManager.listener,
logListenerManager );
// at some other point if we want to remove the listener, we use the
following line
hwc.removeLogListener( logListenerManager.listener,
logListenerManager );

```

Source

hwc-api.js, line 818 on page 324.

removeMessage(msgId) method

Removes (deletes) a message.

Syntax

```
<static> removeMessage( msgId )
```

Parameters

Name	Type	Description
<i>msgId</i>	number	The id of the message to be removed.

Example

```

// remove all messages
var messages = hwc.getAllMessages();
for( var index = 0; index < messages.length; index++ )
{
    hwc.removeMessage( messages[index].getMessageId() );
}

```

Source

hwc-api.js, line 3132 on page 408.

removeMessageListener(MessageListener, [containingObject]) method

Removes the message listener.

The two parameters passed in to this function should match exactly the corresponding parameters passed into *hwc.addMessageListener* on page 155 when the message listener was added.

Syntax

```
<static> removeMessageListener( MessageListener, [containingObject] )
```

Parameters

Name	Type	Argument	Description
<i>MessageListener</i>	<i>anonymous.Message-Listener</i> on page 80		The callback for message changes.
<i>containingObject</i>	Object	(optional)	If the containing object was given to <i>hwc.addMessageListener</i> on page 155 when the message listener was added, then it also must be passed into this function.

Example

```
// soSomething is a global function called by the listener callback.
var doSomething = function()
{
    alert("New message!");
}
// messageListener is the callback function passed to
hwc.addMessageListener.
var messageListener = function( flag, messageId )
{
    if( flag == hwc.MSG_ADDED )
    {
        doSomething();
    }
}
// We do not want to filter the message events the listener will get
invoked for, so pass null for the first parameter.
hwc.addMessageListener( null, messageListener );
// If we want to remove the listener at some other point, use the
following line of code:
hwc.removeMessageListener( messageListener );

// someObject is an object that will contain the listener callback as
well as a variable referenced by the callback.
var someObject = {};
// doSomething is a function referenced by the callback function.
someObject.doSomething = function()
{
    alert("New message!");
}
// messageListener is the callback that will be passed to
hwc.addMessageListener.
someObject.messageListener = function( flag, messageId )
{
    if( flag == hwc.MSG_ADDED )
```



```

    {
      this.doSomething();
    }
  }
  // Create a filter so that not all message events will invoke our
  // callback function.
  // Only events about messages with a subject of "SI<4>" will trigger
  // our callback function.
  var filter = new hwc.MessageFilter( null, "SI<4>", null, null, null,
  null);
  // The callback function references a variable in its containing
  // object, so we need to pass in the containing object
  // in addition to the filter and the callback function.
  hwc.addMessageListener( filter, someObject.messageListener,
  someObject );
  // If we want to remove the listener at some other point, use the
  // following line of code:
  hwc.removeMessageListener( messageListener, someObject );

```

Source

hwc-api.js, line 2850 on page 398.

removePushNotificationListener(PushNotificationListener, [containingObject]) method

Remove the push notification listener.

This function should be called with identical parameters that were used to add the push notification listener with *hwc.addPushNotificationListener* on page 157.

Syntax

<static> removePushNotificationListener(*PushNotificationListener*, [containingObject])

Parameters

Name	Type	Argument	Description
<i>PushNotificationListener</i>	anonymous.PushNotificationListener		The callback for push notifications.
<i>containingObject</i>	Object	(optional)	The containing object of the listener.

Example

```

// pushListener is the callback function that will be passed to
hwc.addPushNotificationListener.
var pushListener = function( notifications )
{
  alert( "push notification:\n"
+ JSON.stringify(notifications) );
  return hwc.NOTIFICATION_CONTINUE;
}

```

```
}
hwc.addPushNotificationListener( pushListener );
// At some other point if we want to remove the push listener, we call
the following line:
hwc.removePushNotificationListener( pushListener );

// pushListenerManager is an object that will contain the listener
callback as well as a variable
// referenced from the callback.
var pushListenerManager = {};
// doSomething is a function that is called from inside the callback.
pushListenerManager.doSomething = function( notifications )
{
    alert( "push notification:\
" + JSON.stringify(notifications) );
    return hwc.NOTIFICATION_CONTINUE;
}
// This is the callback function.
pushListenerManager.listener = function( notifications )
{
    return this.doSomething( notifications );
}
// Since the callback function references variables in its containing
object, the containing object
// must be passed to hwc.addPushNotificationListener as well.
hwc.addPushNotificationListener( pushListenerManager.listener,
pushListenerManager );
// when we want to remove the push listener, we call the following
line:
hwc.removePushNotificationListener( pushListener,
pushListenerManager );
```

Source

hwc-api.js, line 1299 on page 342.

sample_AppListener(event, moduleId, version) method

A sample *anonymous.ApplicationListener* on page 75 callback function.

Syntax

<static> sample_AppListener(*event*, *moduleId*, *version*)

Parameters

Name	Type	Description
------	------	-------------

<i>event</i>	number	A number indicating what event has taken place (will be one of <i>hwc.APP_REFRESH</i> on page 126, <i>hwc.APP_ADDED</i> on page 126, <i>hwc.APP_UPDATED</i> on page 127, <i>hwc.APP_REMOVED</i> on page 127).
<i>moduleId</i>	number	The module id of the hybrid app the event is about.
<i>version</i>	number	module The version of the hybrid app the event is about.

Source

hwc-api.js, line 1662 on page 355.

sample_ConnectionListener(event, errorCode, errorMessage) method

A sample *anonymous.ConnectionStateListener* on page 77 callback function.

Syntax

```
<static> sample_ConnectionListener( event, errorCode, errorMessage )
```

Parameters

Name	Type	Description
<i>event</i>	number	A number indicating the event that occurred (will be <i>hwc.CONNECTED</i> on page 127 or <i>hwc.DISCONNECTED</i> on page 130).
<i>errorCode</i>	number	An error code (0 indicating success).
<i>errorMessage</i>	string	Text of the error message. Will be empty if there is no error.

Source

hwc-api.js, line 480 on page 312.

sample_InstallationAppListener(event, moduleId, version, moduleName, designerVersion, containerVersion) method

Sample application listener callback function

Syntax

```
<static> sample_InstallationAppListener( event, moduleId, version,  
moduleName, designerVersion, containerVersion )
```

Parameters

Name	Type	Description
<i>event</i>	Integer	Installation flags including, BEGIN(1), END(2), FAIL(3)
<i>moduleId</i>	String	Optional Module Id
<i>version</i>	String	Optional Module version
<i>moduleName</i>	String	Optional Module display name
<i>designerVersion</i>	String	Optional Version of designer used to create app
<i>containerVersion</i>	String	Optional Version of hybrid web container

Source

hwc-api.js, line 1021 on page 332.

sample_LogListener(milliseconds, event, optionalString) method

Sample *anonymous.LogListener* on page 79 callback function.

Syntax

```
<static> sample_LogListener( milliseconds, event, optionalString )
```

Parameters

Name	Type	Description
<i>milliseconds</i>	number	The date of the log message represented in milliseconds.

<i>event</i>	number	The that represents which category this event falls under (It will be one of <i>hwc.CONNECTION_ERROR</i> on page 128, <i>hwc.CONNECTION_OTHER</i> on page 129, <i>hwc.CONNECTION_CONNECTED</i> on page 128, <i>hwc.CONNECTION_DISCONNECTED</i> on page 128, <i>hwc.CONNECTION_RETRIEVED_ITEMS</i> on page 129).
<i>optionalString</i>	string	The string carrying the message of the log event.

Source

hwc-api.js, line 855 on page 326.

sample_MessageListener(flag, msgId) method

A sample *anonymous.MessageListener* on page 80 callback function.

Syntax

```
<static> sample_MessageListener( flag, msgId )
```

Parameters

Name	Type	Description
<i>flag</i>	number	A number indicating which message event occurred (will be one of MSG_* constants).
<i>msgId</i>	number	The message id of the affected message.

Source

hwc-api.js, line 2916 on page 400.

sample_PushNotificationListener(notifications) method

A sample implementation of a *anonymous.PushNotificationListener* callback function.

Syntax

```
<static> sample_PushNotificationListener( notifications )
```

Parameters

Name	Type	Description
<i>notifications</i>	Array	Array of notifications.

Source

hwc-api.js, line 1345 on page 344.

saveLoginCertificate(certificate) method

This function saves login credentials from a certificate to the credential cache.

The common name is used for the username and the signed certificate is used for the password.

Syntax

<static> saveLoginCertificate(*certificate*)

Parameters

Name	Type	Description
<i>certificate</i>	object	The values certificate.subjectCN and certificate.signedCertificate must be defined.

Example

```
var certInfo = {};
certInfo.subjectCN = "sampleCommonName";
certInfo.signedCertificate = "samplePassword";
hwc.saveLoginCertificate( certInfo );
```

Source

hwc-comms.js, line 971 on page 466.

saveLoginCredentials(userName, password) method

This function saves login credentials to the credential cache.

Syntax

<static> saveLoginCredentials(*userName, password*)

Parameters

Name	Type	Description
<i>userName</i>	string	The user name to save

<i>password</i>	string	The password to save
-----------------	--------	----------------------

Example

```
hwc.saveLoginCredentials( "sampleUserName", "samplePassword" );
```

Source

hwc-comms.js, line 990 on page 467.

saveSettings(settings) method

Save the connection settings to native application storage.

Device registration will be attempted if and only the following conditions are both satisfied.

1. The registration method is not manual. This can be passed in the `hwc.ConnectionSettings` object, or if that value is null, the currently configured value will be used.
2. The password must be non-empty. This value **MUST** be passed in the `hwc.ConnectionSettings` object.

hwc.startClient() needs to be called after hwc.saveSettings() for the device to complete automatic/manual registration.

Usage Note: It is not mandatory to specify a value for each `hwc.ConnectionSettings` property. Specifying a null or undefined for a *hwc.ConnectionSettings* on page 168 property will effectively cause this method to IGNORE the property and not change it's value. If the `saveSettings()` operation fails, a non-zero number will be returned. See `hwc.REG_ERR_*` for device registration errors. There can be other types of errors not listed here.

Syntax

```
<static> saveSettings( settings ) {number}
```

Parameters

Name	Type	Description
<i>settings</i>	<i>hwc.ConnectionSettings</i> on page 168	The connection settings to be saved.

Returns

A status code indicating success (*hwc.SETTING_SUCCESS* on page 148) or an error (one of *hwc.REG_ERR_AUTO_REG_NOT_ENABLED* on page 144, *hwc.REG_ERR_AUTO_REG_TEMPLATE_NOT_FOUND* on page 145, *hwc.REG_ERR_AUTO_REG_USER_NAME_TOO_LONG* on page 145, *hwc.REG_ERR_AUTO_REG_WRONG_USER_FOR_DEVICE* on page 145, *hwc.REG_ERR_COULD_NOT_REACH_MMS_SERVER* on page 146,

hwc.REG_ERR_INVALID_USER_NAME on page 146,
hwc.REG_ERR_MMS_AUTHENTICATION_FAILED on page 146).

Type:

number

Example

```
// Load the connection settings.
var connectionSettings = hwc.loadSettings();
// Modify the connection settings.
connectionSettings.ServerName = "999.999.999.999";
// Save the modified connection settings.
hwc.saveSettings( connectionSettings );
// Start the client to for the device to complete automatic/manual
registration.
hwc.startClient();
```

Source

hwc-api.js, line 196 on page 301.

setLoggingAlertDialog(newAlertDialogCallback) method

This function sets the callback used by `hwc.log` when it is required to notify the user of a log item.

Syntax

```
<static> setLoggingAlertDialog( newAlertDialogCallback )
```

Parameters

Name	Type	Description
<i>newAlertDialogCallback</i>	<i>anonymous.alertDialogCall- backFunction</i> on page 73	The alert dialog to use.

Example

```
customLogAlert = function( message )
{
    alert( "New log message: " + message );
}
hwc.setLoggingAlertDialog( customLogAlert );
```

Source

hwc-comms.js, line 207 on page 439.

setLoggingCurrentLevel(newLoggingLevel) method

This function sets the logging level.

The logging level set with this function only persists as long as this javascript context does. When the hybrid app is closed, the value set with this function is lost.

Syntax

```
<static> setLoggingCurrentLevel( newLoggingLevel )
```

Parameters

Name	Type	Description
<i>newLoggingLevel</i>	number	The number representing the new logging level. Must be an integer in the range [1..4]. The higher numbers represent more verbose logging levels from 1 for ERROR level logging up to 4 for DEBUG level logging.

Example

```
// Set the logging level to debug.
hwc.setLoggingCurrentLevel( 4 );
```

Source

hwc-comms.js, line 236 on page 441.

setReportErrorFromNativeCallback(callbackToSet) method

This function sets the callback function called when there is a native error reported.

Calling this function will replace any callback that had been set previously.

Syntax

```
<static> setReportErrorFromNativeCallback( callbackToSet )
```

Parameters

Name	Type	Description
<i>callbackToSet</i>	function	The callback function.

Example

```
var errorCallback = function( errorString )
{
    alert( "There was a native error: " + errorString );
}
```

```
}  
hwc.setReportErrorFromNativeCallback( errorCallback );
```

Source

hwc-comms.js, line 274 on page 442.

setScreenTitle_CONT(screenTitle) method

Sets the title of the screen.

Syntax

```
<static> setScreenTitle_CONT( screenTitle )
```

Parameters

Name	Type	Description
<i>screenTitle</i>	string	The screen title to use.

Example

```
hwc.setScreenTitle_CONT( "Custom Screen Title" );
```

Source

hwc-comms.js, line 508 on page 450.

SharedStorage() method

Storage

Syntax

```
<static> SharedStorage()
```

Source

SUPStorage.js, line 379 on page 573.

showAlertDialog(message, [title]) method

Displays an alert dialog to the user.

This function blocks until it receives a response from the user.

Syntax

```
<static> showAlertDialog( message, [title] )
```

Parameters

Name	Type	Argument	Description
------	------	----------	-------------

<i>message</i>	string		The message to display
<i>title</i>	string	(optional)	The title doesn't actually get displayed.

Example

```
hwc.showAlertDialog( "This is a fancy alert dialog", "With a Title" );
```

Source

hwc-comms.js, line 1522 on page 486.

showAttachmentContents_CONT(contents, mimeType, fileName, waitDialogCallbackString) method

Shows the given file contents in a content-appropriate way.

The type of the content is supplied by either the MIME type or the filename, at least one of which must be supplied. The content itself should be presented as a base64-encoded string. Not all file types may be supported on all platforms.

Syntax

```
<static> showAttachmentContents_CONT( contents, mimeType, fileName, waitDialogCallbackString )
```

Parameters

Name	Type	Description
<i>contents</i>	string	The base-64 encoded version of the binary content of the attachment to be displayed.
<i>mimeType</i>	string	The MIME type of the file.
<i>fileName</i>	string	The name of the file.
<i>waitDialogCallbackString</i>	<i>anonymous.genericCallback-Function</i> on page 78	The callback function used to close a wait dialog once the attachment is done opening.

Example

```
var openAttachmentBase64StringPng = function()
{
    // How you want get the base 64 encoding of the file is up to you.
    This string represents a small png image.
    var data =
    "iVBORw0KGgoAAAANSUheUgAAACAAAAAgCAYAAABzenr0AAAAAXNSR0IArs4c6QAAAA
    RnQU1BAACxjwv8YQUAAAAJcEhZcwAADsMAAA7DAcdvqGQAAAAOSURBVFH7dAxEQAAC
```

```
AMx3CAT6eVQwZKh8/dSmc7n6jN
+bQcIECBAGAABAGQIECBAGACBBb3SkJeQ67u1AAAAAE1FTkSuQmCC";
    hwc.showProgressDialog();
    // Don't have to pass the filename because we are passing the MIME
type.
    hwc.showAttachmentContents_CONT( data, "image/png", null,
"hwc.hideProgressDialog()" );
}

var openAttachmentBase64StringTxt = function()
{
    // How you want get the base 64 encoding of the file is up to you.
This string represents a short text file.
    var data = "VGhpcyBpcyBwYXJ0IG9mIGEgaHlicmlkIGFwcC4=";
    // Don't have to pass the MIME type because we are passing the
filename.
    hwc.showAttachmentContents_CONT( data, null, "attach.txt" );
}
```

Source

hwc-comms.js, line 1070 on page 470.

showAttachmentFromCache_CONT(uniqueKey, mimeType, fileName, waitDialogCallbackString) method

Shows the given file contents in a content-appropriate way.

The type of the content is supplied by either the MIME type or the filename, at least one of which must be supplied. The content itself will be a unique key supplied earlier to a call to `doAttachmentDownload`.

Syntax

`<static> showAttachmentFromCache_CONT(uniqueKey, mimeType, fileName, waitDialogCallbackString)`

Parameters

Name	Type	Description
<i>uniqueKey</i>	string	The unique key for the attachment.
<i>mimeType</i>	string	The MIME type of the file.
<i>fileName</i>	string	The name of the file.
<i>waitDialogCallbackString</i>	string	string with the value for the 'callback=' parameter.

Source

hwc-comms.js, line 1106 on page 471.

showCertificatePicker() method

This function opens a form on the device that allows the user to specify the credentials for the use of certificate-based authentication.

If the user picks a certificate, then that certificate is saved in the credentials cache.

Syntax

```
<static> showCertificatePicker()
```

Example

```
hwc.showCertificatePicker();
```

Source

hwc-comms.js, line 948 on page 465.

showConfirmDialog(message, [title]) method

Shows a confirm dialog to the user.

This function blocks until it receives a response from the user.

Syntax

```
<static> showConfirmDialog( message, [title] ) {boolean}
```

Parameters

Name	Type	Argument	Description
<i>message</i>	string		The message to display in the dialog.
<i>title</i>	string	(optional)	The title doesn't actually get displayed.

Returns

The user's choice from the confirm dialog.

Type:

boolean

Example

```
var userConfirm = hwc.showConfirmDialog( "Are you sure you want to
see an alert message?", "Confirm Alert" );
if( userConfirm )
{
    alert( "This is what you wanted." );
}
```

Source

hwc-comms.js, line 1553 on page 487.

showLocalAttachment(key) method

Shows a local attachment.

Syntax

<static> showLocalAttachment(*key*)

Parameters

Name	Type	Description
<i>key</i>	string	The key of the attachment. This is the path to the file, with the root being the folder that manifest.xml is located.

Example

```
hwc.showLocalAttachment ( "html/images/samplePic.gif" );
```

Source

hwc-comms.js, line 1136 on page 472.

showProgressDialog([message]) method

This function shows a progress dialog with spinner.

The dialog created by this function will block all user input until *hwc.hideProgressDialog* on page 197 is called. It is important to be sure that *hwc.hideProgressDialog* on page 197 will be called after a call to this function.

Syntax

<static> showProgressDialog([*message*])

Parameters

Name	Type	Argument	Description
------	------	----------	-------------

<i>message</i>	string	(optional)	The message to show on the progress dialog. This message is displayed on Android platforms only - other platforms show only a spinner.
----------------	--------	------------	--

Example

```
var showProgress = function()
{
    hwc.showProgressDialog( "a message" );
    setTimeout( hideProgress, 10000 );
}

var hideProgress = function()
{
    hwc.hideProgressDialog();
}
```

Source

hwc-comms.js, line 1475 on page 485.

showUrlInBrowser(url) method

This function opens the supplied URL in a browser.

The browser opens on top of the hybrid app - the context of the hybrid app is undisturbed.

Syntax

<static> showUrlInBrowser(*url*)

Parameters

Name	Type	Description
<i>url</i>	string	The URL to be shown in a browser.

Example

```
hwc.showUrlInBrowser( "http://www.google.com" );
```

Source

hwc-comms.js, line 1017 on page 468.

shutdown() method

Shutdown the client connection to the SUP server.

Companion function to *hwc.startClient* on page 246. If a hybrid app is running in the context of the Hybrid Web Container, then it will probably never have to call this function. If you want to temporarily stop the connection, then call *hwc.disconnectFromServer* on page 175 instead.

Syntax

```
<static> shutdown()
```

Example

```
hwc.shutdown();
```

Source

hwc-api.js, line 564 on page 315.

startClient([onNotification]) method

Start the client connection to the SUP server.

Companion function to *hwc.shutdown* on page 246. If a hybrid app is running in the context of the Hybrid Web Container then it will probably never have to call this function unless *hwc.shutdown* on page 246 client was called first.

Syntax

```
<static> startClient( [onNotification] )
```

Parameters

Name	Type	Argument	Description
<i>onNotification</i>	<i>anonymous.LogListener</i> on page 79	(optional)	A log listener callback function. If you are interested in the connection state it is recommended that you call <i>hwc.addConnectionListener</i> on page 151 before calling <i>hwc.startClient</i> .

Example

```
hwc.startClient();

// Add a log listener while calling hwc.startClient.
var logListener = function( time, event, message )
{
```



```

    alert(message);
}
hwc.startClient( logListener );

```

Source

hwc-api.js, line 531 on page 314.

this.getIconUrl(processed) method

Gets the URL of this custom icon.

It is possible to call this function directly, but generally it is easier simply to call *hwc.getAppIconUrl* on page 177 or *hwc.getMsgIconUrl* on page 189. Those functions handle both cases where there is and isn't a custom icon for the hybrid app or message.

Syntax

```
<static> this.getIconUrl( processed ) {string}
```

Parameters

Name	Type	Description
<i>processed</i>	boolean	When set to true, the URL of the processed icon will be returned. When set to false, the URL of the unprocessed icon will be returned.

Returns

The URL to the target icon.

Type:

string

Example

```

var apps = hwc.getInstalledApps();
var app = apps[0];
// If app doesn't have a custom icon, then customIcon will be null.
var customIcon = app.getDefaultCustomIcon();
if( customIcon != null )
{
    // Create the image element.
    var image = document.createElement( "img" );
    // Set the source of the image to the icon URL.
    image.setAttribute( 'src', customIcon.getIconUrl() );
    // Add the image element to the page.
    document.body.appendChild( image );
}

```

Source

hwc-api.js, line 2241 on page 376.

updateMessageProcessed(msgId, status) method

Updates the message processed status.

Syntax

<static> updateMessageProcessed(*msgId, status*)

Parameters

Name	Type	Description
<i>msgId</i>	number	The id of message to update the processed status for.
<i>status</i>	boolean	Whether the message will be set to processed (true) or unprocessed (false).

Example

```
// set all messages as processed
var messages = hwc.getAllMessages();
for( var index = 0; index < messages.length; index++ )
{
    hwc.updateMessageProcessed( messages[index].getMessageId(),
    true );
}
```

Source

hwc-api.js, line 3104 on page 407.

updateMessageRead(msgId, status) method

Updates the message read status.

Syntax

<static> updateMessageRead(*msgId, status*)

Parameters

Name	Type	Description
<i>msgId</i>	number	The id of message to update the read status for.

<i>status</i>	boolean	Whether the message will be set to read (true) or unread (false).
---------------	---------	---

Example

```
// set all messages as read
var messages = hwc.getAllMessages();
for( var index = 0; index < messages.length; index++ )
{
    hwc.updateMessageRead( messages[index].getMessageId(), true );
}
```

Source

hwc-api.js, line 3075 on page 406.

activationRequired() method

Deprecated: Deprecated since version 2.2 - use `hwc.activationRequired()`

Syntax

`activationRequired()`

Source

hwc-comms.js, line 77 on page 435.

clearCache() method

Deprecated: Deprecated since version 2.2 - use `hwc.clearCache()`

Syntax

`clearCache()`

Source

hwc-comms.js, line 52 on page 434.

clearCacheItem() method

Deprecated: Deprecated since version 2.2 - use `hwc.clearCacheItem(cachekey)`

Syntax

`clearCacheItem()`

Source

hwc-comms.js, line 47 on page 434.

closeWorkflow() method

Deprecated: Deprecated since version 2.2 - use hwc.close()

Syntax

```
closeWorkflow()
```

Source

hwc-comms.js, line 42 on page 433.

expireCredentials() method

Deprecated: Deprecated since version 2.2 - use hwc.close()

Syntax

```
expireCredentials()
```

Source

hwc-comms.js, line 57 on page 434.

getXMLHttpRequest() method

Deprecated: Deprecated since version 2.2 - use hwc.getXMLHttpRequest()

Syntax

```
getXMLHttpRequest()
```

Source

hwc-comms.js, line 32 on page 433.

guid() method

Deprecated: Deprecated since version 2.2 - use hwc.guid()

Syntax

```
guid()
```

Source

hwc-comms.js, line 27 on page 433.

logToWorkflow() method

Deprecated: Deprecated since version 2.2 - use hwc.log(sMsg, eLevel, notifyUser)

Syntax

`logToWorkflow()`

Source

hwc-comms.js, line 37 on page 433.

markAsActivated() method

Deprecated: Deprecated since version 2.2 - use `hwc.markAsActivated()`

Syntax

`markAsActivated()`

Source

hwc-comms.js, line 92 on page 435.

markAsProcessed() method

Deprecated: Deprecated since version 2.2 - use `hwc.markAsProcessed()`

Syntax

`markAsProcessed()`

Source

hwc-comms.js, line 87 on page 435.

processDataMessage(incomingDataMessageValue, noUI, loading, fromActivationFlow, dataType) method

Delegate for data message processing details.

In the custom case, the user is expected to provide their own implementation. In the default SUP HybridApp case, this updates values then sets the next screen to navigate to.

Syntax

`processDataMessage(incomingDataMessageValue, noUI, loading, fromActivationFlow, dataType)`

Parameters

Name	Type	Description
<i>incomingDataMessageValue</i>	string	The XML formatted string for the incoming message

<i>noUI</i>	boolean	true if this has no UI
<i>loading</i>	boolean	If true, this is being called while the application is loading
<i>fromActivationFlow</i>	boolean	If true, this is being called from within an activation flow
<i>dataType</i>	string	If supplied, the data type of the value display on target screen

Source

hwc-comms.js, line 104 on page 436.

processWorkflowMessage() method

Deprecated: Deprecated since version 2.2 - use `hwc.processDataMessage(incomingDataMessageValue, noUI, loading, fromActivationFlow, dataType)`

Syntax

`processWorkflowMessage()`

Source

hwc-comms.js, line 118 on page 436.

saveLoginCertificate() method

Deprecated: Deprecated since version 2.2 - use `hwc.saveLoginCertificate(certificate)`

Syntax

`saveLoginCertificate()`

Source

hwc-comms.js, line 67 on page 434.

saveLoginCredentials() method

Deprecated: Deprecated since version 2.2 - use `hwc.saveLoginCredentials(userName, password)`

Syntax

`saveLoginCredentials()`

*Source**hwc-comms.js, line 72 on page 435.***showCertificatePicker() method**Deprecated: Deprecated since version 2.2 - use `hwc.showCertificatePicker()`*Syntax*`showCertificatePicker()`*Source**hwc-comms.js, line 62 on page 434.***showUrlInBrowser() method**Deprecated: Deprecated since version 2.2 - use `hwc.showUrlInBrowser(url)`*Syntax*`showUrlInBrowser()`*Source**hwc-comms.js, line 82 on page 435.***Source code****Callbacks.js**

```

1      /*
2      * Sybase Hybrid App version 2.3.4
3      *
4      * Callbacks.js
5      * This file will not be regenerated, so it is possible to
modify it, but it
6      * is not recommended.
7      *
8      * Copyright (c) 2012 Sybase Inc. All rights reserved.
9      */
10
11     /**

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
12     * The namespace for the Hybrid Web Container javascript
13     * @namespace
14     */
15     hwc = (typeof hwc === "undefined" || !hwc) ? {} :
hwc;      // SUP 'namespace'
16
17
18     (function(hwc, window, undefined) {
19
20         /**
21         * Constructs CallbackSet object. This object is not meant
for general use.
22         * @private
23         * @constructor
24         * @memberOf hwc
25         */
26         hwc.CallbackSet = function() {
27             hwc.CallbackSet.setCount++;
28             this.setid = hwc.CallbackSet.setCount;
29         };
30
31         /**
32         * @private
33         * @static
34         * @memberOf hwc.CallbackSet
35         */
36         hwc.CallbackSet.setCount = 0;
37
38         /**
39         * @private
40         * @static
41         * @memberOf hwc.CallbackSet
42         */
```



```

43     hwc.CallbackSet.callbacks = {};
44
45     /**
46      * Registers a callback to be handled from container
47      * @memberOf hwc.CallbackSet
48      * @private
49      * @param {string} methodName The name of the callback.
50      * @param {function} callback The function pointer to the
51      * callback
52      * @returns {string} callbackId that can be used by the
53      * container
54      */
55     hwc.CallbackSet.prototype.registerCallback = function
56     (methodName, callback) {
57         if (!hwc.CallbackSet.callbacks[this.setId]) {
58             hwc.CallbackSet.callbacks[this.setId] = {};
59         }
60         hwc.CallbackSet.callbacks[this.setId][methodName] =
61         callback;
62         return this.setId + ':' + methodName;
63     };
64
65     /**
66      * Invoked asynchronously to handle callback from
67      * container
68      * @memberOf hwc.CallbackSet
69      * @static
70      * @private
71      * @param {string} callbackId The id of the callback.
72      * Format is "setid:methodname"
73      * @param {boolean} removeSet True if the callback set
74      * should be removed
75      * @param {array} args The arguments to be passed to the
76      * registered callback

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
70      */
71      hwc.CallbackSet.callbackHandler = function(callbackId,
removeSet, args) {
72          var callbackSet, c, callback;
73          c = callbackId.split(':', 2);
74
75          if ( c && c.length === 2 ) {
76              callbackSet = hwc.CallbackSet.callbacks[c[0]];
77
78              if (callbackSet) {
79                  callback = callbackSet[c[1]];
80
81                  if (removeSet) {
82                      delete hwc.CallbackSet.callbacks[c[0]];
83                  }
84
85                  if (callback) {
86                      callback.apply(callback, args);
87                  }
88              }
89          }
90      };
91
92      window.CallbackSet = [];
93      window.CallbackSet.callbackHandler =
hwc.CallbackSet.callbackHandler;
94
95      })(hwc, window);
96
97
98
```

Camera.js

```
1      /*
2      * Sybase Hybrid App version 2.3.4
3      *
4      * Camera.js
5      * This file will not be regenerated, so it is possible to
modify it, but it
6      * is not recommended.
7      *
8      * Copyright (c) 2012 Sybase Inc. All rights reserved.
9      */
10
11     /* The feature comment is necessary at the class level for
the custom template to work.
12     */
13     /**
14     * The namespace for the Hybrid Web Container javascript
15     * @namespace
16     * @desc Camera
17     */
18     hwc = (typeof hwc === "undefined" || !hwc) ? {} : hwc;    //
SUP 'namespace'
19
20     (function(hwc, window, undefined) {
21
22         /**
23         * An array that holds all possible option codes for use
with getPicture()
24         * @private
25         */
26         hwc.PictureOption = [];
27
28         /**
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
29         * @memberOf hwc.PictureOption
30     */
31     hwc.PictureOption.SourceType = {
32         /**
33         * Constant that specifies the built-in camera as the
34         image source for selecting the image using the {@link hwc.getPicture}
35         method.
36         * @memberOf hwc.PictureOption.SourceType
37         */
38         CAMERA: 1, // Specifies the built-in camera
39         as the image source where image content is not persisted by the
40         device
41         /**
42         * Constant that specifies the photo library as the
43         image source
44         * @memberOf hwc.PictureOption.SourceType
45         */
46         PHOTOLIBRARY: 2, // Specifies the photo library
47         as the image source where image content is already persisted on the
48         device
49         /**
50         * Constant that specifies the built-in camera and the
51         photo library be used as an image source for selecting the image
52         using the {@link hwc.getPicture} method.
53         * @memberOf hwc.PictureOption.SourceType
54         */
55         BOTH: 3 // Specifies the built-in camera
56         as the image source where image content is persisted by the device
57     };
58     /**
59     * @memberOf hwc.PictureOption
60     */
61     hwc.PictureOption.DestinationType = {
62         /**
63         * Use this constant to specify that base64 encoded image
64         data be returned by the {@link hwc.getPicture} method.
```

```

55         * @memberOf hwc.PictureOption.DestinationType
56         * @deprecated
57         */
58         IMAGE_DATA: 0,           // Returns base64 encoded string
(deprecated)
59         /**
60         * Use this constant to specify that the image URI be
returned by the {@link hwc.getPicture} method.
61         * @memberOf hwc.PictureOption.DestinationType
62         */
63         IMAGE_URI: 1             // Returns uniform reference
identifier for the image
64     };
65
66     /**
67     * Open a platform-specific application allowing the user
to capture an image
68     * using the built-in camera.
69     * @deprecated
70     */
71     hwc.PictureOption.CAMERA =
hwc.PictureOption.SourceType.CAMERA;
72
73     /**
74     * Open a platform-specific application allowing the user
to select an
75     * existing picture from a gallery.
76     * @deprecated
77     */
78     hwc.PictureOption.PHOTOLIBRARY =
hwc.PictureOption.SourceType.PHOTOLIBRARY;
79
80     /**
81     * An array that holds all possible error codes
82     */

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
83         hwc.PictureError = [];  
84  
85         /**  
86          * Constant indicating that the {@link hwc.getPicture}  
method was successful.  
87          * @memberOf hwc  
88          */  
89         hwc.PictureError.NO_ERROR      = 0;  
90  
91         /**  
92          * Constant indicating that the {@link hwc.getPicture}  
method is not implemented, camera not present, etc.  
93          * @memberOf hwc  
94          */  
95         hwc.PictureError.NOT_SUPPORTED = -1;  
96  
97         /**  
98          * Constant indicating that the {@link hwc.getPicture}  
method has been invoked, but has not completed yet.  
99          * @memberOf hwc  
100          */  
101         hwc.PictureError.IN_PROGRESS  = -2;  
102  
103         /**  
104          * Constant indicating that the user has cancelled the  
{@link hwc.getPicture} invocation.  
105          * @memberOf hwc  
106          */  
107         hwc.PictureError.USER_REJECT  = -3;  
108  
109         /**  
110          * Constant indicating that the supplied options were not  
recognized by the {@link hwc.getPicture} method  
111          * @memberOf hwc
```

```

112     */
113     hwc.PictureError.BAD_OPTIONS    = -4;
114
115     /**
116     * Constant indicating that the returned image size was
117     * too large to be handled by JavaScript.
118     * @memberOf hwc
119     */
120     hwc.PictureError.TOO_LARGE      = -5;
121
122     /**
123     * Constant indicating that the an unknown error occurred
124     * during the execution of {@link hwc.getPicture} method.
125     * @memberOf hwc
126     */
127     hwc.PictureError.UNKNOWN        = -6;
128
129     /**
130     * A namespace for our private use
131     * @private
132     */
133     var _Picture = new function() {}; // private
134     object '_Picture' within 'hwc'
135
136     /**
137     * Requests retrieval of a picture asynchronously.
138     *
139     * @param {anonymous.onGetPictureError} onGetPictureError
140     * Function to be invoked if the attempt to get
141     * a picture fails. err will be one of the PictureError
142     * codes.
143     *
144     * @param {anonymous.onGetPictureSuccess}
145     * onGetPictureSuccess Function to be invoked if a picture is
146     * successfully retrieved. response will either be a
147     * Base64-encoded JPG string or a URI.

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
140     * @param {anonymous.PictureOptions} options the options
to control the sourceType and destinationType.
141     * @desc Camera
142     * @memberOf hwc
143     * @public
144     * @example
145     * // Error handler. will be invoked asynchronously.
146     * fail = function(errorCode){
147     *     // handle error code and take appropriate
action.
148     * }
149     * // Success handler. will be invoked asynchronously.
150     * success = function(fileName, content){
151     *     // handle the content. content may be a location or
base64 encoded string that is
152     *     // determined by the options passed to the
destinationType argument.
153     * }
154     *
155     * getPicture(fail,
156     *             success,
157     *             { sourceType:
PictureOption.SourceType.CAMERA,
158     *               destinationType:
PictureOption.DestinationType.IMAGE_URI
159     *             });
160     */
161     hwc.getPicture = function(onGetPictureError,
onGetPictureSuccess, options)
162     {
163         hwc.traceEnteringMethod("hwc.getPicture");
164         try {
165             // Return if callback functions are not
provided
166             if (typeof onGetPictureError !== 'function' ||
```



```
167         typeof onGetPictureSuccess !== 'function')
168     {
169         return;
170     }
171     if ("_onGetPictureSuccess" in _Picture &&
172         _Picture._onGetPictureSuccess !== null)
173     {
174         // Already requested but not yet complete
175         onGetPictureError(hwc.PictureError.IN_PROGRESS);
176         return;
177     }
178     _Picture._onGetPictureError =
179     onGetPictureError;
180     _Picture._onGetPictureSuccess =
181     onGetPictureSuccess;
182     // Convert options parameter to object notation if
183     // number type and return image data to preserve behavior
184     // of previous release
185     if (typeof options === 'number') {
186         options = { destinationType:
187             hwc.PictureOption.DestinationType.IMAGE_DATA,
188             sourceType: options
189         };
190     }
191     // Convert options object to serialized JSON text
192     // in preparation for submission to the container
193     options = JSON.stringify(options);
194     if (hwc.isWindowsMobile())
195     {
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
194         hwc.getDataFromContainer("getPicture",
"PictureOptions=" + encodeURIComponent(options));
195     }
196     else if (hwc.isIOS())
197     {
198         // Only difference between iOS and WindowsMobile
above is the leading '&'
199         hwc.getDataFromContainer("getPicture",
"&PictureOptions=" + encodeURIComponent(options));
200     }
201     else
202     {
203         _HWC.getPicture(options);
204     }
205     } finally {
206         hwc.traceLeavingMethod("hwc.getPicture");
207     }
208 };
209
210 /**
211     * (Internal) Invoked asynchronously when the image
arrives.
212     *
213     * @private
214     * @param result The PictureError code, or
PictureError.NO_ERROR for
215     *     success.
216     * @param {string} filename Filename corresponding to the
image.
217     * @param {string} imageData Base64-encoded String
containing the image data. Undefined
218     *     if the result parameter indicates an error or the
image URI was requested.
219     * @param {string} imageUri Uniform resource indicator of
the image resource. Undefined
```

```

220      *      if the result parameter indicates an error or the
image data was requested.
221      */
222      _Picture._getPictureComplete = function(result, fileName,
imageData, _imageUri) {
223          var response, successFunc, errorFunc;
224
225          hwc.traceEnteringMethod("_Picture._getPictureComplete");
226          try {
227              successFunc = _Picture._onGetPictureSuccess;
228              errorFunc = _Picture._onGetPictureError;
229
230              _Picture._onGetPictureSuccess = null;
231              _Picture._onGetPictureError = null;
232
233              if (result === hwc.PictureError.NO_ERROR) {
234                  if (imageData) {
235                      // For WM client, the picture data is too big
to be passed from url, so only
236                      // the unique key is sent from container to
JavaScript. JavaScript needs to send
237                      // another xmlhttprequest to fetch the
actual data
238                      if (hwc.isWindowsMobile()) {
239                          response =
hwc.getDataFromContainer("getpicturedata", "pictureid=" +
imageData);
240                          successFunc(fileName, response);
241                      } else {
242                          successFunc(fileName, imageData);
243                      }
244                      } else if (imageUri) {
245                          successFunc(fileName, imageUri);
246                      } else {

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
247         errorFunc(hwc.PictureError.UNKNOWN);
248     }
249     } else {
250         errorFunc(result);
251     }
252     } finally {
253 hwc.traceLeavingMethod("_Picture._getPictureComplete");
254     }
255     };
256
257     window._Picture = _Picture;
258   })(hwc, window);
259
260
261   /**
262    * Used to group anonymous objects and callback functions used
263    * as method parameters. Methods and fields in this
264    * namespace cannot be instantiated. Used for API docs
265    * generation only.
266    * @namespace
267    */
268
269   anonymous = (typeof anonymous === "undefined" || !
270 anonymous) ? {} : anonymous; // SUP 'namespace'
271
272   /**
273    * User provided function that is invoked when the {@link
274    hwc.getPicture} function fails.
275    *
276    * @name anonymous.onGetPictureError
277    * @param {number} err the error code returned. Possible
278    values are
279    * <ol>
280    * <li>PictureError.NO_ERROR = 0;</li>
```

```

275      * <li>PictureError.NOT_SUPPORTED = -1;  getPicture() not
implemented, camera not present,</li>
276      * <li>PictureError.IN_PROGRESS = -2; getPicture() has
already been requested but has not yet completed.</li>
277      * <li>PictureError.USER_REJECT = -3; the user has canceled
the request.</li>
278      * <li>PictureError.BAD_OPTIONS = -4; supplied options were
not recognized.</li>
279      * <li>PictureError.TOO_LARGE = -5; the returned image size
was too large to be handled by JavaScript</li>
280      * <li>PictureError.UNKNOWN = -6; an unknown error
occurred.</li>
281      * </ol>
282      * @desc Camera
283      * @function
284      */
285
286      /**
287      * User provided function that will be invoked when the
{@link hwc.getPicture} function is successful.
288      *
289      * @name anonymous.onGetPictureSuccess
290      *
291      * @param {string} filename file name of the image
292      * @param {string} response the response will be either a
Base64-encoded JPG string or a URI depending on the options passed
to
293      * the {@link hwc.getPicture} function.
294      * <ul>
295      * <li> if options.destinationType ==
PictureOption.DestinationType.IMAGE_URI, response is an uniform
reference identifier for the image. onGetPictureSuccess(fileName,
imageURI)</li>
296      * <li> if options.destinationType ==
PictureOption.DestinationType.IMAGE_DATA, response is a Base64-
encoded string. onGetPictureSuccess(fileName, imageData )</li>
297      * </ul>
298      * @function

```

```
299      */
300
301      /**
302      * Options object that is used with the {@link
303      hwc.getPicture} method. Contains 2 fields that can be specified.
304      *
305      * <ul>
306      * <li> sourceType: One of {@link hwc.Picture.SourceType}
307      values </li>
308      * <li> destinationType: One of {@link
309      hwc.Picture.DestinationType} values </li>
310      * </ul>
311      * @name anonymous.PictureOptions
312      * @see hwc.getPicture for an example.
313      */
```

Certificate.js

```
1      /*
2      * Sybase Hybrid App version 2.3.4
3      *
4      * Certificate.js
5      * This file will not be regenerated, so it is possible to
6      modify it, but it
7      * is not recommended.
8      *
9      * Last Updated: 2011/6/29
10     *
11     * Copyright (c) 2012 Sybase Inc. All rights reserved.
12     *
13     * Note a certificate object will have the following fields
14     - issuerCN - The common name (CN) from the certificate
15     issuer's distinguished name.
16     - issuerDN - The certificate issuer's distinguished name,
17     in string form.
18     - notAfter - End time for certificate's validity period,
19     with date/time fields as they would appear in UTC.
```

```

16     - notBefore - Start time for the certificate's validity
    period, with date/time fields as they would appear in UTC.
17     - signedCertificate - The digitally signed certificate in
    Base64 format
18     - subjectCN - The common name (CN) from the certificate
    subject's distinguished name.
19     - subjectDN - The certificate subject's distinguished
    name, in string form.
20     */
21
22     /**
23     * This class represents an X.509 public certificate store.
24     */
25
26     /**
27     * The namespace for the Hybrid Web Container javascript
28     * @namespace
29     */
30     hwc = (typeof hwc === "undefined" || !hwc) ? {} : hwc;    //
    SUP 'namespace'
31
32
33     (function(hwc, window, undefined) {
34     /**
35     * Use these functions for X.509 credential handling.
36     * <p>
37     * Use these functions to create a user interface in HTML and
    JavaScript, that uses X.509 certificates as the Workflow
    credentials.
38     * </p>
39     * <p>
40     * This file contains the functions that allow parsing a
    certificate date, creating a certificate from a JSON string value,
    retrieving a certificate from a file (Android), retrieving a
    certificate from the server (iOS), and so on.
41     * </p>

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
42     * @classdesc
43     * @memberOf hwc
44     */
45     hwc.CertificateStore = function() {
46     };
47
48     (function() {
49         /**
50         * Private function
51         * Convert string type date to JavaScript Date
52         * Format: 2014-05-24T20:00:12Z -> Sat May 24 2014
53         * 16:00:12 GMT-0400 (Eastern Daylight Time)
54         *
55         * @private
56         * @param {string} value Date string to parse
57         * @returns Javascript type Date object
58         */
59         function parseCertDate(value) {
60             var a = /^(\d{4})-(\d{2})-(\d{2})T(\d{2}):(\d{2}):
61             (\d{2})(?:\.\d*)?Z$/i.exec(value);
62             return new Date(Date.UTC(+a[1], +a[2] - 1, +a[3],
63             +a[4], +a[5], +a[6]));
64         }
65         /**
66         * Private function
67         * Create certificate object
68         *
69         * @private
70         * @param {string} value JSON string type certificate
71         *
72         * {"subjectDN":"CN=android, OU=SUP, O=Sybase,
73         * L=Dublin, ST=California, C=US",
74         * "notBefore":"2012-05-24T20:00:12Z",
75         * "notAfter":"2014-05-24T20:00:12Z",
```



```

72         *           "subjectCN":"android",
73         *           "signedCertificate":"base64 encoded string
here",
74         *           "issuerDN":"CN=teva, CN=sybase.com,
OU=Unwired Enterprise, O=Sybase Inc., L=Dublin, ST=California,
C=US",
75         *           "issuerCN":"teva"}
76         * @returns Certificate object
77         */
78         function createCert(value) {
79             var cert;
80             if (value === null || typeof value === 'undefined' ||
value.length === 0) {
81                 return null;
82             }
83
84             cert = JSON.parse(value);
85
86             if (cert.notAfter) {
87                 cert.notAfter = new
Date(parseCertDate(cert.notAfter));
88             }
89             if (cert.notBefore) {
90                 cert.notBefore = new
Date(parseCertDate(cert.notBefore));
91             }
92
93             return cert;
94         }
95
96         /**
97         * Returns a list of all the certificate labels in this
store (can be empty). Each certificate in this store has a unique
label.
98         *

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
99      * <b>Supported Platforms: </b> Windows Mobile and
BlackBerry.
100     * @desc Certificate
101     * @public
102     * @memberOf hwc.CertificateStore
103     * @param {String} filterSubject filter of subject
104     * @param {String} filterIssuer filter of issuer
105     * @returns {String[]} Only filtered certificate labels
106     * @example
107     * // The following script gets all the labels for
certificates
108     * // with the provided subject and issuer
109     * var certStore = CertificateStore.getDefault();
110     * var labels = certStore.certificateLabels("MyUser",
"mydomain.com");
111     */
112     hwc.CertificateStore.prototype.certificateLabels =
function(filterSubject, filterIssuer) {
113         var response = "";
114
115         hwc.traceEnteringMethod("hwc.CertificateStore.certificateLabels");
116         try {
117             filterSubject = filterSubject ? filterSubject :
"";
118             filterIssuer = filterIssuer ? filterIssuer :
"";
119
120             if (hwc.isWindowsMobile()) {
121                 response =
hwc.getDataFromContainer("certificatestore",
"&command=certificateLabels" +
122                     "&filterSubject=" +
encodeURIComponent(filterSubject) + "&filterIssuer=" +
encodeURIComponent(filterIssuer));
123             }
124             else if (hwc.isBlackBerry()) {
```

```

125         response =
_HWC.getCertificateLabels(filterSubject, filterIssuer);
126     }
127     else {
128         throw "Not supported on this platform";
129     }
130
131     return eval('(' + response + ')');
132 } finally {
133 hwc.traceLeavingMethod("hwc.CertificateStore.certificateLabels");
134 }
135 };
136
137 /**
138  * Returns a certificate without the signedCertificate part
set.
139  * @desc Certificate
140  * @public
141  * @memberOf hwc.CertificateStore
142  * @returns {hwc.CertificateStore} a certificate without
the signedCertificate part set
143  */
144 hwc.CertificateStore.getDefault = function() {
145     return new hwc.CertificateStore();
146 };
147
148 /**
149  * Returns a certificate without the signedCertificate part
set.
150  *
151  * <b> Supported Platforms </b>: Windows Mobile and
BlackBerry.
152  * @desc Certificate

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
153      * @public
154      * @memberOf hwc.CertificateStore
155      * @param {String} label label of the desired
certificate
156      * @returns certificate object
157      * @example
158      * // The following script gets the certificate data for
the first
159      * // certificate to match the provided subject and
issuer
160      * var certStore = CertificateStore.getDefault();
161      * var labels = certStore.certificateLabels("MyUser",
"mydomain.com");
162      * var cert = certStore.getPublicCertificate(labels[0]);
163      */
164      hwc.CertificateStore.prototype.getPublicCertificate =
function(label) {
165          var response = "";
166
167          hwc.traceEnteringMethod("hwc.CertificateStore.getPublicCertificate"
);
168          try {
169              if (hwc.isWindowsMobile()) {
170                  response =
hwc.getDataFromContainer("certificatestore",
"&command=getPublicCertificate" +
171                      "&label=" +
encodeURIComponent(label));
172              }
173              else if (hwc.isBlackBerry()) {
174                  response =
_HWC.getPublicCertificate(label);
175              }
176              else {
177                  throw "Not supported on this platform";
178              }

```

```

179
180         return createCert(response);
181     } finally {
182         hwc.traceLeavingMethod("hwc.CertificateStore.getPublicCertificate")
183         ;
184     }
185
186
187     /**
188     * Returns the certificate with the specified label, and
189     * decrypts it if necessary using the specified password,
190     * or returns null if the certificate is encrypted and the
191     * password is incorrect.
192     *
193     * <b>Supported Platforms</b>: Windows Mobile and
194     * BlackBerry
195     * @desc Certificate
196     *
197     * @public
198     * @memberOf hwc.CertificateStore
199     * @param {String} label label of the desired
200     * certificate
201     * @param {String} password Access password for the private
202     * key of the certificate. Pass null unless the platform requires a
203     * password.
204     * @returns Certificate object
205     * @example
206     * // The following script gets the signed certificate data
207     * for the first
208     * // certificate to match the provided subject and
209     * issuer
210     * var certStore = CertificateStore.getDefault();
211     * var labels = certStore.certificateLabels("MyUser",
212     * mydomain.com");
213     * var cert = certStore.getSignedCertificate(labels[0]);

```

```
205      *
206      * var username = cert.subjectCN;
207      * var password = cert.signedCertificate;
208      */
209      hwc.CertificateStore.prototype.getSignedCertificate =
function(label, password) {
210          var response = "";
211
212      hwc.traceEnteringMethod("hwc.CertificateStore.getSignedCertificate"
);
213          try {
214              if (hwc.isWindowsMobile()) {
215                  response =
hwc.getDataFromContainer("certificatestore",
"&command=getSignedCertificate" +
216                      "&label=" +
encodeURIComponent(label));
217              } else if (hwc.isBlackBerry()) {
218                  response =
_HWC.getSignedCertificate(label);
219              } else {
220                  throw "Not supported on this platform";
221              }
222
223              return createCert(response);
224          } finally {
225      hwc.traceLeavingMethod("hwc.CertificateStore.getSignedCertificate")
;
226          }
227      };
228
229      /**
230      * Returns a list of full path names for the certificate
files found in the
```

```

231      * file system for import.
232      *
233      * <b>Supported Platforms</b>: Android
234      * @desc Certificate
235      * @memberOf hwc.CertificateStore
236      * @public
237      * @param {String} sFolder Folder in which to search for
files. This should be a full
238      *      absolute path, based on the root of the device file
system. The
239      *      separator may be either "/" or "\". For example,
"\sdcard\mycerts"
240      *      or "/sdcard/mycerts" is acceptable. Do not
include any http
241      *      prefixes, such as "file:".
242      * @param {String} sFileExtension File extension to which
the list should be
243      *      restricted. Pass the string expected after the
"." in the file
244      *      name. For example, to match *.p12, pass "p12" as
the argument.
245      *      Pass null to return all files in the folder.
246      * @returns {String[]} A list of Strings, each String being
the full path name of a
247      *      matched file in the given folder.
248      * @example
249      * // The following script gets an array of file paths for
files on
250      * // the sdcard with the extension p12
251      * var certStore = CertificateStore.getDefault();
252      * var certPaths =
certStore.listAvailableCertificatesFromFileSystem("/sdcard/",
"p12");
253      */
254      hwc.CertificateStore.prototype.listAvailableCertificatesFromFileSys
tem = function(sFolder, sFileExtension) {

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
255         var response = "";
256
257 hwc.traceEnteringMethod("hwc.CertificateStore.listAvailableCertificatesFromFileSystem");
258         try {
259             if (hwc.isAndroid()) {
260                 response =
                _HWC.listAvailableCertificatesFromFileSystem(sFolder,
                sFileExtension);
261             } else {
262                 throw "Not supported on this platform";
263             }
264
265             return eval('(' + response + ')');
266         } finally {
267 hwc.traceLeavingMethod("hwc.CertificateStore.listAvailableCertificatesFromFileSystem");
268         }
269     };
270
271     /**
272     * Gets a certificate from a file.
273     *
274     * <b>Supported Platforms</b>: Android
275     * @desc Certificate
276     * @public
277     * @memberOf hwc.CertificateStore
278     * @param {String} filePath The absolute path to the
    file.
279     * @param {String} password The password needed to access
    the certificate's private data.
280     * @example
281     * // The following script gets the signed certificate
    data for the first
```



```
282     * // p12 file found on the sdcard
283     * var certStore = CertificateStore.getDefault();
284     * var certPaths =
certStore.listAvailableCertificatesFromFileSystem("/sdcard/",
"p12");
285     * var cert =
certStore.getSignedCertificateFromFile(certPaths[0], "password");
286     */
287
hwc.CertificateStore.prototype.getSignedCertificateFromFile =
function(filePath, password) {
288         var response = "";
289
290 hwc.traceEnteringMethod("hwc.CertificateStore.getSignedCertificateF
romFile");
291         try {
292             if (hwc.isAndroid()) {
293                 response =
_HWC.getSignedCertificateFromFile(filePath, password);
294             } else if (hwc.isIOS()) {
295                 response =
hwc.getDataFromContainer("certificatestore",
"&command=getSignedCertificateFromFile" +
296
                "&filePath=" +
encodeURIComponent(filePath) + "&password=" +
encodeURIComponent(password));
297             }
298             else {
299                 throw "Not supported on this platform";
300             }
301
302             return createCert(response);
303         } finally {
304 hwc.traceLeavingMethod("hwc.CertificateStore.getSignedCertificateFr
omFile");
305         }
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
306     };
307
308
309     /**
310     * Gets a certificate from the server.
311     *
312     * <b>Supported Platforms</b>: iOS
313     * @desc Certificate
314     * @public
315     * @memberOf hwc.CertificateStore
316     * @param {String} username The username for the Windows
    user (in the form "DOMAIN\\username")
317     * @param {String} serverPassword The password for the
    Windows user
318     * @param {String} certPassword The password needed to
    access the certificate (may be the same or different from the Windows
    password)
319     * @example
320     * // The following script gets the signed certificate data
    for the
321     * // user MYDOMAIN\MYUSERNAME from the server
322     * var certStore = CertificateStore.getDefault();
323     * cert =
    certStore.getSignedCertificateFromServer("MYDOMAIN\MYUSERNAME",
    "myserverpassword", "mycertpassword");
324     */
325
    hwc.CertificateStore.prototype.getSignedCertificateFromServer =
    function(username, serverPassword, certPassword) {
326         var response = "";
327
328
    hwc.traceEnteringMethod("hwc.CertificateStore.getSignedCertificateF
    romServer");
329         try {
330             if (hwc.isIOS()) {
```

```

331         response =
hwc.getDataFromContainer("certificatestore",
"&command=getSignedCertificateFromServer" +
332             "&username=" +
encodeURIComponent(username) + "&serverPassword=" +
encodeURIComponent(serverPassword) +
333             "&certPassword=" +
encodeURIComponent(certPassword));
334     } else {
335         throw "Not supported on this platform";
336     }
337
338     return eval('(' + response + ')');
339 } finally {
340 hwc.traceLeavingMethod("hwc.CertificateStore.getSignedCertificateFr
omServer");
341     }
342 };
343
344 /**
345  * Gets a certificate from the Afaria server.
346  * To retrieve an x509 certificate from Afaria, you must
get a CertificateStore and then call getSignedCertificateFromAfaria.
If Afaria is installed and configured on the device, this gets the
Afaria seeding file from the Afaria server.
347  * If the seeding file is retrieved from the Afaria server,
the user is prompted to update user specific information in the
Settings screen.
348  *
349  * <b>Supported Platforms</b>: iOS, Android & BlackBerry
350  * @desc Certificate
351  * @public
352  * @memberOf hwc.CertificateStore
353  * @param {String} commonName Common name used to generate
the certificate by Afaria
354  * @param {String} challengeCode Challenge code for the
user so that CA can verify and sign it

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
355      * @returns JSON object with CertBlob in Base64 encoded
format and other information about certificate
356      * @throws If called on a platform that is not
supported.
357      * @example
358      * // The following script gets a signed certificate from
the Afaria server.
359      * var certStore = CertificateStore.getDefault();
360      * cert =
certStore.getSignedCertificateFromAfaria("Your_CN",
"CA_challenge_code");
361      */
362
hwc.CertificateStore.prototype.getSignedCertificateFromAfaria =
function(commonName, challengeCode) {
363      var response = "";
364
365      hwc.traceEnteringMethod("hwc.CertificateStore.getSignedCertificateF
romAfaria");
366      try {
367          if (hwc.isIOS()) {
368              response =
hwc.getDataFromContainer("certificatestore",
"&command=getSignedCertificateFromAfaria" +
369                  "&commonname=" +
encodeURIComponent(commonName) + "&challengecode=" +
encodeURIComponent(challengeCode));
370          } else if (hwc.isAndroid() || hwc.isBlackBerry())
{
371              response =
_HWC.getSignedCertificateFromAfaria(commonName, challengeCode);
372          }
373          else {
374              throw "Not supported on this platform";
375          }
376
377      return eval('(' + response + ')');
```

```

378         } finally {
379             hwc.traceLeavingMethod("hwc.CertificateStore.getSignedCertificateFromAfarria");
380         }
381     };
382     } ());
383
384     })(hwc, window);
385

```

ExternalResource.js

```

1     /*
2     * Sybase Hybrid App version 2.3.4
3     *
4     * ExternalResource.js
5     *
6     * This file will not be regenerated, so it is possible to
modify it, but it
7     * is not recommended.
8     *
9     * Copyright (c) 2012 Sybase Inc. All rights reserved.
10    *
11    */
12    /**
13    * The namespace for the Hybrid Web Container javascript
14    * @namespace
15    */
16    hwc = (typeof hwc === "undefined" || !hwc) ? {} : hwc; //
SUP 'namespace'
17
18    (function() {
19
20    /**

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
21      * Makes an external cross domain request.
22      *
23      * @public
24      * @memberOf hwc
25      * @param {String} url The url to make request to
26      * @param {anonymous.options} options a set of key/value
  pairs that configure the underlying request.
27      *
28      * @example
29      *
30      *     var options = {
31      *         method: "GET",
32      *         data: "data",
33      *         async: true,
34      *         headers: {
35      *             "Content-Type": "text/plain;charset=UTF-8"
36      *         },
37      *         complete: function(response) {
38      *             // invoked when the request completes
  (asynchronous mode)
39      *             if (response.status === 200)
40      *                 alert("Update successful");
41      *             else
42      *                 alert("Update Failed");
43      *         }
44      *     };
45      *
46      *     getExternalResource(url, options);
47      *
48      */
49      hwc.getExternalResource = function(url, options) {
50          var key, _options, params=[], queryString, request,
  callbackSet, jsonOptions, jsonText, xmlhttp;
```

```
51
52     hwc.traceEnteringMethod("hwc.getExternalResource");
53     try {
54         // Default options
55         _options = {
56             method: "GET",
57             async: true
58             //headers: {},
59             //data: '',
60             //complete: function() {}
61         };
62
63         // Fill in options
64         options = options || {};
65
66         for (key in options) {
67             _options[key] = options[key];
68         }
69
70         options = _options;
71         options.method = options.method.toUpperCase();
72
73         if (typeof (options.data) === 'string') {
74             params.push(options.data);
75         }
76         else if
77         (Object.prototype.toString.call(options.data) === '[object Array]')
78         {
79             params = options.data;
80         }
81         else {
82             for (key in options.data) {
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
81         params.push(encodeURIComponent(key) + "=" +
encodeURIComponent(options.data[key]));
82     }
83 }
84
85     // Format query string and post data
86     queryString = params.join("&");
87
88     if (queryString) {
89         if (options.method === "GET") {
90             url = url + (url.indexOf("?") === -1 ? '?' :
'&') + queryString;
91             options.data = "";
92         }
93         else {
94             options.data = queryString;
95         }
96     }
97
98     // Make request
99     if (hwc.isBlackBerry()) {
100         request = hwc.getXMLHttpRequest();
101         request.open(options.method, url,
options.async);
102
103         if (options.headers) {
104             for (key in options.headers) {
105                 request.setRequestHeader(key,
options.headers[key]);
106             }
107         }
108
109         request.onreadystatechange = function() {
110             if (request.readyState === 4) {
```



```
111             handleResponse(options, request);
112         }
113     };
114
115         request.send(options.data);
116     }
117     else if (hwc.isAndroid()){
118         if (options.async) {
119             // Setup callbacks
120             callbackSet = new hwc.CallbackSet();
121             options.callback =
callbackSet.registerCallback("callback", function(response)
{ handleResponse(options, response); });
122         }
123
124         // Create a json string for options
125         jsonOptions = JSON.stringify(options);
126
127         jsonText = _HWC.makeExternalRequest(url,
jsonOptions) + "";
128
129         if (!options.async && jsonText) {
130             handleResponse(options,
JSON.parse(jsonText));
131         }
132     }
133     else if (hwc.isWindowsMobile() || hwc.isWindows())
{
134         // Create a json string for options
135         jsonOptions = JSON.stringify(options);
136
137         try {
138             //make xmlhttp request to load the rmi
response from server
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
139         xmlhttp = hwc.getXMLHttpRequest();
140
141         //container always sends the request as
synced, javascript sends the request based on
142         //caller's choice
143         xmlhttp.open("POST", "/sup.amp?
querytype=externalresource&" + hwc.versionURLParam, options.async);
144
145         xmlhttp.onreadystatechange = function()
{
146             if (xmlhttp.readyState === 4) {
147                 // Success
148                 if (xmlhttp.status === 200) {
149                     handleResponse(options,
JSON.parse(xmlhttp.responseText));
150                 }
151             }
152         };
153
154         xmlhttp.send("url=" +
encodeURIComponent(url) + "&options=" +
encodeURIComponent(jsonOptions));
155     }
156     catch (ex) {
157         alert(ex);
158     }
159 }
160     else if (hwc.isIOS()) {
161         // Create a json string for options
162         jsonOptions = JSON.stringify(options);
163
164         try {
165             //make xmlhttp request to load the rmi
response from server
166             xmlhttp = hwc.getXMLHttpRequest();
```

```
167
168             //container always sends the request as
synced, javascript sends the request based on
169             //caller's choice
170             xmlhttp.open("GET", "http://localhost/
sup.amp?querytype=externalresource&" + hwc.versionURLParam + "&url="
+ encodeURIComponent(url) + "&options=" +
encodeURIComponent(jsonOptions), options.async);
171             xmlhttp.onreadystatechange = function()
{
172                 if (xmlhttp.readyState === 4) {
173                     // Success
174                     handleResponse(options,
JSON.parse(xmlhttp.responseText));
175                 }
176             };
177
178             xmlhttp.send("");
179
180         }
181         catch (err) {
182             alert(err);
183         }
184     }
185     } finally {
186         hwc.traceLeavingMethod("hwc.getExternalResource");
187     }
188 };
189
190 /**
191  * Internal method to wrap response in a fake xhr
192  * @private
193  * @param {anonymous.options} options The options provided
for the request
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
194     * @param {object} response The response provided by the
container
195     */
196     function handleResponse(options, response) {
197         hwc.traceEnteringMethod("handleResponse");
198         try {
199             var fakeXHR = {
200                 "status": response.status,
201                 "statusText": response.statusText,
202                 "responseText": response.responseText,
203                 "getResponseHeader": function(key) {
204                     var headerValue, header;
205
206                     hwc.traceEnteringMethod("fakeXHR.getResponseHeader");
207                     try {
208                         if (response.getResponseHeader) {
209                             headerValue =
response.getResponseHeader(key);
210                         }
211                         else if (response.headers)
212                             {
213                             for ( header in response.headers)
214                                 if ( key.toLowerCase() ===
header.toLowerCase() )
215                                     {
216                                     headerValue =
response.headers[header];
217                                     break;
218                                 }
219                             }
220                         }
221
```

```
222         return headerValue === undefined ? null :
headerValue;
223     } finally {
224         hwc.traceLeavingMethod("fakeXHR.getResponseHeader");
225     }
226     },
227     "getAllResponseHeaders": function() {
228         var allHeaders, key;
229         hwc.traceEnteringMethod("fakeXHR.getAllResponseHeaders");
230         try {
231             if (response.getAllResponseHeaders)
232                 return
response.getAllResponseHeaders();
233             }
234             if (response.headers) {
235                 for (key in response.headers) {
236                     if (allHeaders) {
237                         allHeaders += "\r\n";
238                     }
239
240                     allHeaders += (key + ":" +
response.headers[key]);
241                 }
242                 return allHeaders;
243             }
244
245             return null;
246         } finally {
247             hwc.traceLeavingMethod("fakeXHR.getAllResponseHeaders");
248         }
249     }
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
250         };
251
252         if (options.complete) {
253             options.complete(fakeXHR);
254         }
255     } finally {
256         hwc.traceLeavingMethod("handleResponse");
257     }
258 }
259 } ());
260
261 /**
262  * Used to group anonymous objects and callback functions used
263  * as method parameters only for purposes of API docs generation only.
264  * Methods and fields in this namespace cannot be
265  * instantiated.
266  * <br/>
267  * <b>Used for API docs generation only.</b>
268  * @namespace
269  */
270
271 anonymous = (typeof anonymous === "undefined" || !
anonymous) ? {} : anonymous; // SUP 'namespace'
272
273 /**
274  * Options object used with the {@link
275  * getExternalResource} function.
276  *
277  * Supported options are:
278  * <ul>
279  * <li> method: one of GET, PUT, DELETE, HEAD, OPTIONS,
280  * or POST. The default is GET.</li>
281  * <li> HTTP and HTTPS urls are supported. </li>
282  * <li> async: request should be sent asynchronously.
283  * The default is true. </li>
```

```

278      * <li> headers: request headers to be sent with request.
</li>
279      * <li> data: data to be sent. If this is an array, it is
converted to a query string. For a GET request, this is added to the
end of the URL. </li>
280      * <li> {@link anonymous.complete} is a callback
function that will be invoked with the resultXHR when this method
completes </li>
281      * </ul>
282      * @name anonymous.options
283      */
284
285      /**
286      * Callback function used in the {@link Options}
object.
287      *
288      * @name anonymous.complete
289      * @param {object} resultXHR the response object.
290      * <br/>
291      * The fields/methods available on resultXHR are
292      * <ol>
293      * <li> status</li>
294      * <li> statusText</li>
295      * <li> responseText</li>
296      * <li> getResponseHeader(key)</li>
297      * <li> getAllResponseHeaders()</li>
298      * </ol>
299      * These fields and methods are not supported for
resultXHR:
300      * <ul>
301      * <li> open() </li>
302      * </ul>
303      * @function
304      */
305

```

hwc-api.js

```
1      /*
2      * Sybase Hybrid App version 2.3.4
3      *
4      * hwc-api.js
5      * This file will not be regenerated, so it is possible to
modify it, but it
6      * is not recommended.
7      *
8      * Copyright (c) 2011,2012 Sybase Inc. All rights reserved.
9      */
10     /**
11     * Holds all the Hybrid Web Container javascript
12     * @namespace
13     */
14     hwc = (typeof hwc === "undefined" || !hwc) ? {} : hwc;    //
SUP 'namespace'
15
16
17     /**
18     * Container API
19     */
20     (function(hwc, undefined) {
21
22     /**
23     * Constant definitions for registration methods
24     */
25     /**
26     * Constant indicating no registration method preference.
The application implementation decides the default method to use.
27     * This is handled as Manual registration by the HWC.
28     * Used in {@link hwc.ConnectionSettings}.
```



```

29      * @type number */
30      hwc.REGISTRATION_METHOD_NO_PREFERENCE = 0;
31      /**
32      * Constant indicating that automatic registration using
password is the preferred method. Used in {@link
hwc.ConnectionSettings}.
33      * @type number */
34      hwc.REGISTRATION_METHOD_AUTOMATIC = 1;
35      /**
36      * Constant indicating that manual registration is the
preferred method. Used in {@link hwc.ConnectionSettings}.
37      * @type number */
38      hwc.REGISTRATION_METHOD_MANUAL = 2;
39      /**
40      * Constant indicating that automatic registration using a
certificate from Afaria is the preferred method. Used in {@link
hwc.ConnectionSettings}.
41      * @type number */
42      hwc.REGISTRATION_METHOD_AFARIA = 3;
43      /**
44      * Constant indicating that automcatic registration using a
local certificate is the preferred method. Used in {@link
hwc.ConnectionSettings}.
45      * @type number */
46      hwc.REGISTRATION_METHOD_CERTIFICATE = 4;
47
48      /**
49      * Represents the connection settings for connecting to the
SUP Server. Used in {@link hwc.loadSettings} and {@link
hwc.saveSettings}.
50      *
51      * @classdesc
52      * @memberOf hwc
53      * @public
54      * @param {number} regmethod A number representing the
registration method (must be one of {@link

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
hwc.REGISTRATION_METHOD_NO_PREFERENCE}, {@link
hwc.REGISTRATION_METHOD_MANUAL},

55      * {@link hwc.REGISTRATION_METHOD_AUTOMATIC}, {@link
hwc.REGISTRATION_METHOD_AFARIA}, {@link
hwc.REGISTRATION_METHOD_CERTIFICATE}).

56      * @param {string} server The SUP/Relay server name.

57      * @param {number} port The SUP/Relay server port
number.

58      * @param {string} server The farm id.

59      * @param {string} user The user name.

60      * @param {string} activationcode The activation code.

61      * @param {string} protocol The protocol to use. Must be
"HTTP" or "HTTPS".

62      * @param {string} password The password for automatic
registration.

63      * @param {string} urlsuffix The url suffix (used only when
connecting to a relay server).

64      * @example

65      * // Create a new ConnectionSettings object.

66      * var connectionSettings = new
hwc.ConnectionSettings( hwc.REGISTRATION_METHOD_MANUAL,

67      *
"999.999.999.999",

68      *
5001,

69      *
0,

70      *
"sampleUsername",

71      *
123,

72      *
"HTTP",

73      *
"samplePassword",

74      *
"/" );

75      * // Use the ConnectionSettings object we just created to
set the connection settings.

76      * hwc.saveSettings( connectionSettings );

77      *

78      */
```

```

79     hwc.ConnectionSettings = function (regmethod, server, port,
farm, user, activatecode, protocol, password, urlsuffix)
80     {
81         this.RegistrationMethod = regmethod;
82         this.ServerName = server;
83         this.Port = port;
84         this.FarmID = farm;
85         this.UserName = user;
86         this.ActivationCode = activatecode;
87         this.Protocol = protocol;
88         this.Password = password;
89         this.UrlSuffix = urlsuffix;
90     };
91
92     /**
93     * Loads the current connection settings from the native
application storage.
94     * @memberOf hwc
95     * @public
96     * @returns {hwc.ConnectionSettings} The connection
settings or null if there are no cached settings.
97     * @example
98     * // Load the connection settings.
99     * var connectionSettings = hwc.loadSettings();
100    */
101    hwc.loadSettings = function () {
102        var settings, response, jsonobj;
103        settings = null;
104
105        hwc.traceEnteringMethod("hwc.loadSettings");
106        try {
107            response =
hwc.getDataFromContainer("loadsettings");
108            jsonobj = JSON.parse(response);

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
109             if (jsonobj !== null && jsonobj !== undefined)
110             {
111                 settings = new
112                 hwc.ConnectionSettings(jsonobj.enableautoregistration,
113                 jsonobj.servername,
114
115                 jsonobj.port, jsonobj.farmid, jsonobj.username,
116
117                 jsonobj.activationcode, jsonobj.protocol, jsonobj.password,
118                 jsonobj.urlsuffix);
119
120             }
121         }catch (ex){
122             hwc.log("loadSettings error:" + ex.message,
123             "ERROR", false);
124         } finally {
125             hwc.traceLeavingMethod("hwc.loadSettings");
126         }
127     }
128     return settings;
129 };
130
131 /**
132  * Constant definitions for device management in add device
133  registration.
134
135  * Some other error numbers may apply for technical
136  support.
137
138  */
139 /**
140  * Constant indicating that MMS Authentication failed.
141  Possible return value for {@link hwc.saveSettings}.
142
143  * @type number */
144 hwc.REG_ERR_MMS_AUTHENTICATION_FAILED =
145 14814;
146 /**
```

```

134      * Constant indicating that the connection to the MMS
service failed. Possible return value for {@link
hwc.saveSettings}.

135      * @type number */

136      hwc.REG_ERR_COULD_NOT_REACH_MMS_SERVER          =
14813;

137      /**

138      * Constant indicating that no MBS template was found for
given AppId and/or Security configuration. Possible return value for
{@link hwc.saveSettings}.

139      * @type number */

140      hwc.REG_ERR_AUTO_REG_TEMPLATE_NOT_FOUND        =
14850;

141      /**

142      * Constant indicating that auto registration was not
enabled in the template. Possible return value for {@link
hwc.saveSettings}.

143      * @type number*/

144      hwc.REG_ERR_AUTO_REG_NOT_ENABLED              =
14851;

145      /**

146      * Constant indicating that the given device id is already
registered for another user. Possible return value for {@link
hwc.saveSettings}.

147      * @type number */

148      hwc.REG_ERR_AUTO_REG_WRONG_USER_FOR_DEVICE    =
14853;

149      /**

150      * Constant indicating that the user name is longer than
the legal limit. Possible return value for {@link
hwc.saveSettings}.

151      * @type number */

152      hwc.REG_ERR_AUTO_REG_USER_NAME_TOO_LONG      =
14854;

153      /**

154      * Constant indicating that the user name contains invalid
characters. Possible return value for {@link hwc.saveSettings}.

155      * @type number */

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
156     hwc.REG_ERR_INVALID_USER_NAME           =
14856;

157     /**

158     * Constant indicating {@link hwc.saveSettings} completed
successfully. Possible return value for {@link hwc.saveSettings}.

159     * @type number */

160     hwc.SETTING_SUCCESS                       = 0;

161

162     /**

163     * Save the connection settings to native application
storage.

164     * Device registration will be attempted if and only the
following conditions are both satisfied.

165     * <ol>

166     * <li> The registration method is not manual. This can be
passed in the hwc.ConnectionSettings object, or if that value is
null, the currently configured value will be used. </li>

167     * <li> The password must be non-empty. This value MUST be
passed in the hwc.ConnectionSettings object. </li>

168     * </ol>

169     * <p>

170     * <b> hwc.startClient() needs to be called after
hwc.saveSettings() for the device to complete automatic/manual
registration. </b>

171     * </p>

172     * <p>

173     * <b> Usage Note: </b> It is not mandatory to specify a
value for each {@link hwc.ConnectionSettings} property. Specifying a
null or undefined for a {@link hwc.ConnectionSettings}

174     * property will effectively cause this method to IGNORE
the property and not change it's value.

175     * </p>

176     * If the saveSettings() operation fails, a non-zero number
will be returned. See hwc.REG_ERR_* for device registration errors.

177     * There can be other types of errors not listed here.

178     *

179     * @public
```

```

180      * @memberOf hwc
181      * @param {hwc.ConnectionSettings} settings The connection
settings to be saved.
182      *
183      * @returns {number} A status code indicating success
({@link hwc.SETTING_SUCCESS}) or an error (one of {@link
hwc.REG_ERR_AUTO_REG_NOT_ENABLED},
184      * {@link hwc.REG_ERR_AUTO_REG_TEMPLATE_NOT_FOUND}, {@link
hwc.REG_ERR_AUTO_REG_USER_NAME_TOO_LONG}, {@link
hwc.REG_ERR_AUTO_REG_WRONG_USER_FOR_DEVICE},
185      * {@link hwc.REG_ERR_COULD_NOT_REACH_MMS_SERVER}, {@link
hwc.REG_ERR_INVALID_USER_NAME}, {@link
hwc.REG_ERR_MMS_AUTHENTICATION_FAILED}).
186      * @example
187      * // Load the connection settings.
188      * var connectionSettings = hwc.loadSettings();
189      * // Modify the connection settings.
190      * connectionSettings.ServerName = "999.999.999.999";
191      * // Save the modified connection settings.
192      * hwc.saveSettings( connectionSettings );
193      * // Start the client to for the device to complete
automatic/manual registration.
194      * hwc.startClient();
195      */
196      hwc.saveSettings = function (settings) {
197          hwc.traceEnteringMethod("hwc.saveSettings");
198          try {
199              // First compose the URL argument string
200              var argumentString, ret;
201              argumentString = "";
202              ret = "";
203
204              if (settings.RegistrationMethod !== null &&
settings.RegistrationMethod !== undefined)
205                  {

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
206         argumentString = "&enableautoregistration=" +
settings.RegistrationMethod;
207     }
208     if (settings.ServerName !== null &&
settings.ServerName !== undefined)
209     {
210         argumentString = argumentString + "&servername=" +
encodeURIComponent(settings.ServerName);
211     }
212     if (settings.Port !== null && settings.Port !==
undefined)
213     {
214         argumentString = argumentString + "&port=" +
settings.Port;
215     }
216     if (settings.FarmID !== null && settings.FarmID !==
undefined)
217     {
218         argumentString = argumentString + "&farmid=" +
encodeURIComponent(settings.FarmID);
219     }
220     if (settings.UserName !== null &&
settings.UserName !== undefined)
221     {
222         argumentString = argumentString + "&username=" +
encodeURIComponent(settings.UserName);
223     }
224     if (settings.ActivationCode !== null &&
settings.ActivationCode !== undefined)
225     {
226         argumentString = argumentString +
"&activationcode=" + encodeURIComponent(settings.ActivationCode);
227     }
228     if (settings.Protocol !== null &&
settings.Protocol !== undefined)
229     {
```



```
230         argumentString = argumentString + "&protocol=" +
encodeURIComponent(settings.Protocol);
231     }
232     if (settings.Password !== null &&
settings.Password !== undefined)
233     {
234         argumentString = argumentString + "&password=" +
encodeURIComponent(settings.Password);
235     }
236     if (settings.UrlSuffix !== null &&
settings.UrlSuffix !== undefined)
237     {
238         argumentString = argumentString + "&urlsuffix=" +
encodeURIComponent(settings.UrlSuffix);
239     }
240
241     // Only invoke the native function if we're saving at
least one setting
242     if (argumentString !== "")
243     {
244         ret = hwc.getDataFromContainer("saveSettings",
argumentString);
245         return parseInt(ret, 10);
246     }
247     else
248     {
249         return hwc.SETTING_SUCCESS;
250     }
251     } catch (ex){
252         hwc.log("saveSettings error:" + ex.message, "ERROR",
false);
253         throw ex;
254     } finally {
255         hwc.traceLeavingMethod("hwc.saveSettings");
256     }
```

```
257     };
258
259
260     /**
261      * Start of connection state listener callback functions
262      */
263     /**
264      * An array of {@link anonymous.ConnectionStateListener}
265      * callback functions.
266      * @type Array
267      * @private
268      */
269     hwc._connectionListeners = [];
270
271     /**
272      * An array of objects containing {@link
273      * anonymous.ConnectionStateListener} callback functions.
274      * The containing objects need to be kept track of since the
275      * callback functions may reference
276      * variables in the containing object.
277      * @type Array
278      * @private
279      */
280     hwc._connectionListenerContainingObjects = [];
281
282     /**
283      * This is the main entry of connection event notification.
284      * The native code
285      * calls this function internally
286      * @private
287      * @param {number} event A flag indicating the current
288      * connection state (will be either {@link hwc.CONNECTED} or {@link
289      * hwc.DISCONNECTED}).
290      * @param {number} errorCode An error code. Will be 0 if
291      * there is no error.
```

```
284      * @param {string} errorMessage Text of an error message.  
Will be the empty string if there is no error.  
285      */  
286      hwc.connectionListenerNotification = function (event,  
errorCode, errorMessage)  
287      {  
288          var i, containingObject;  
289          hwc.traceEnteringMethod("hwc.connectionListenerNotification");  
290          try {  
291              if (hwc._connectionListeners.length === 0) {  
292                  return;  
293              }  
294  
295              for (i = 0; i < hwc._connectionListeners.length; i  
++)  
296              {  
297                  containingObject =  
hwc._connectionListenerContainingObjects[i];  
298                  if (containingObject !== null &&  
containingObject !== undefined)  
299                  {  
300                      hwc._connectionListeners[i].call(containingObject, event, errorCode,  
errorMessage);  
301                  }  
302                  else  
303                  {  
304                      hwc._connectionListeners[i](event, errorCode,  
errorMessage);  
305                  }  
306              }  
307          } finally {  
308              hwc.traceLeavingMethod("hwc.connectionListenerNotification");  
309          }
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
310     };
311
312     /**
313     * Register the connection state listener.
314     *
315     * @public
316     * @memberOf hwc
317     * @param {anonymous.ConnectionStateListener}
318     ConnectionStateListener Callback for connection state changes.
319     * @param {Object} [containingObject] Object containing
320     definition for ConnectionStateListener. If a connection state
321     callback function
322     * references variables in its containing object, then the
323     containing object should be passed to this function.
324     * @example
325     * // doSomething is a global function that gets called
326     from the connection listener.
327     * var doSomething = function()
328     * {
329     *     alert("sample function that gets executed when the hwc
330     becomes connected");
331     * }
332     * // connectionListener is the callback function that is
333     given to addConnectionListener.
334     * // When there is a connection event, connectionListener
335     will be invoked with the details.
336     * var connectionListener = function( event, errorCode,
337     errorMessage )
338     * {
339     *     if( event == hwc.CONNECTED )
340     *     {
341     *         doSomething();
342     *     }
343     * }
344     * hwc.addConnectionListener( connectionListener );
345     *
```

```
337      * @example
338      * // connectionStateManager is an object that will contain
the connection listener callback as well as
339      * // a variable used by the callback.
340      * var connectionStateManager = {};
341      * // The connectionStateManager keeps track of whether the
HWC is connected or not.
342      * connectionStateManager.connected = false;
343      * // A function called by the listener.
344      * connectionStateManager.doSomething = function()
345      * {
346      *     if( this.connected )
347      *     {
348      *         alert("this alert gets displayed if the hwc is
connected");
349      *     }
350      * }
351      * // This is the callback function that will be passed to
addConnectionListener. This callback references variables
352      * // from the containing object (this.connected and
this.doSomething), so when we call addConnectionListener we have
353      * // to give the containing object as the second
parameter.
354      * connectionStateManager.listener = function( event,
errorCode, errorMessage )
355      * {
356      *     if( event == hwc.CONNECTED )
357      *     {
358      *         this.connected = true;
359      *     }
360      *     else
361      *     {
362      *         this.connected = false;
363      *     }
364      *     this.doSomething();
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
365     * }
366     * // Pass both the listener and the containing object.
367     * This enables the listener to refer to variables in the containing
368     * object when it is invoked.
369     *
370     * hwc.addConnectionListener( connectionStateManager.listener,
371     * connectionStateManager );
372     */
373     hwc.addConnectionListener = function
374     (ConnectionStateListener, containingObject)
375     {
376         hwc.traceEnteringMethod("hwc.addConnectionListener");
377         try {
378             hwc._connectionListeners.push(ConnectionStateListener);
379             hwc._connectionListenerContainingObjects.push(containingObject);
380             if (hwc._connectionListeners.length === 1)
381             {
382                 hwc.getDataFromContainer("startconnectionlistener");
383             }
384         } finally {
385             hwc.traceLeavingMethod("hwc.addConnectionListener");
386         }
387     };
388
389     /**
390     * Remove the connection state listener. This function
391     * should be called with identical parameters that were used
392     * when adding the connection state listener with {@link
393     * hwc.addConnectionListener}.
394     *
395     * @public
396     * @memberOf hwc
```

```

390      * @param {anonymous.ConnectionStateListener}
ConnectionStateListener Callback function with connection state
changes

391      * @param {Object} [containingObject] Optional Object
containing definition of ConnectionStateListener

392      * @example

393      * // doSomething is a global function that gets called
from the connection listener.

394      * var doSomething = function()

395      * {

396      *     alert("sample function that gets executed when the hwc
becomes connected");

397      * }

398      * // connectionListener is the callback function that is
given to addConnectionListener.

399      * // When there is a connection event, connectionListener
will be invoked with the details.

400      * var connectionListener = function( event, errorCode,
errorMessage )

401      * {

402      *     if( event == hwc.CONNECTED )

403      *     {

404      *         doSomething();

405      *     }

406      * }

407      * hwc.addConnectionListener( connectionListener );

408      * // At some other point if we want to remove the listener,
we use the following line:

409      * hwc.removeConnectionListener( connectionListener );

410      *

411      * @example

412      * // connectionStateManager is an object that will contain
the connection listener callback as well as

413      * // a variable used by the callback.

414      * var connectionStateManager = {};

415      * // The connectionStateManager keeps track of whether the
HWC is connected or not.

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
416     * connectionStateManager.connected = false;
417     * // A function called by the listener.
418     * connectionStateManager.doSomething = function()
419     * {
420     *     if( this.connected )
421     *     {
422     *         alert("this alert gets displayed if the hwc is
connected");
423     *     }
424     * }
425     * // This is the callback function that will be passed to
addConnectionListener. This callback references variables
426     * // from the containing object (this.connected and
this.doSomething), so when we call addConnectionListener we have
427     * // to give the containing object as the second
parameter.
428     * connectionStateManager.listener = function( event,
errorCode, errorMessage )
429     * {
430     *     if( event == hwc.CONNECTED )
431     *     {
432     *         this.connected = true;
433     *     }
434     *     else
435     *     {
436     *         this.connected = false;
437     *     }
438     *     this.doSomething();
439     * }
440     * // Pass both the listener and the containing object.
This enables the listener to refer to variables in the containing
object when it is invoked.
441     *
hwc.addConnectionListener( connectionStateManager.listener,
connectionStateManager );
```



```
442      * // At some other point if we want to remove the listener,
we use the following line:
443      *
hwc.removeConnectionListener( connectionStateManager.listener,
connectionStateManager );
444      */
445      hwc.removeConnectionListener = function
(ConnectionStateListener, containingObject)
446      {
447          var i;
448      hwc.traceEnteringMethod("hwc.removeConnectionListener");
449          try {
450              if (hwc._connectionListeners.length === 0) {
451                  return;
452              }
453
454              for (i = 0; i < hwc._connectionListeners.length; i
++)
455              {
456                  if (hwc._connectionListeners[i] ===
ConnectionStateListener &&
457                      hwc._connectionListenerContainingObjects[i]
=== containingObject)
458                  {
459                      hwc._connectionListeners.splice(i, 1);
460                      hwc._connectionListenerContainingObjects.splice(i, 1);
461                      if (hwc._connectionListeners.length === 0)
462                      {
463                          hwc.getDataFromContainer("stopconnectionlistener");
464                      }
465                      return;
466                  }
467              }
```

```
468         } finally {
469
470             hwc.traceLeavingMethod("hwc.removeConnectionListener");
471         }
472     };
473
474     /**
475      * A sample {@link anonymous.ConnectionStateListener}
476      * callback function.
477      *
478      * @param {number} event A number indicating the event that
479      * occurred (will be {@link hwc.CONNECTED} or {@link
480      * hwc.DISCONNECTED}).
481      *
482      * @param {number} errorCode An error code (0 indicating
483      * success).
484      *
485      * @param {string} errorMessage Text of the error message.
486      * Will be empty if there is no error.
487      */
488     hwc.sample_ConnectionListener = function (event,
489     errorCode, errorMessage) {
490
491         switch (event)
492         {
493             case hwc.CONNECTED:
494                 alert('Connected event');
495                 break;
496             case hwc.DISCONNECTED:
497                 alert('Disconnected event');
498                 break;
499         }
500
501         if (errorCode !== null && errorMessage !== null)
502         {
503             alert('Connection error\n' +
504                 'Code: ' + errorCode + '\n' +
505                 'Message: ' + errorMessage);
506         }
507     };
508 }
```

```
496     }
497   };
498
499   /**
500    * Constant indicating that the hwc is connected. Used in
501    * {@link anonymous.ConnectionStateListener} callback functions.
502    * @type number
503    */
504   hwc.CONNECTED = 1;
505
506   /**
507    * Constant indicating that the hwc is disconnected. Used
508    * in {@link anonymous.ConnectionStateListener} callback functions.
509    * @type number
510    */
511   hwc.DISCONNECTED = 2;
512
513   /**
514    * Start the client connection to the SUP server.
515    * Companion function to {@link hwc.shutdown}.
516    * If a hybrid app is running in the context of the Hybrid
517    * Web Container
518    * then it will probably never have to call this function
519    * unless {@link hwc.shutdown} client was called first.
520    * @public
521    * @memberOf hwc
522    * @param {anonymous.LogListener} [onNotification] A log
523    * listener callback function. If you are interested in
524    * the connection state it is recommended that you call
525    * {@link hwc.addConnectionListener} before calling hwc.startClient.
526    * @example
527    * hwc.startClient();
528    *
529    * @example
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
524      * // Add a log listener while calling hwc.startClient.
525      * var logListener = function( time, event, message )
526      * {
527      *     alert(message);
528      * }
529      * hwc.startClient( logListener );
530      */
531      hwc.startClient = function (onNotification) {
532          hwc.traceEnteringMethod("hwc.startClient");
533          try {
534              if (hwc._defaultLogListener !== null &&
535                  hwc._defaultLogListener !== undefined)
536                  {
537                      hwc.removeLogListener(hwc._defaultLogListener,
538                          null);
539                      hwc._defaultLogListener = null;
540                  }
541              if (onNotification !== null && onNotification !==
542                  undefined)
543                  {
544                      hwc.addLogListener( onNotification, null );
545                      hwc._defaultLogListener = onNotification;
546                  }
547              hwc.getDataFromContainer( "startclient" );
548              return 0;
549          } catch (ex){
550              hwc.log("startClient error:" + ex.message, "ERROR",
551                  false);
552          } finally {
553              hwc.traceLeavingMethod("hwc.startClient");
554          }
555      };
```

```
554
555     /**
556     * Shutdown the client connection to the SUP server.
557     * Companion function to {@link hwc.startClient}.
558     * If a hybrid app is running in the context of the Hybrid
559     * Web Container, then it will probably never have to call
560     * this function. If you want to temporarily stop the
561     * connection, then call {@link hwc.disconnectFromServer} instead.
562     * @public
563     * @memberOf hwc
564     * @example
565     * hwc.shutdown();
566     */
567     hwc.shutdown = function () {
568         hwc.traceEnteringMethod("hwc.shutdown");
569         try {
570             hwc.getDataFromContainer("shutdownclient");
571             if (hwc._defaultLogListener !== null &&
572                 hwc._defaultLogListener !== undefined)
573             {
574                 hwc.removeLogListener(hwc._defaultLogListener,
575                     null);
576                 hwc._defaultLogListener = null;
577             }
578             } catch (ex){
579                 hwc.log("shutdown error:" + ex.message, "ERROR",
580                     false);
581             } finally {
582                 hwc.traceLeavingMethod("hwc.shutdown");
583             }
584         };
585     /**
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
582      * Resumes the connection to the SUP server. Companion
function to {@link hwc.disconnectFromServer}. This function should
583      * only be called after the connection to the SUP server has
been suspended with a call to {@link hwc.disconnectFromServer}.
584      *
585      * @public
586      * @memberOf hwc
587      * @param {anonymous.LogListener} [onNotification] A log
listener callback function. If you are interested in
588      * the connection state it is recommended that you call
{@link hwc.addConnectionListener} before calling
hwc.connectToServer.
589      *
590      * @example
591      * hwc.connectToServer();
592      *
593      * @example
594      * // Add a log listener while calling
hwc.connectToServer.
595      * var logListener = function( time, event, message )
596      * {
597      *     alert(message);
598      * }
599      * hwc.connectToServer( logListener );
600      */
601      hwc.connectToServer = function (onNotification) {
602          hwc.traceEnteringMethod("hwc.connectToServer");
603          try {
604              if (hwc._defaultLogListener !== null &&
hwc._defaultLogListener !== undefined)
605              {
606                  hwc.removeLogListener(hwc._defaultLogListener,
null);
607                  hwc._defaultLogListener = null;
608              }
```

```
609
610         if (onNotification !== null && onNotification !==
undefined)
611         {
612             hwc.addLogListener( onNotification, null );
613             hwc._defaultLogListener = onNotification;
614         }
615
616         hwc.getDataFromContainer( "connecttoserver" );
617
618         return 0;
619     } catch (ex){
620         hwc.log("connectToServer error:" + ex.message,
"ERROR", false);
621         throw ex;
622     } finally {
623         hwc.traceLeavingMethod("hwc.connectToServer");
624     }
625 };
626
627 /**
628  * This is the default one to keep the listener added in the
connectionToServer call.
629  * @private
630  */
631     hwc._defaultLogListener = null;
632
633 /**
634  * Suspends the connection to the SUP server. Companion
function to {@link hwc.connectToServer}.
635  * @public
636  * @memberOf hwc
637  * @example
638  * hwc.disconnectFromServer();
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
639     */
640     hwc.disconnectFromServer = function () {
641         hwc.traceEnteringMethod("hwc.disconnectFromServer");
642         try {
643             hwc.getDataFromContainer("disconnectfromserver");
644
645             if (hwc._defaultLogListener !== null &&
hwc._defaultLogListener !== undefined)
646                 {
647                     hwc.removeLogListener(hwc._defaultLogListener,
null);
648                     hwc._defaultLogListener = null;
649                 }
650
651             } catch (ex){
652                 hwc.log("disconnectFromServer error:" + ex.message,
"ERROR", false);
653             } finally {
654 hwc.traceLeavingMethod("hwc.disconnectFromServer");
655             }
656         };
657
658     /**
659     * Start of connection functions
660     */
661
662     /**
663     * An array of log listener callback functions.
664     * @type Array
665     * @private
666     */
667     hwc._logListeners = [];
668     /**
```



```

669      * An array of objects containing log listener callback
        functions. The containing objects
670      * need to be kept track of because the callback functions
        may reference variables in the
671      * containing object.
672      * @type Array
673      * @private
674      */
675      hwc._logListenerContainingObjects = [];
676
677      /**
678      * This is the main entry of log event notification. The
        native code
679      * calls this function internally.
680      *
681      * @param {number} milliseconds The date of the log message
        represented in milliseconds.
682      * @param {number} event The that represents which category
        this event falls under (It will be one of hwc.CONNECTION_*
        constants).
683      * @param {string} optionalString The string carrying the
        message of the log event.
684      * @private
685      */
686      hwc._logListenerNotification = function ( milliseconds,
        event, optionalString )
687      {
688          var date, i, containingObject;
689          hwc.traceEnteringMethod("hwc._logListenerNotification");
690          try {
691              if (hwc._logListeners.length === 0) {
692                  return;
693              }
694

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
695         // The incoming date is number of millisecond, we
        need to change it to real JavaScript Date type.
696         date = new Date(milliseconds);
697
698         for (i = 0; i < hwc._logListeners.length; i++)
699         {
700             containingObject =
hwc._logListenerContainingObjects[i];
701             if (containingObject !== null &&
containingObject !== undefined)
702             {
703                 hwc._logListeners[i].call(containingObject,
date, event, optionalString);
704             }
705             else
706             {
707                 hwc._logListeners[i](date, event,
optionalString);
708             }
709         }
710     } finally {
711         hwc.traceLeavingMethod("hwc._logListenerNotification");
712     }
713 };
714
715 /**
716  * Register the log listener.
717  * @public
718  * @memberOf hwc
719  * @param {anonymous.LogListener} LogListener Callback for
changes to the log.
720  * @param {Object} [containingObject] Object containing
definition for LogListener. If a log listener callback function
721  * references variables in its containing object, then the
containing object should be passed to this function.
```

```
722      *
723      * @example
724      * // A global function called by the log listener.
725      * var doSomething = function()
726      * {
727      *     alert("this gets displays when there is a log
728      * event.");
729      * }
730      * // The log listener callback function that will be
731      * passed to hwc.addLogListener.
732      * // This function will be invoked whenever there is a log
733      * event.
734      * var logListener = function( event, errorCode,
735      * errorMessage )
736      * {
737      *     doSomething();
738      * }
739      * // Add the log listener.
740      * hwc.addLogListener( logListener );
741      *
742      * @example
743      * // logListenerManager is an object that will contain the
744      * listener callback as well
745      * // as a function that will be invoked from the listener
746      * callback function.
747      * var logListenerManager = {};
748      * // This is a function that is called from the listener
749      * callback.
750      * logListenerManager.doSomething = function()
751      * {
752      *     alert("this gets displays when there is a log
753      * event.");
754      * }
755      * // This is the listener callback that will be passed to
756      * hwc.addLogListener.
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
748      * // Since a variable is referenced from the containing
object, the containing object
749      * // will need to be passed to hwc.addLogListener.
750      * logListenerManager.listener = function( event,
errorCode, errorMessage )
751      * {
752      *     this.doSomething();
753      * }
754      * // Pass both the listener callback and the containing
object.
755      * hwc.addLogListener( logListenerManager.listener,
logListenerManager );
756      */
757      hwc.addLogListener = function ( LogListener,
containingObject)
758      {
759          hwc.traceEnteringMethod("hwc.addLogListener");
760          try {
761              hwc._logListeners.push(LogListener);
762              hwc._logListenerContainingObjects.push(containingObject);
763              if (hwc._logListeners.length === 1)
764              {
765                  hwc.getDataFromContainer("startloglistener");
766              }
767          } finally {
768              hwc.traceLeavingMethod("hwc.addLogListener");
769          }
770      };
771
772      /**
773      * Remove the log listener. This function should be called
with identical parameters that were used
774      * when adding the log listener with {@link
hwc.addLogListener}.
```

```
775      *
776      * @public
777      * @memberOf hwc
778      * @param {anonymous.LogListener} LogListener The callback
function for log events.
779      * @param {Object} [containingObject] Object containing
definition of ConnectionStateListener
780      * @example
781      * // A global function called by the log listener.
782      * var doSomething = function()
783      * {
784      *     alert("this gets displays when there is a log
event.");
785      * }
786      * // The log listener callback function that will be
passed to hwc.addLogListener.
787      * // This function will be invoked whenever there is a log
event.
788      * var logListener = function( event, errorCode,
errorMessage )
789      * {
790      *     doSomething();
791      * }
792      * // Add the log listener.
793      * hwc.addLogListener( logListener );
794      * // at some other point if we want to remove the listener,
we use the following line
795      * hwc.removeLogListener( logListener );
796      *
797      * @example
798      * // logListenerManager is an object that will contain the
listener callback as well
799      * // as a function that will be invoked from the listener
callback function.
800      * var logListenerManager = {};
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
801      * // This is a function that is called from the listener
callback.
802      * logListenerManager.doSomething = function()
803      * {
804      *     alert("this gets displays when there is a log
event.");
805      * }
806      * // This is the listener callback that will be passed to
hwc.addLogListener.
807      * // Since a variable is referenced from the containing
object, the containing object
808      * // will need to be passed to hwc.addLogListener.
809      * logListenerManager.listener = function( event,
errorCode, errorMessage )
810      * {
811      *     this.doSomething();
812      * }
813      * // Pass both the listener callback and the containing
object.
814      * hwc.addLogListener( logListenerManager.listener,
logListenerManager );
815      * // at some other point if we want to remove the listener,
we use the following line
816      * hwc.removeLogListener( logListenerManager.listener,
logListenerManager );
817      */
818      hwc.removeLogListener = function (LogListener,
containingObject)
819      {
820          var i;
821          hwc.traceEnteringMethod("hwc.removeLogListener");
822          try {
823              if (hwc._logListeners.length === 0){
824                  return;
825              }
826
```

```
827         for (i = 0; i < hwc._logListeners.length; i++)
828         {
829             if (hwc._logListeners[i] === LogListener &&
830                 hwc._logListenerContainingObjects[i] ===
831                 containingObject)
832             {
833                 hwc._logListeners.splice(i, 1);
834                 hwc._logListenerContainingObjects.splice(i,
835                 1);
836             }
837             if (hwc._logListeners.length === 0)
838             {
839                 hwc.getDataFromContainer("stoploglistener");
840             }
841             return;
842         } finally {
843             hwc.traceLeavingMethod("hwc.removeLogListener");
844         }
845     };
846
847     /**
848     * Sample {@link anonymous.LogListener} callback
849     * function.
850     *
851     * @param {number} milliseconds The date of the log message
852     *     represented in milliseconds.
853     *
854     * @param {number} event The that represents which category
855     *     this event falls under (It will be one of {@link
856     *     hwc.CONNECTION_ERROR},
857     *
858     * @param {@link hwc.CONNECTION_OTHER}, {@link
859     *     hwc.CONNECTION_CONNECTED}, {@link hwc.CONNECTION_DISCONNECTED},
860     *     {@link hwc.CONNECTION_RETRIEVED_ITEMS}).
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
853      * @param {string} optionalString The string carrying the
message of the log event.
854      */
855      hwc.sample_LogListener = function ( date, event,
optionalString ) {
856      };
857
858      // Connection event definitions
859      /**
860      * A constant indicating that the log message is about a
connection error. Used in {@link anonymous.LogListener} callback
functions.
861      * @type number
862      */
863      hwc.CONNECTION_ERROR = -1;
864      /**
865      * A constant indicating that the log message is not about
the connection. Used in {@link anonymous.LogListener} callback
functions.
866      * @type number
867      */
868      hwc.CONNECTION_OTHER = 0;
869      /**
870      * A constant indicating that the log message is about the
connection being established. Used in {@link anonymous.LogListener}
callback functions.
871      * @type number
872      */
873      hwc.CONNECTION_CONNECTED = 1;
874      /**
875      * A constant indicating that the log message is about the
connection being disconnected. Used in {@link
anonymous.LogListener} callback functions.
876      * @type number
877      */
878      hwc.CONNECTION_DISCONNECTED = 2;
```



```
879      /**
880       * a constant indicating that the log message is about
retrieved items. Used in {@link anonymous.LogListener} callback
functions.
881       * @type number
882       */
883       hwc.CONNECTION_RETRIEVED_ITEMS = 3;
884
885
886      /**
887       * Start of hybrid app installation callback functions
888       */
889
890      /**
891       * An array of app installation listeners
892       * @private
893       * @type Array
894       */
895       hwc._appInstallationListeners = [];
896
897      /**
898       * This is the main entry of installation notification. The
native code should be
899       * hardcoded to call this function internally.
900       *
901       * @private
902       * @param {number} event A constant indicating whether the
app installation is beginning or has just ended
903       * (will be either {@link hwc.INSTALLATION_BEGIN} or {@link
hwc.INSTALLATION_END}).
904       * @param {number} moduleId The module ID of the hybrid app
being installed.
905       * @param {number} version The version of the hybrid app
being installed.
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
906      * @param {string} moduleName The display name of the
hybrid app being installed.
907      */
908      hwc.appInstallationListenerNotification = function (event,
moduleId, version, moduleName)
909      {
910          var i;
911          hwc.traceEnteringMethod("hwc.appInstallationListenerNotification");
912          try {
913              if (hwc._appInstallationListeners.length === 0) {
914                  return;
915              }
916
917              for (i = 0; i < hwc._appInstallationListeners.length;
i++)
918              {
919                  hwc._appInstallationListeners[i](event, moduleId,
version, moduleName);
920              }
921          } finally {
922              hwc.traceLeavingMethod("hwc.appInstallationListenerNotification");
923          }
924      };
925
926      /**
927      * Register the application installation listener.
928      *
929      * @param {anonymous.AppInstallationListener}
AppInstallationListener A callback for application installation
changes.
930      *
931      * @example
932      * // appInstallListener is the callback function that will
be passed to hwc.addAppInstallationListener.
```

```
933     * var appInstallListener = function( event, moduleId,
version, moduleName )
934     * {
935     *     if( event == hwc.INSTALLATION_BEGIN )
936     *     {
937     *         alert(moduleName + " has just started the
installation process.");
938     *     }
939     *     else if( event == hwc.INSTALLATION_END )
940     *     {
941     *         alert(moduleName + " has just finished the
installation process.");
942     *     }
943     * }
944     * hwc.addAppInstallationListener( appInstallListener );
945     */
946     hwc.addAppInstallationListener = function
(AppInstallationListener)
947     {
948     hwc.traceEnteringMethod("hwc.addAppInstallationListener");
949         try {
950     hwc._appInstallationListeners.push(AppInstallationListener);
951         if(hwc._appInstallationListeners.length === 1)
952         {
953     hwc.getDataFromContainer("startAppInstallationListener");
954         }
955         } finally {
956     hwc.traceLeavingMethod("hwc.addAppInstallationListener");
957         }
958     };
959
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
960      /**
961      * Remove the application installation listener. This
function should be called with identical parameters
962      * that were used to add the application installation
listener with {@link hwc.addAppInstallationListener}.
963      *
964      * @public
965      * @memberOf hwc
966      * @param {anonymous.AppInstallationListener}
AppInstallationListener The callback for application installation
changes.
967      * @example
968      * // appInstallListener is the callback function that will
be passed to hwc.addAppInstallationListener.
969      * var appInstallListener = function( event, moduleId,
version, moduleName )
970      * {
971      *     if( event == hwc.INSTALLATION_BEGIN )
972      *     {
973      *         alert(moduleName + " has just started the
installation process.");
974      *     }
975      *     else if( event == hwc.INSTALLATION_END )
976      *     {
977      *         alert(moduleName + " has just finished the
installation process.");
978      *     }
979      * }
980      * hwc.addAppInstallationListener( appInstallListener );
981      * // when we want to remove this listener, we call the
following line:
982      *
hwc.removeAppInstallationListener( appInstallListener );
983      */
984      hwc.removeAppInstallationListener = function
(AppInstallationListener)
985      {
```

```
986         var i;
987 hwc.traceEnteringMethod("hwc.removeAppInstallationListener");
988         try {
989             if (hwc._appInstallationListeners.length === 0) {
990                 return;
991             }
992
993             for (i = 0; i < hwc._appInstallationListeners.length;
994 i++)
995                 {
996                     if (hwc._appInstallationListeners[i] ===
997 AppInstallationListener)
998                         {
999                             hwc._appInstallationListeners.splice(i, 1);
1000                             break;
1001                         }
1002
1003                     if (hwc._appInstallationListeners.length === 0)
1004                         {
1005                             hwc.getDataFromContainer("stopAppInstallationListener");
1006                         }
1007                 } finally {
1008                     hwc.traceLeavingMethod("hwc.removeAppInstallationListener");
1009                 }
1010             };
1011
1012             /**
1013             * Sample application listener callback function
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1014      * @param {Integer} event          Installation flags
including, BEGIN(1), END(2), FAIL(3)

1015      * @param {String} moduleId        Optional Module Id

1016      * @param {String} version        Optional Module
version

1017      * @param {String} moduleName    Optional Module display
name

1018      * @param {String} designerVersion Optional Version of
designer used to create app

1019      * @param {String} containerVersion Optional Version of
hybrid web container

1020      */

1021      hwc.sample_InstallationAppListener = function (event,
moduleId, version, moduleName, designerVersion, containerVersion) {

1022      };

1023

1024      // Installation event definitions

1025      /**

1026      * A constant indicating that the application is starting
to be installed. Used in {@link anonymous.AppInstallationListener}
callback functions.

1027      * @type number

1028      */

1029      hwc.INSTALLATION_BEGIN = 1;

1030      /**

1031      * A constant indicating that the application has finished
being installed. Used in {@link anonymous.AppInstallationListener}
callback functions.

1032      * @type number

1033      */

1034      hwc.INSTALLATION_END = 2;

1035      hwc.INSTALLATION_FAIL = 3;

1036

1037      /**

1038      * Call this function to get an array of {@link
hwc.LogEntry} objects. There will be one
```

```
1039      * {@link hwc.LogEntry} object for each line in the HWC
log.
1040      *
1041      * @public
1042      * @memberOf hwc
1043      * @returns {hwc.LogEntry[]} An array of hwc.LogEntry
objects.
1044      * @example
1045      * var log = hwc.getLogEntries();
1046      */
1047      hwc.getLogEntries = function () {
1048          var response, logEntries, i, entries, entry;
1049
1050          hwc.traceEnteringMethod("hwc.getLogEntries");
1051          response = "";
1052          logEntries = [];
1053
1054          try {
1055              response =
hwc.getDataFromContainer("getlogentries");
1056
1057              if (response !== null && response !== undefined &&
response !== "")
1058                  {
1059                      entries = JSON.parse(response);
1060                      for (i=0; i<entries.length; i++) {
1061                          entry = entries[i];
1062                          logEntries[i] = new hwc.LogEntry(new
Date(entry.milliseconds), entry.event, entry.message);
1063                      }
1064                  }
1065          } catch (ex){
1066              hwc.log("getLogEntries error:" + ex.message, "ERROR",
false);
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1067         } finally {
1068             hwc.traceLeavingMethod("hwc.getLogEntries");
1069         }
1070
1071         return logEntries;
1072     };
1073
1074     /**
1075     * This object represents a log entry.
1076     * @classdesc
1077     * @public
1078     * @memberOf hwc
1079     * @param {number} date The date the log entry was
1080     recorded, in milliseconds since January 1, 1970, 00:00:00 GMT
1081     * @param {number} event The event ID of the log entry
1082 (will be one of {@link hwc.CONNECTION_ERROR}, {@link
1083 hwc.CONNECTION_OTHER},
1084     * {@link hwc.CONNECTION_CONNECTED}, {@link
1085 hwc.CONNECTION_DISCONNECTED}, {@link
1086 hwc.CONNECTION_RETRIEVED_ITEMS})
1087     * @param {string} msg The message of the log entry.
1088     */
1089     hwc.LogEntry = function (date, event, msg)
1090     {
1091         this.logdate = date;
1092         this.eventID = event;
1093         this.message = msg;
1094     }
1095
1096     /**
1097     * Gets the date of the log entry.
1098     * @public
1099     * @memberOf hwc.LogEntry
1100     * @returns {number} The date the log entry was created
1101 in the HWC, in milliseconds.
```



```
1095     */
1096     this.getDate = function ()
1097     {
1098         return this.logdate;
1099     };
1100
1101     /**
1102     * Gets the event ID of the log entry to see what this
1103     log entry is about.
1104     * @public
1105     * @memberOf hwc.LogEntry
1106     * @returns {number} A constant indication what this log
1107     entry is about (will be one of {@link hwc.CONNECTION_ERROR}, {@link
1108     hwc.CONNECTION_OTHER},
1109     * {@link hwc.CONNECTION_CONNECTED}, {@link
1110     hwc.CONNECTION_DISCONNECTED}, {@link
1111     hwc.CONNECTION_RETRIEVED_ITEMS}).
1112     */
1113     this.getEventID = function ()
1114     {
1115         return this.eventID;
1116     };
1117
1118     /**
1119     * Gets the message text of the log entry.
1120     * @public
1121     * @memberOf hwc.LogEntry
1122     * @returns {string} The message text of the log
1123     entry.
1124     */
1125     this.getMessage = function ()
1126     {
1127         return this.message;
1128     };
1129 }
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1123     };
1124
1125     /**
1126      * An array of push notification listeners
1127      * @private
1128      * @type Array
1129      */
1130     hwc._pushnotificationlisteners = [];
1131     /**
1132      * An array of objects containing push notification
1133      * listeners
1134      * @private
1135      * @type Array
1136      */
1137     hwc._pushnotificationlistenerContainingObjects = [];
1138     /**
1139      * This is the main entry of push notification. The native
1140      * code
1141      * calls this function internally.
1142      * @private
1143      * @param {string} jsonString      The notifications in JSON
1144      * encoding
1145      * @param {Integer} [id]          ID of the communication
1146      * area if required
1147      */
1148     hwc._pushnotificationListenerNotification =
1149     function(jsonString, id)
1150     {
1151         var ret, i, notifications, containingObject;
```

```
1151         ret = hwc.NOTIFICATION_CONTINUE;
1152     try
1153     {
1154         if (hwc._pushnotificationlisteners.length > 0)
1155         {
1156             notifications = JSON.parse(jsonString);
1157             // We must have a valid push data to
1158             continue
1159             if (!(notifications === null ||
1160                 notifications === undefined || notifications.length === 0))
1161             {
1162                 for (i = 0; i <
1163                     hwc._pushnotificationlisteners.length; i++)
1164                 {
1165                     try
1166                     {
1167                         ret =
1168                         hwc.NOTIFICATION_CONTINUE; // default status
1169
1170                         containingObject =
1171                         hwc._pushnotificationlistenerContainingObjects[i];
1172                         if (containingObject !== null &&
1173                             containingObject !== undefined)
1174                         {
1175                             ret =
1176                             hwc._pushnotificationlisteners[i].call(containingObject,
1177                                 notifications);
1178                         }
1179                     }
1180                     else
1181                     {
1182                         ret =
1183                         hwc._pushnotificationlisteners[i](notifications);
1184                     }
1185                 }
1186             }
1187         }
1188     }
1189     // If the return status is
1190     hwc.NOTIFICATION_CANCEL, we need to return immediately.
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1176                                     if (ret ===
hwc.NOTIFICATION_CANCEL) {
1177                                     break;
1178                                     }
1179                                     }
1180                                     catch (ex)
1181                                     {
1182                                     // Don't pop alert here because it
will block the whole process of notifications
1183                                     }
1184                                     } //for
1185                                     } //if
1186                                     } //if
1187                                     }
1188                                     catch (ex1)
1189                                     {
1190                                     // Don't pop alert here because it will block
the whole process of notifications
1191                                     }
1192                                     if (hwc.isBlackBerry() || hwc.isIOS() )
1193                                     {
1194                                     return ret;
1195                                     }
1196                                     else
1197                                     {
1198                                     hwc.getDataFromContainer("jsmethodreturn",
"&id=" + id + "&jsreturnvalue=" + ret);
1199                                     }
1200
1201                                     } finally {
1202                                     hwc.traceLeavingMethod("hwc._pushnotificationListenerNotification")
;
1203                                     }
```

```
1204     };
1205
1206     /**
1207     * Register a push notification listener.
1208     *
1209     * @public
1210     * @memberOf hwc
1211     * @param {function} PushNotificationListener The callback
1212     * for push notifications.
1213     * @param {Object} [containingObject] Object containing
1214     * definition for PushNotificationListener. If the listener callback
1215     * function
1216     * references variables in its containing object, then the
1217     * containing object should be passed to this function.
1218     * @example
1219     * // pushListener is the callback function that will be
1220     * passed to hwc.addPushNotificationListener.
1221     * var pushListener = function( notifications )
1222     * {
1223     *     alert( "push notification:\n" +
1224     *     JSON.stringify(notifications) );
1225     *     return hwc.NOTIFICATION_CONTINUE;
1226     * }
1227     * hwc.addPushNotificationListener( pushListener );
1228     *
1229     * @example
1230     * // pushListenerManager is an object that will contain
1231     * the listener callback as well as a variable
1232     * // referenced from the callback.
1233     * var pushListenerManager = {};
1234     * // doSomething is a function that is called from inside
1235     * the callback.
1236     * pushListenerManager.doSomething =
1237     * function( notifications )
1238     * {
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1230      *      alert( "push notification:\n" +
JSON.stringify(notifications) );
1231      *      return hwc.NOTIFICATION_CONTINUE;
1232      *  }
1233      *  // This is the callback function.
1234      *  pushListenerManager.listener =
function( notifications )
1235      *  {
1236      *      return this.doSomething( notifications );
1237      *  }
1238      *  // Since the callback function references variables in
its containing object, the containing object
1239      *  // must be passed to hwc.addPushNotificationListener as
well.
1240      *
hwc.addPushNotificationListener( pushListenerManager.listener,
pushListenerManager );
1241      */
1242      hwc.addPushNotificationListener =
function(PushNotificationListener, containingObject)
1243      {
1244      hwc.traceEnteringMethod("hwc.addPushNotificationListener");
1245      try {
1246      hwc._pushnotificationlisteners.push(PushNotificationListener);
1247      hwc._pushnotificationlistenerContainingObjects.push(containingObject);
1248      // The native side will start to notify the
notification when the first
1249      // listener is added
1250      if (hwc._pushnotificationlisteners.length ===
1)
1251      {
1252      hwc.getDataFromContainer("startpushnotificationlistener");
1253      }
```

```
1254         } finally {
1255 hwc.traceLeavingMethod("hwc.addPushNotificationListener");
1256     }
1257 };
1258
1259 /**
1260     * Remove the push notification listener. This function
1261     * should be called with identical parameters that were used
1262     * to add the push notification listener with {@link
1263     * hwc.addPushNotificationListener}.
1264     *
1265     * @public
1266     * @memberOf hwc
1267     * @param {anonymous.PushNotificationListener}
1268     * PushNotificationListener The callback for push notifications.
1269     * @param {Object} [containingObject] The containing object
1270     * of the listener.
1271     * @example
1272     * // pushListener is the callback function that will be
1273     * passed to hwc.addPushNotificationListener.
1274     * var pushListener = function( notifications )
1275     * {
1276     *     alert( "push notification:\n" +
1277     *     JSON.stringify(notifications) );
1278     *     return hwc.NOTIFICATION_CONTINUE;
1279     * }
1280     * hwc.addPushNotificationListener( pushListener );
1281     * // At some other point if we want to remove the push
1282     * listener, we call the following line:
1283     * hwc.removePushNotificationListener( pushListener );
1284     *
1285     * @example
1286     * // pushListenerManager is an object that will contain
1287     * the listener callback as well as a variable
1288     * // referenced from the callback.
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1281     * var pushListenerManager = {};  
1282     * // doSomething is a function that is called from inside  
1283     * // the callback.  
1283     * pushListenerManager.doSomething =  
1283     * function( notifications )  
1284     * {  
1285     *     alert( "push notification:\n" +  
1285     *     JSON.stringify(notifications) );  
1286     *     return hwc.NOTIFICATION_CONTINUE;  
1287     * }  
1288     * // This is the callback function.  
1289     * pushListenerManager.listener =  
1289     * function( notifications )  
1290     * {  
1291     *     return this.doSomething( notifications );  
1292     * }  
1293     * // Since the callback function references variables in  
1293     * // its containing object, the containing object  
1294     * // must be passed to hwc.addPushNotificationListener as  
1294     * // well.  
1295     *  
1295     * hwc.addPushNotificationListener( pushListenerManager.listener,  
1295     * hwc.addPushNotificationListener );  
1296     * // when we want to remove the push listener, we call the  
1296     * // following line:  
1297     * hwc.removePushNotificationListener( pushListener,  
1297     * hwc.removePushNotificationListener );  
1298     */  
1299     hwc.removePushNotificationListener =  
1299     * function(PushNotificationListener, containingObject)  
1300     * {  
1301     *     var i;  
1302     *     hwc.traceEnteringMethod("hwc.removePushNotificationListener");  
1303     *     try {  
1304     *         if (hwc._pushnotificationlisteners.length === 0)  
1304     *         {
```



```

1305         return;
1306     }
1307
1308     for (i = 0; i <
hwc._pushnotificationlisteners.length; i++)
1309     {
1310         if (hwc._pushnotificationlisteners[i] ===
PushNotificationListener &&
1311 hwc._pushnotificationlistenerContainingObjects[i] ===
containingObject)
1312     {
1313         hwc._pushnotificationlisteners.splice(i,
1);
1314 hwc._pushnotificationlistenerContainingObjects.splice(i, 1);
1315         if (hwc._pushnotificationlisteners.length
=== 0)
1316     {
1317 hwc.getDataFromContainer("stoppushnotificationlistener");
1318     }
1319         return;
1320     }
1321     }
1322     } finally {
1323 hwc.traceLeavingMethod("hwc.removePushNotificationListener");
1324     }
1325     };
1326
1327     /**
1328     * A constant indicating that other push notification
listeners should continue to be called.
1329     * Used as a return value for {@link
anonymous.PushNotificationListener} functions.
1330     * @type number

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1331     */
1332     hwc.NOTIFICATION_CONTINUE = 0;
1333     /**
1334     * A constant indicating that no more push notification
1335     * listeners should be called.
1336     * Used as a return value for {@link
1337     * anonymous.PushNotificationListener} functions.
1338     * @type number
1339     */
1340     hwc.NOTIFICATION_CANCEL = 1;
1341     /**
1342     * A sample implementation of a {@link
1343     * anonymous.PushNotificationListener} callback function.
1344     * @param {Array} notifications Array of notifications.
1345     */
1346     hwc.sample_PushNotificationListener =
1347     function(notifications)
1348     {
1349         return hwc.NOTIFICATION_CONTINUE;
1350     };
1351     /**
1352     * This object represents a hybrid app.
1353     * @classdesc
1354     * @public
1355     * @memberOf hwc
1356     * @param {number} moduleId The module id of this hybrid
1357     * app.
1358     * @param {number} version The version of this hybrid
1359     * app.
1360     * @param {string} displayName The display name of this
1361     * hybrid app.
```

```

1359      * @param {number} iconIndex The index specifying the icon
representing this Hybrid App.

1360      * @param {hwc.CustomIcon} defaultCustomIcon The default
custom icon for this hybrid app.

1361      * @param {hwc.CustomIcon[]} customIconList An array of
custom icon objects.

1362      */

1363      hwc.HybridApp = function (moduleId, version, displayName,
iconIndex, defaultCustomIcon, customIconList)

1364      {

1365          this.ModuleID = moduleId;

1366          this.Version = version;

1367          this.DisplayName = displayName;

1368          this.IconIndex = iconIndex;

1369          this.defIcon = defaultCustomIcon;

1370          this.IconList = customIconList;

1371

1372          /**

1373          * Gets the module ID for this hybrid app.

1374          * @public

1375          * @memberOf hwc.HybridApp

1376          * @returns {number} The module ID.

1377          */

1378          this.getModuleID = function ()

1379          {

1380              return this.ModuleID;

1381          };

1382

1383          /**

1384          * Gets the version number for this hybrid app.

1385          * @public

1386          * @memberOf hwc.HybridApp

1387          * @returns {number} The version.

1388          */

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1389         this.getVersion = function ()
1390         {
1391             return this.Version;
1392         };
1393
1394         /**
1395         * Gets the display name for this hybrid app.
1396         * @public
1397         * @memberOf hwc.HybridApp
1398         * @returns {string} The display name.
1399         */
1400         this.getDisplayName = function ()
1401         {
1402             return this.DisplayName;
1403         };
1404
1405         /**
1406         * Gets the icon index used in the list of built-in
1407         * icons.
1408         * @public
1409         * @memberOf hwc.HybridApp
1410         * @returns {number} The icon index
1411         */
1411         this.getIconIndex = function ()
1412         {
1413             return this.IconIndex;
1414         };
1415
1416         /**
1417         * Gets the default custom icon object of this hybrid
1418         * app.
1419         * @public
1420         * @memberOf hwc.HybridApp
```

```

1420      * @returns {hwc.CustomIcon} The default custom icon of
this hybrid app. Null if this hybrid app does not have a custom
icon.
1421      */
1422      this.getDefaultCustomIcon = function ()
1423      {
1424          return this.defIcon;
1425      };
1426
1427      /**
1428      * Gets the list of custom icons associated with this
hybrid app.
1429      * @public
1430      * @memberOf hwc.HybridApp
1431      * @returns {hwc.CustomIcon[]} The array of custom icon
objects. Null if this hybrid app has no custom icons.
1432      */
1433      this.getCustomIconList = function ()
1434      {
1435          return this.IconList;
1436      };
1437
1438      /**
1439      * Return a {@link hwc.ClientVariables} object for the
given module id and version.
1440      * @public
1441      * @memberOf hwc.HybridApp
1442      * @returns {hwc.ClientVariables} The {@link
hwc.ClientVariables} object for this hybrid app.
1443      */
1444      this.getClientVariables = function()
1445      {
1446          return hwc.getClientVariables( this.ModuleID,
this.Version );
1447      };

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1448     };
1449
1450     /**
1451      * An array of {@link anonymous.ApplicationListener}
1452      * callback functions.
1453      * @private
1454      * @type {anonymous.ApplicationListener[]}
1455      */
1456     hwc._applicationListeners = [];
1457
1458     /**
1459      * An array of objects containing {@link
1460      * anonymous.ApplicationListener} callback functions.
1461      * The containing objects need to be kept track of in the
1462      * case that a callback function references
1463      * a variable from its containing object.
1464      * @private
1465      * @type {Array}
1466      */
1467     hwc._applicationListenerContainingObjects = [];
1468
1469     /**
1470      * This is the main entry of application notification. The
1471      * native code should be
1472      * hardcoded to call this function internally.
1473      * @private
1474      */
1475     hwc._applicationListenerNotification = function (event,
1476     moduleId, version)
1477     {
1478         var i, containingObject;
1479         hwc.traceEnteringMethod("hwc._applicationListenerNotification");
1480
1481         try {
```

```
1476         if (hwc._applicationListeners.length === 0){
1477             return;
1478         }
1479
1480         for (i = 0; i < hwc._applicationListeners.length;
1481 i++)
1482         {
1483             containingObject =
1484 hwc._applicationListenerContainingObjects[i];
1485             if (containingObject !== null &&
1486 containingObject !== undefined)
1487             {
1488 hwc._applicationListeners[i].call(containingObject, event, moduleId,
1489 version);
1490             }
1491         }
1492
1493     } finally {
1494 hwc.traceLeavingMethod("hwc._applicationListenerNotification");
1495     }
1496 };
1497
1498 /**
1499  * Register the application listener.
1500  *
1501  * @param {anonymous.ApplicationListener}
1502 ApplicationListener The callback function for application changes.
1503  * @param {Object} [containingObject] The containing object
1504 of the listener method. This parameter is only
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1503      * required if the ApplicationListener references the
1504      * containing object.
1505      * @public
1506      * @memberOf hwc
1507      * @example
1508      * // This is the callback function that will be passed to
1509      * hwc.addAppListener.
1510      * var appListener = function( event, moduleId,
1511      * version )
1512      * {
1513      *     if( event == hwc.APP_ADDED )
1514      *     {
1515      *         alert("A hybrid app has been added.");
1516      *     }
1517      * }
1518      * hwc.addAppListener( appListener );
1519      *
1520      * @example
1521      * // appListenerManager is an object that will contain the
1522      * callback function as well as variables
1523      * // the callback function references.
1524      * var appListenerManager = {};
1525      * // doSomething is a function that is called from inside
1526      * the callback function.
1527      * appListenerManager.doSomething = function( event )
1528      * {
1529      *     if( event == hwc.APP_REMOVED )
1530      *     {
1531      *         alert("A hybrid app has been removed.");
1532      *     }
1533      * }
1534      * // This is the callback function that will be passed to
1535      * hwc.addAppListener. It calls doSomething,
1536      * // the definition of which is in the containing
1537      * function.
```



```

1531     * appListenerManager.listener = function( event,
moduleId, version )
1532     * {
1533     *     this.doSomething( event );
1534     * }
1535     * // Since the listener callback function references a
variable from its containing object,
1536     * // the containing object must be passed to
hwc.addAppListener.
1537     * hwc.addAppListener( appListenerManager.listener,
appListenerManager );
1538     */
1539     hwc.addAppListener = function (ApplicationListener,
containingObject)
1540     {
1541         hwc.traceEnteringMethod("hwc.addAppListener");
1542         try {
1543             hwc._applicationListeners.push(ApplicationListener);
1544             hwc._applicationListenerContainingObjects.push(containingObject);
1545             // The native side will start to notify the
notification when the first
1546             // listener is added
1547             if (hwc._applicationListeners.length === 1)
1548             {
1549                 hwc.getDataFromContainer("startapplistener");
1550             }
1551         } finally {
1552             hwc.traceLeavingMethod("hwc.addAppListener");
1553         }
1554     };
1555
1556     /**
1557     * Remove the application listener. This function should
be called with identical parameters

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1558      * that were used to add the application listener with
{@link hwc.addAppListener}.
1559      *
1560      * @public
1561      * @memberOf hwc
1562      * @param {anonymous.ApplicationListener}
ApplicationListener The callback for application changes.
1563      * @param {Object} [containingObject] The containing object
of the application listener function.
1564      * @example
1565      * // This is the callback function that will be passed to
hwc.addAppListener.
1566      * var appListener = function( event, moduleId,
version )
1567      * {
1568      *     if( event == hwc.APP_ADDED )
1569      *     {
1570      *         alert("A hybrid app has been added.");
1571      *     }
1572      * }
1573      * hwc.addAppListener( appListener );
1574      * // At some other point, if we want to remove the listener
we use the following line of code:
1575      * hwc.removeAppListener( appListener );
1576      *
1577      * @example
1578      * // appListenerManager is an object that will contain the
callback function as well as variables
1579      * // the callback function references.
1580      * var appListenerManager = {};
1581      * // doSomething is a function that is called from inside
the callback function.
1582      * appListenerManager.doSomething = function( event )
1583      * {
1584      *     if( event == hwc.APP_REMOVED )
```

```
1585     *   {
1586     *       alert("A hybrid app has been removed.");
1587     *   }
1588     * }
1589     * // This is the callback function that will be passed to
1590     * // hwc.addAppListener. It calls doSomething,
1591     * // the definition of which is in the containing
1592     * // function.
1593     * // appListenerManager.listener = function( event,
1594     * // moduleId, version )
1595     * {
1596     *     this.doSomething( event );
1597     * }
1598     * // Since the listener callback function references a
1599     * // variable from its containing object,
1600     * // the containing object must be passed to
1601     * // hwc.addAppListener.
1602     * // hwc.addAppListener( appListenerManager.listener,
1603     * // appListenerManager );
1604     * // At some other point, if we want to remove the listener
1605     * // we use the following line of code:
1606     * // hwc.removeAppListener( appListenerManager.listener,
1607     * // appListenerManager );
1608     */
1609     hwc.removeAppListener = function (ApplicationListener,
1610     containingObject)
1611     {
1612         var i;
1613         hwc.traceEnteringMethod("hwc.removeAppListener");
1614         try {
1615             if (hwc._applicationListeners.length === 0) {
1616                 return;
1617             }
1618         }
1619         for (i = 0; i < hwc._applicationListeners.length;
1620 i++)
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1611         {
1612             if (hwc._applicationListeners[i] ===
ApplicationListener &&
1613                 hwc._applicationListenerContainingObjects[i]
=== containingObject)
1614             {
1615                 hwc._applicationListeners.splice(i, 1);
1616                 hwc._applicationListenerContainingObjects.splice(i, 1);
1617                 if (hwc._applicationListeners.length === 0)
1618                 {
1619                     hwc.getDataFromContainer("stopapplistener");
1620                 }
1621                 return;
1622             }
1623         }
1624     } finally {
1625         hwc.traceLeavingMethod("hwc.removeAppListener");
1626     }
1627 };
1628
1629 /**
1630  * A constant indicating that the application list requires
a refresh.
1631  * Used in {@link anonymous.ApplicationListener} callback
functions as a possible value for event.
1632  * @type number
1633  */
1634     hwc.APP_REFRESH = 1;
1635 /**
1636  * A constant indicating that a hybrid app has been
added.
1637  * Used in {@link anonymous.ApplicationListener} callback
functions as a possible value for event.
```

```
1638      * @type number
1639      */
1640      hwc.APP_ADDED = 2;
1641      /**
1642      * A constant indicating that a hybrid app was updated.
1643      * Used in {@link anonymous.ApplicationListener} callback
1644      * functions as a possible value for event.
1645      * @type number
1646      */
1647      hwc.APP_UPDATED = 3;
1648      /**
1649      * A constant indicating that a hybrid app was removed.
1650      * Used in {@link anonymous.ApplicationListener} callback
1651      * functions as a possible value for event.
1652      * @type number
1653      */
1654      hwc.APP_REMOVED = 4;
1655      /**
1656      * A sample {@link anonymous.ApplicationListener} callback
1657      * function.
1658      * @param {number} event A number indicating what event has
1659      * taken place (will be one of {@link hwc.APP_REFRESH},
1660      * {@link hwc.APP_ADDED}, {@link hwc.APP_UPDATED}, {@link
1661      * hwc.APP_REMOVED}).
1662      * @param {number} moduleId The module id of the hybrid app
1663      * the event is about.
1664      * @param {number} version module The version of the hybrid
1665      * app the event is about.
1666      */
1667      hwc.sample_AppListener = function (event, moduleId,
1668      version) {
1669      };
1670
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1665     /**
1666     * Gets the hybrid app that is currently open.
1667     *
1668     * @public
1669     * @memberOf hwc
1670     * @returns {hwc.HybridApp} The hybrid app that is
    currently open.
1671     * @example
1672     * var openHybridApp = hwc.getCurrentApp();
1673     */
1674     hwc.getCurrentApp = function()
1675     {
1676         var response, currentApp, app ;
1677
1678         hwc.traceEnteringMethod("hwc.getCurrentApp");
1679         response = "";
1680
1681         try {
1682             response =
    hwc.getDataFromContainer("getcurrentapp");
1683
1684             if (response !== "")
1685             {
1686                 app = JSON.parse(response);
1687                 currentApp = new hwc.HybridApp(app.moduleId,
    app.version, app.displayName, app.iconIndex,
1688                 hwc.createCustomIconObject(app.defaultCustomIcon, app.moduleId,
    app.version, hwc.DEFAULT_CUSTOM_ICON_INDEX),
1689                 hwc.createCustomIconList(app.customIconList, app.moduleId,
    app.version));
1690             }
1691         } catch (ex){
```

```

1692         hwc.log("getCurrentApp error:" + ex.message, "ERROR",
false);
1693     } finally {
1694         hwc.traceLeavingMethod("hwc.getCurrentApp");
1695     }
1696
1697     return currentApp;
1698 };
1699
1700 /**
1701  * Returns an array of {@link hwc.HybridApp} objects.
1702  *
1703  * @public
1704  * @memberOf hwc
1705  * @param {boolean} [completeList] If this parameter is set
to true, then all apps that are user invocable or require
1706  *     activation will be returned. If set to false or if
it is not set, then if there is a default hybrid app
1707  *     only the default hybrid app will be returned (and
if there is no default hybrid app it will return all hybrid apps
1708  *     that are user invocable or require
activation).
1709  *
1710  * @returns {hwc.HybridApp[]} An array of hybrid app
objects.
1711  * @example
1712  * var apps = hwc.getInstalledApps();
1713  *
1714  * @example
1715  * var apps = hwc.getInstalledApps( true );
1716  */
1717     hwc.getInstalledApps = function( completeList )
1718     {
1719         var formattedCompleteList, response, installedApps, app,
apps, i;

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1720
1721     hwc.traceEnteringMethod("hwc.getInstalledApps");
1722     formattedCompleteList = false;
1723     response = "";
1724     installedApps = [];
1725
1726     if( completeList )
1727     {
1728         formattedCompleteList = true;
1729     }
1730
1731
1732     try {
1733         response =
1734         hwc.getDataFromContainer("getinstalledapps", "&getcompletelist=" +
1735         formattedCompleteList);
1736
1737         if (response !== null && response !== undefined &&
1738         response !== "")
1739         {
1740             apps = JSON.parse(response);
1741             for (i=0; i<apps.length; i++) {
1742                 app = apps[i];
1743                 installedApps[i] = new
1744                 hwc.HybridApp(app.moduleId, app.version, app.displayName,
1745                 app.iconIndex,
1746                 hwc.createCustomIconObject(app.defaultCustomIcon, app.moduleId,
1747                 app.version, hwc.DEFAULT_CUSTOM_ICON_INDEX),
1748                 hwc.createCustomIconList(app.customIconList, app.moduleId,
1749                 app.version));
1750             }
1751         }
1752     } catch (ex){
```



```
1746         hwc.log("getInstalledApps error:" + ex.message,
"ERROR", false);
1747     } finally {
1748         hwc.traceLeavingMethod("hwc.getInstalledApps");
1749     }
1750
1751     return installedApps;
1752 };
1753
1754 /**
1755  * Returns an array of {@link hwc.HybridApp} objects that
are server initiated.
1756  *
1757  * @public
1758  * @memberOf hwc
1759  * @returns {hwc.HybridApp[]} An array of server initiated
hybrid apps.
1760  * @example
1761  * var serverInitiatedApps =
hwc.getServerInitiatedApps();
1762  */
1763     hwc.getServerInitiatedApps = function()
1764     {
1765         var response = "", serverInitiatedApps = [], app, apps,
i;
1766
1767         hwc.traceEnteringMethod("hwc.getServerInitiatedApps");
1768         try {
1769             response =
hwc.getDataFromContainer("getserverinitiatedapps");
1770
1771             if (response !== null && response !== undefined &&
response !== "")
1772                 {
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1773         apps = JSON.parse(response);
1774         for (i=0; i<apps.length; i++) {
1775             app = apps[i];
1776             serverInitiatedApps[i] = new
hwc.HybridApp(app.moduleId, app.version, app.displayName,
app.iconIndex,
1777             hwc.createCustomIconObject(app.defaultCustomIcon, app.moduleId,
app.version, hwc.DEFAULT_CUSTOM_ICON_INDEX),
1778             hwc.createCustomIconList(app.customIconList, app.moduleId,
app.version));
1779         }
1780     }
1781     } catch (ex){
1782         hwc.log("getServerInitiatedApps error:" + ex.message,
"ERROR", false);
1783     } finally {
1784         hwc.traceLeavingMethod("hwc.getServerInitiatedApps");
1785     }
1786
1787     return serverInitiatedApps;
1788 };
1789
1790 /**
1791  * Gets a {@link hwc.HybridApp} object with the given
module id and version.
1792  *
1793  * @public
1794  * @memberOf hwc
1795  * @param {number} moduleId The module ID of the hybrid
app.
1796  * @param {number} version The version of the hybrid
app.
1797  *
```

```

1798      * @returns {hwc.HybridApp} The hybrid app object, or null
if there is no hybrid app with the given ID and version.
1799      *
1800      * @example
1801      * // Messages do not have a direct link to the hybrid app
they belong to. Instead they have
1802      * // the module ID and version of the hybrid app they
belong to. If you have a message and
1803      * // need to access its hybrid app, first you must call
hwc.getAppByID.
1804      * var messages = hwc.getAllMessages();
1805      * if( messages.length > 0 )
1806      * {
1807      *   var app = hwc.getAppByID( messages[0].getModuleId(),
messages[0].getModuleVersion() );
1808      * }
1809      */
1810      hwc.getAppByID = function (moduleID, version)
1811      {
1812          var response, appInstance, app, params;
1813
1814          hwc.traceEnteringMethod("hwc.getAppByID");
1815          response = "";
1816          params = "&moduleid=" + moduleID + "&moduleversion=" +
version;
1817
1818          try {
1819              response = hwc.getDataFromContainer("getappbyid",
params);
1820
1821              if (response !== "")
1822              {
1823                  app = JSON.parse(response);
1824                  appInstance = new hwc.HybridApp(app.moduleId,
app.version, app.displayName, app.iconIndex,

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1825 hwc.createCustomIconObject(app.defaultCustomIcon, app.moduleId,
hwc.version, hwc.DEFAULT_CUSTOM_ICON_INDEX),
1826 hwc.createCustomIconList(app.customIconList, app.moduleId,
app.version));
1827     }
1828     } catch (ex){
1829         hwc.log("getAppByID error:" + ex.message, "ERROR",
false);
1830     } finally {
1831         hwc.traceLeavingMethod("hwc.getAppByID");
1832     }
1833
1834     return appInstance;
1835 };
1836
1837 /**
1838  * A constant indicating that {@link hwc.openApp} completed
successfully.
1839  * This is a possible return value for {@link
hwc.openApp}.
1840  * @type number
1841  */
1842     hwc.OPEN_APP_SUCCESS = 0;
1843 /**
1844  * A constant indicating that {@link hwc.openApp} failed
because the specified app does not exist.
1845  * This is a possible return value for {@link
hwc.openApp}.
1846  * @type number
1847  */
1848     hwc.OPEN_APP_NOT_EXIST = 1;
1849 /**
1850  * A constant indicating that {@link hwc.openApp} failed
for an unspecified reason.
```

```

1851      * This is a possible return value for {@link
hwc.openApp}.
1852      * @type number
1853      */
1854      hwc.OPEN_APP_OTHER = 2;
1855
1856      /**
1857      * Launch the hybrid app with the given module ID and
version. The hybrid app will be opened on top of the hybrid app
1858      * that is open when hwc.openApp is called. When the
hybrid app that was opened with hwc.openApp exits, it will exit
1859      * to the hybrid app that was open when hwc.openApp was
called. It is possible to nest open hybrid apps, but it is
1860      * best not to have too many nested hybrid apps (eg:
recursively opening hybrid apps) because each open hybrid app
1861      * takes up device memory.
1862      *
1863      * @param {number} moduleId Module id of the hybrid app.
1864      * @param {number} version Version of the hybrid app.
1865      *
1866      * @returns {number} A constant indicating the result of
opening the hybrid app (will be one of {@link
hwc.OPEN_APP_SUCCESS},
1867      * {@link hwc.OPEN_APP_NOT_EXIST}, {@link
hwc.OPEN_APP_OTHER}).
1868      * @public
1869      * @memberOf hwc
1870      *
1871      * @example
1872      * var apps = hwc.getInstalledApps();
1873      * if( apps.length > 0 )
1874      * {
1875      *     // Check to make sure the first app is not this app
(the app that is currently running),
1876      *     // since we don't want to recursively open this app
until memory runs out.

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1877      *   if( hwc.getCurrentHybridApp.getDisplayName() !=
apps[0].getDisplayName() )
1878      *   {
1879      *       hwc.openApp( apps[0].getModuleID(),
apps[0].getVersion() );
1880      *   }
1881      * }
1882      */
1883      hwc.openApp = function (moduleId, version)
1884      {
1885          var response;
1886          hwc.traceEnteringMethod("hwc.openApp");
1887          try {
1888              response = hwc.getDataFromContainer("openhybridapp",
"&moduleid=" + moduleId + "&moduleversion=" + version);
1889              return parseInt(response, 10);
1890          } catch (ex){
1891              hwc.log("app.open error:" + ex.message, "ERROR",
false);
1892          } finally {
1893              hwc.traceLeavingMethod("hwc.openApp");
1894          }
1895      };
1896
1897      /**
1898      * A constant indicating the custom icon index.
1899      * @type number
1900      */
1901      hwc.DEFAULT_CUSTOM_ICON_INDEX = -1;
1902
1903      /**
1904      * Gets the Hybrid Web Container application connection
ID.
1905      *
```

```
1906     * @public
1907     * @memberOf hwc
1908     * @returns {string} Application connection ID
1909     * @example
1910     * var appConnectionID =
hwc.getApplicationConnectionID();
1911     */
1912     hwc.getApplicationConnectionID = function() {
1913         var response = "";
1914
1915         hwc.traceEnteringMethod("hwc.getApplicationConnectionID");
1916         try
1917         {
1918             response =
hwc.getDataFromContainer("getconnectionid");
1919         }
1920         catch (ex) {
1921             hwc.log("get connection id error:" + ex.message,
"ERROR", false);
1922         } finally {
1923             hwc.traceLeavingMethod("hwc.getApplicationConnectionID");
1924         }
1925
1926         return String(response);
1927     };
1928
1929     /**
1930     * Gets the client variables of the hybrid app with given
module id and version.
1931     *
1932     * @public
1933     * @memberOf hwc
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1934      * @param {number} moduleID The module ID of the hybrid
app.
1935      * @param {number} version The version of the hybrid
app.
1936      *
1937      * @returns {hwc.ClientVariables} A {@link
hwc.ClientVariables} object, or null if there are no
1938      * ClientVariables for the hybrid app with the given module
id and version.
1939      * @example
1940      * var apps = hwc.getInstalledApps();
1941      * // Loop through the apps, showing the client variables
for each one.
1942      * for( var i = 0; i < apps.length; i++ )
1943      * {
1944      *     var app = apps[i];
1945      *     // Get the client variables.
1946      *     var clientVariables =
hwc.getClientVariables( app.getModuleID(), app.getVersion() );
1947      *     if( clientVariables.getCount() > 0 )
1948      *     {
1949      *         // Get all the names of the variables for this
app.
1950      *         var keys =
clientVariables.getAllVariableNames();
1951      *         // Loop through all the variable for this app.
1952      *         for( var index = 0; index < keys.length; index+
+ )
1953      *         {
1954      *             // Get a specific variable by name.
1955      *             var variable =
clientVariables.getVariableValueByName( keys[index] );
1956      *             alert( "variable name: " + keys[index] +
"\nvariable value: " + variable );
1957      *         }
1958      *     }
1959      * }
```



```
1960     */
1961     hwc.getClientVariables = function (moduleID, version)
1962     {
1963         var response, clientVariables, parsedResponse,
1964         params;
1965         hwc.traceEnteringMethod("hwc.getClientVariables");
1966         response = "";
1967         clientVariables = null;
1968
1969         params = "&moduleid=" + moduleID + "&moduleversion=" +
1970         version;
1971         try
1972         {
1973             response =
1974             hwc.getDataFromContainer("getclientvariables", params);
1975
1976             if (response !== "")
1977             {
1978                 parsedResponse = JSON.parse( response );
1979                 clientVariables = new
1980                 hwc.ClientVariables( parsedResponse.version,
1981                 parsedResponse.items );
1982             }
1983         }
1984         catch (ex)
1985         {
1986             hwc.log("getClientVariables error:" + ex.message,
1987             "ERROR", false);
1988         } finally {
1989             hwc.traceLeavingMethod("hwc.getClientVariables");
1990         }
1991
1992         return clientVariables;
1993     }
1994 }
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1988     };
1989
1990     /**
1991     * Represents a ClientVariables object.
1992     *
1993     * @classdesc
1994     *
1995     * @public
1996     * @param {number} clientVariablesVersion The version of
client variables.
1997     * @param {Object} clientVariableItems The json object
that contains key/value pairs of client variable items.
1998     * @memberOf hwc
1999     */
2000     hwc.ClientVariables = function ( clientVariablesVersion,
clientVariablesItems )
2001     {
2002         this.version = clientVariablesVersion;
2003         this.items = clientVariablesItems;
2004
2005         /**
2006         * Gets the version of the client variables.
2007         * @returns {number} The version of the client
variables.
2008         * @public
2009         * @memberOf hwc.ClientVariables
2010         */
2011         this.getVersion = function ()
2012         {
2013             return this.version;
2014         };
2015
2016         /**
```

```

2017      * Gets the number of variables this {@link
hwc.ClientVariables} contains.
2018      * @public
2019      * @memberOf hwc.ClientVariables
2020      * @returns {number} The number of variables.
2021      */
2022      this.getCount = function ()
2023      {
2024          var keys = this.getAllVariableNames();
2025
2026          return keys.length;
2027      };
2028
2029      /**
2030      * Gets an array containing the names of all variables in
this {@link hwc.ClientVariables}.
2031      *
2032      * @public
2033      * @memberOf hwc.ClientVariables
2034      * @returns {string[]} The array holding the names of all
variables contained in this {@link hwc.ClientVariables}.
2035      */
2036      this.getAllVariableNames = function ()
2037      {
2038          var result, prop;
2039          hwc.traceEnteringMethod("hwc.ClientVariables.getAllVariableNames");
2040          try {
2041              result = [];
2042
2043              if ( this.items !== undefined && this.items !==
null )
2044              {
2045                  for ( prop in this.items )

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2046         {
2047             if ( this.items.hasOwnProperty( prop ) &&
typeof this.items[ prop ] === 'string' )
2048                 {
2049                     result.push( prop );
2050                 }
2051             }
2052         }
2053         result.sort();
2054         return result;
2055     } finally {
2056         hwc.traceLeavingMethod("hwc.ClientVariables.getAllVariableNames");
2057     }
2058 };
2059
2060
2061     /**
2062     * Check if this {@link hwc.ClientVariables} has a
variable by the given name.
2063     *
2064     * @public
2065     * @memberOf hwc.ClientVariables
2066     * @param {string} variableName The name of variable to
check for.
2067     *
2068     * @returns {boolean} True if this {@link
hwc.ClientVariables} has a variable by the given name, false
otherwise.
2069     */
2070     this.containsName = function ( variableName )
2071     {
2072         if ( this.items === undefined || this.items === null
|| ( typeof this.items[ variableName ] !== 'string' ) )
2073             {
```

```
2074         return false;
2075     }
2076
2077         return true;
2078     };
2079
2080     /**
2081     * Gets the value of the variable with the given name. If
2082     * this {@link hwc.ClientVariables} does not have a variable
2083     * by the given name, a {@link
2084     * hwc.ClientVariablesException} will be thrown.
2085     *
2086     * @public
2087     * @memberOf hwc.ClientVariables
2088     * @param {string} variableName The name of the variable
2089     * to get the value of.
2090     *
2091     * @returns {string} The value of the variable.
2092     *
2093     * @throws {hwc.ClientVariableException} This exception
2094     * is thrown when there is no variable by the given name in this
2095     * {@link hwc.ClientVariables}.
2096     */
2097     this.getVariableValueByName = function
2098     ( variableName )
2099     {
2100         if ( !this.containsName( variableName ) )
2101         {
2102             throw new
2103             hwc.ClientVariablesException( hwc.ClientVariables.ITEM_NOT_FOUND,
2104             "Unable to find variable name: " + variableName );
2105         }
2106     }
2107
2108     return this.items[ variableName ];
2109 };
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2101     };
2102
2103     /**
2104      * This exception is thrown when {@link
2105      hwc.ClientVariables#getVariableValueByName} is called with a
2106      variable name that does not exist.
2107      * @public
2108      * @memberOf hwc
2109      * @param {number} errCode The error code (will be {@link
2110      hwc.ClientVariables.ITEM_NOT_FOUND}).
2111      * @param {string} errMsg A message describing the
2112      error.
2113      * @classdesc
2114      */
2115     hwc.ClientVariablesException = function(errCode, errMsg)
2116     {
2117         this.errCode = errCode;
2118         this.errMsg = errMsg;
2119     };
2120
2121     /**
2122      * A constant indicating that a variable does not exist in
2123      a {@link hwc.ClientVariables} object.
2124      * @type number
2125      */
2126     hwc.ClientVariables.ITEM_NOT_FOUND = 1;
2127
2128     /**
2129      * Represents a CustomIcon. Used with the {@link
2130      hwc.HybridApp} object.
2131      * @classdesc
2132      * @public
2133      * @memberOf hwc
2134      * @param {number} width The width of this custom icon.
```

```

2128      * @param {number} height The height of this custom
icon.
2129      * @param {string} type The image type of this custom
icon.
2130      * @param {string} name The name of this custom icon.
2131      * @param {string} path The file path of the unprocessed
icon.
2132      * @param {string} processedPath The file path of the
processed icon.
2133      * @param {number} moduleId The module ID of the hybrid app
this icon is for.
2134      * @param {number} moduleVersion The module version of the
hybrid app this icon is for.
2135      * @param {number} index The index of this custom icon.
2136      */
2137      hwc.CustomIcon = function (width, height, type, name, path,
processedPath, moduleId, moduleVersion, index)
2138      {
2139          this.w = width;
2140          this.h = height;
2141          this.t = type;
2142          this.n = name;
2143          this.p = path;
2144          this.pp = processedPath;
2145          this.mi = moduleId;
2146          this.mv = moduleVersion;
2147          this.index = index;
2148
2149          /**
2150           * Gets the width of this custom icon.
2151           * @public
2152           * @memberOf hwc.CustomIcon
2153           * @returns {number} The width of this custom icon.
2154           */
2155          this.getWidth = function ()

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2156     {
2157         return this.w;
2158     };
2159
2160     /**
2161     * Gets the height of this custom icon.
2162     * @public
2163     * @memberOf hwc.CustomIcon
2164     * @returns {number} The height of this custom icon.
2165     */
2166     this.getHeight = function ()
2167     {
2168         return this.h;
2169     };
2170
2171     /**
2172     * Gets the image type of this custom icon.
2173     * @public
2174     * @memberOf hwc.CustomIcon
2175     * @returns {string} The file type of the image.
2176     */
2177     this.getType = function ()
2178     {
2179         return this.t;
2180     };
2181
2182     /**
2183     * Gets the name of this custom icon.
2184     * @public
2185     * @memberOf hwc.CustomIcon
2186     * @returns {string} The name of this custom icon.
2187     */
```



```
2188         this.getName = function ()
2189         {
2190             return this.n;
2191         };
2192
2193         /**
2194          * Gets the file path of the unprocessed icon.
2195          * @public
2196          * @memberOf hwc.CustomIcon
2197          * @returns {string} The file path of the unprocessed
2198          icon.
2199          */
2200         this.getImagePath = function ()
2201         {
2202             return this.p;
2203         };
2204
2205         /**
2206          * Gets the file path of the processed icon.
2207          * @public
2208          * @memberOf hwc.CustomIcon
2209          * @returns {string} The file path of the processed
2210          icon.
2211          */
2212         this.getProcessedImagePath = function ()
2213         {
2214             return this.pp;
2215         };
2216
2217         /**
2218          * Gets the URL of this custom icon. It is possible to
2219          call this function directly, but generally
2220          * it is easier simply to call {@link hwc.getAppIconUrl}
2221          or {@link hwc.getMsgIconUrl}. Those
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2218      * functions handle both cases where there is and isn't a
2219      * custom icon for the hybrid app or message.
2220      * @public
2221      * @memberOf hwc
2222      * @param {boolean} processed When set to true, the URL
2223      * of the processed icon will be returned.
2224      *
2225      * When set to false, the URL of the unprocessed icon
2226      * will be returned.
2227      *
2228      * @returns {string} The URL to the target icon.
2229      * @example
2230      * var apps = hwc.getInstalledApps();
2231      * var app = apps[0];
2232      * // If app doesn't have a custom icon, then customIcon
2233      * will be null.
2234      * var customIcon = app.getDefaultCustomIcon();
2235      * if( customIcon != null )
2236      * {
2237      *     // Create the image element.
2238      *     var image = document.createElement( "img" );
2239      *     // Set the source of the image to the icon URL.
2240      *     image.setAttribute( 'src',
2241      * customIcon.getIconUrl() );
2242      *     // Add the image element to the page.
2243      *     document.body.appendChild( image );
2244      * }
2245      */
2246      this.getIconUrl = function (processed)
2247      {
2248          return hwc.getCustomIconUrl(this.mi, this.mv,
2249          this.index, processed);
2250      };
2251      };
2252      };
```

```
2246
2247     /**
2248     * This method is called internally.
2249     * @private
2250     * @param {Object} jsonObj The JSON object containing
    information about the custom icon.
2251     * @param {number} moduleId The module ID of the hybrid app
    this custom icon belongs to.
2252     * @param {number} moduleVersion The module version of the
    hybrid app this custom icon belongs to.
2253     * @param {number} index The index of this custom icon.
2254     * @returns {hwc.CustomIcon} The new CustomIcon object.
2255     */
2256     hwc.createCustomIconObject = function(jsonObj, moduleId,
    moduleVersion, index)
2257     {
2258         if (jsonObj === null) {
2259             return null;
2260         }
2261
2262         if (jsonObj === undefined) {
2263             return undefined;
2264         }
2265
2266         return new hwc.CustomIcon(jsonObj.width,
    jsonObj.height, jsonObj.type, jsonObj.name, jsonObj.path,
    jsonObj.processedPath,
2267             moduleId, moduleVersion, index);
2268     };
2269
2270     /**
2271     * This method is called internally
2272     * @private
2273     * @param {Array} jsonArr An array of JSON objects that
    contain information about custom icons
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2274      * @param {number} moduleId The module ID that will be
associated with the custom icons
2275      * @param {number} moduleVersion The module version that
will be associated with the custom icons
2276      * @returns {hwc.CustomIcon[]} An array of CustomIcon
objects
2277      */
2278      hwc.createCustomIconList = function (jsonArr, moduleId,
moduleVersion)
2279      {
2280          var iconArray, i, icon;
2281          iconArray = [];
2282
2283          if (jsonArr === null) {
2284              return null;
2285          }
2286
2287          if (jsonArr === undefined) {
2288              return undefined;
2289          }
2290
2291          if (jsonArr.length > 0)
2292          {
2293              for (i=0; i<jsonArr.length; i++)
2294              {
2295                  icon = hwc.createCustomIconObject (jsonArr[i],
moduleId, moduleVersion, i);
2296                  if (icon !== null && icon !== undefined){
2297                      iconArray.push(icon);
2298                  }
2299              }
2300          }
2301
2302          return iconArray;
```

```

2303     };
2304
2305
2306     /**
2307     * Gets the URL to the custom icon. This function is used
2308     * by {@link hwc.CustomIcon#getIconUrl}.
2309     *
2310     * @public
2311     * @memberOf hwc
2312     * @param {number} moduleId The module Id of the hybrid app
2313     * the custom icon belongs to.
2314     * @param {number} moduleVersion The version of the hybrid
2315     * app the custom icon belongs to.
2316     * @param {number} iconIndex The index of the custom
2317     * icon.
2318     * @param {boolean} processed Whether to get the processed
2319     * icon (true), or the unprocessed icon (false).
2320     *
2321     * @returns {string} The URL to the target icon.
2322     */
2323     hwc.getCustomIconUrl = function (moduleId, moduleVersion,
2324     iconIndex, processed)
2325     {
2326         return getRequestUrl("customicon", "moduleid=" +
2327         moduleId+ "&moduleversion=" + moduleVersion + "&iconindex=" +
2328         iconIndex + "&processed=" + processed);
2329     };
2330
2331
2332     /**
2333     * Gets the icon URL for the built-in icon. This function
2334     * is used by {@link hwc.getMsgIconUrl} and {@link hwc.getAppIconUrl}.
2335     *
2336     * It is possible to call this function directly, but
2337     * generally it is easier simply to call {@link hwc.getAppIconUrl} or
2338     *
2339     * {@link hwc.getMsgIconUrl} instead. Those functions
2340     * handle both cases where there is and isn't a custom icon for the
2341     * hybrid app or message.
2342     *
2343     */

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2328      * @public
2329      * @memberOf hwc
2330      * @param {number} iconIndex The index of the built-in
icon.
2331      * @param {boolean} processed Whether or not to get the URL
of the processed icon (true) or the unprocessed icon (false).
2332      *
2333      * @returns {string} The URL to the icon.
2334      * @example
2335      * // Create the image element.
2336      * var builtInIcon = document.createElement( "img" );
2337      * // Set the source of the image to the icon URL.
2338      * builtInIcon.setAttribute( 'src',
hwc.getBuiltInIconUrl(56, false) );
2339      * // Add the image element to the page.
2340      * document.body.appendChild( builtInIcon );
2341      */
2342      hwc.getBuiltInIconUrl = function (iconIndex, processed)
2343      {
2344          return getRequestUrl("clienticon", "iconindex=" +
iconIndex + "&processed=" + processed);
2345      };
2346
2347      /**
2348      * This function gets the URL of the icon for a message
object depending on its
2349      * processed status and whether there are custom icons
defined.
2350      *
2351      * @public
2352      * @memberOf hwc
2353      * @param {hwc.Message} msg The message object
2354      *
2355      * @returns {string} The url to access the icon.
```

```
2356      * @example
2357      * var messages = hwc.getAllMessages();
2358      * if( messages.length > 0 )
2359      * {
2360      *     // Create the image element.
2361      *     var messageIcon = document.createElement("img");
2362      *     // Set the source of the image to the icon URL.
2363      *     messageIcon.setAttribute( 'src',
2364      hwc.getMsgIconUrl( messages[0] ) );
2365      *     // Add the image element to the page.
2366      *     document.body.appendChild( messageIcon );
2367      * }
2368      */
2369      hwc.getMsgIconUrl = function (msg)
2370      {
2371          hwc.traceEnteringMethod("hwc.getMsgIconUrl");
2372          try {
2373              var app = hwc.getAppById(msg.getModuleId(),
2374              msg.getModuleVersion());
2375              if (app === null || app === undefined) {
2376                  return hwc.getBuiltInIconUrl(msg.getIconIndex(),
2377                  msg.isProcessed());
2378              } else {
2379                  return hwc.getAppIconUrl(app,
2380                  msg.isProcessed());
2381              }
2382              } finally {
2383                  hwc.traceLeavingMethod("hwc.getMsgIconUrl");
2384              }
2385          };
2386      /**
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2385      * This function gets the URL of the icon for a hybrid app
2386      * depending on whether custom icons are defined.
2387      *
2388      * @public
2389      * @memberOf hwc
2390      * @param {hwc.HybridApp} app The hybrid app for which the
2391      * icon URL is desired.
2392      * @param {boolean} processed Whether to get the URL of the
2393      * processed icon (true) or the URL of the unprocessed icon (false).
2394      *
2395      * @returns {string} The URL of the icon.
2396      * @example
2397      * var apps = hwc.getInstalledApps();
2398      * if( apps.length > 0 )
2399      * {
2400      *     var hybridApp = apps[0];
2401      *     // Create the image element.
2402      *     var hybridAppIcon = document.createElement("img");
2403      *     // Set the source of the image to the icon URL.
2404      *     hybridAppIcon.setAttribute( 'src',
2405      * hwc.getAppIconUrl( hybridApp, false ) );
2406      *     // Add the image element to the page.
2407      *     document.body.appendChild( hybridAppIcon );
2408      * }
2409      */
2410      hwc.getAppIconUrl = function(app, processed)
2411      {
2412          hwc.traceEnteringMethod("hwc.getAppIconUrl");
2413          try {
2414              var ci = app.getDefaultCustomIcon();
2415              if (ci !== null && ci !== undefined)
2416              {
2417                  return ci.getIconUrl(processed);
2418              }
2419          }
2420      }
```



```

2415         else
2416         {
2417             return hwc.getBuiltInIconUrl (app.getIconIndex (),
processed);
2418         }
2419     } finally {
2420         hwc.traceLeavingMethod("hwc.getAppIconUrl");
2421     }
2422 };
2423
2424 /**
2425  * Represents a message received by the HWC.
2426  *
2427  * @classdesc
2428  * @public
2429  * @memberOf hwc
2430  * @param {number} msgId The message ID of this message.
2431  * @param {Date} date The date this message was
received.
2432  * @param {number} icon The icon index for this message.
2433  * @param {string} sender The sender of this message.
2434  * @param {boolean} isRead Whether this message has been
read or not.
2435  * @param {boolean} processed Whether this message has been
processed or not.
2436  * @param {number} priority The priority of this message
(must be either {@link hwc.MSG_PRIORITY_HIGH} or {@link
hwc.MSG_PRIORITY_NORMAL}).
2437  * @param {string} subject The subject of this message.
2438  * @param {number} module The module ID of the hybrid app
associated with this message.
2439  * @param {number} version The version of the hybrid app
associated with this message.
2440  */
2441     hwc.Message = function (msgId, date, icon, sender, isRead,
processed, priority, subject, module, version)

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2442     {
2443         this.msgId = msgId;
2444         this.recvDate = date;
2445         this.iconIndex = icon;
2446         this.subject = subject;
2447         this.moduleId = module;
2448         this.version = version;
2449         this.processed = processed;
2450         this.sender = sender;
2451         this.isread = isRead;
2452         this.priority = priority;
2453
2454         /**
2455          * Gets the message ID of this message.
2456          * @public
2457          * @memberOf hwc.Message
2458          * @returns {number} The message ID of this message.\
2459          */
2460         this.getMessageId = function ()
2461         {
2462             return this.msgId;
2463         };
2464
2465         /**
2466          * Gets the date this message was received.
2467          * @public
2468          * @memberOf hwc.Message
2469          * @returns {Date} The date this message was
2470          received.
2471          */
2472         this.getReceivedDate = function ()
```

```
2473         return this.recvDate;
2474     };
2475
2476     /**
2477      * Gets the icon index of this message.
2478      * @public
2479      * @memberOf hwc.Message
2480      * @returns {number} The icon index of this message.
2481      */
2482     this.getIconIndex = function ()
2483     {
2484         return this.iconIndex;
2485     };
2486
2487     /**
2488      * Gets the sender of this message.
2489      * @public
2490      * @memberOf hwc.Message
2491      * @returns {string} The sender of this message.
2492      */
2493     this.getSender = function ()
2494     {
2495         return this.sender;
2496     };
2497
2498     /**
2499      * Gets whether this message has been read or not.
2500      * @public
2501      * @memberOf hwc.Message
2502      * @returns {boolean} Whether this message has been read
2503      (true) or not (false).
2504     */
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2504         this.isRead = function ()
2505         {
2506             return this.isread;
2507         };
2508
2509         /**
2510          * Gets the subject of this message.
2511          * @public
2512          * @memberOf hwc.Message
2513          * @returns {string} The subject of this message.
2514          */
2515         this.getSubject = function ()
2516         {
2517             return this.subject;
2518         };
2519
2520         /**
2521          * Gets the module ID of the hybrid app this message
2522          * belongs to.
2523          * @public
2524          * @memberOf hwc.Message
2525          * @returns {number} The module ID of the hybrid app this
2526          * message belongs to.
2527          */
2528         this.getModuleId = function ()
2529         {
2530             return this.moduleId;
2531         };
2532
2533         /**
2534          * Gets the version of the hybrid app this message
2535          * belongs to.
2536          * @public
```

```

2534         * @memberOf hwc.Message
2535         * @returns {number} The version of the hybrid app this
message belongs to.
2536         */
2537         this.getModuleVersion = function ()
2538         {
2539             return this.version;
2540         };
2541
2542         /**
2543         * Gets whether this message has been processed or not. A
message is generally marked as processed once
2544         * the user submits changes from the hybrid app that was
launched from the message.
2545         *
2546         * @public
2547         * @memberOf hwc.Message
2548         * @returns {boolean} True if this message has been
processed, false otherwise.
2549         */
2550         this.isProcessed = function ()
2551         {
2552             return this.processed;
2553         };
2554
2555         /**
2556         * Gets the priority of the message.
2557         *
2558         * @public
2559         * @memberOf hwc.Message
2560         * @returns {number} A constant indicating the priority
of the message.
2561         * Will be either {@link hwc.MSG_PRIORITY_NORMAL} or
{@link hwc.MSG_PRIORITY_HIGH}.

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2562     */
2563     this.getPriority = function ()
2564     {
2565         if (this.priority === hwc.MSG_PRIORITY_HIGH) {
2566             return hwc.MSG_PRIORITY_HIGH;
2567         } else {
2568             return hwc.MSG_PRIORITY_NORMAL;
2569         }
2570     };
2571
2572
2573     /**
2574     * Updates the read status of the message.
2575     *
2576     * @public
2577     * @memberOf hwc.Message
2578     * @param {boolean} status The new read status.
2579     */
2580     this.updateRead = function(status)
2581     {
2582         hwc.updateMessageRead(this.msgId, status);
2583         this.isread = status;
2584     };
2585
2586     /**
2587     * Updates the processed status of the message.
2588     *
2589     * @public
2590     * @memberOf hwc.Message
2591     * @param {boolean} status The new processed status.
2592     */
2593     this.updateProcessed = function(status)
```

```
2594         hwc.updateMessageProcessed(this.msgId, status);
2595         this.processed = status;
2596     };
2597 };
2598
2599 /**
2600  * Represents a filter used to filter messages.
2601  * Pass in null for any parameter you do not wish to filter
2602  * (or do not pass in such parameters at all).
2603  *
2604  * @classdesc
2605  * @public
2606  * @memberOf hwc
2607  * @param {string} [sender] The sender of the message.
2608  * @param {string} [subject] The subject of the message.
2609  * @param {number} [moduleId] The associated application
2610  * module ID.
2611  * @param {number} [version] The associated application
2612  * module versions.
2613  * @param {boolean} [isread] The read status.
2614  * @param {boolean} [processed] The processed status.
2615  *
2616  */
2617 hwc.MessageFilter = function (sender, subject, moduleId,
2618 version, isread, processed)
2619 {
2620     this.sender = sender;
2621     this.subject = subject;
2622     this.moduleId = moduleId;
2623     this.version = version;
2624     this.isRead = isread;
2625     this.processed = processed;
2626 };
2627
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2624     /**
2625     * An array of message listener callback functions.
2626     * @private
2627     * @type {anonymous.MessageListener[]}
2628     */
2629     hwc._messageListeners = [];
2630     /**
2631     * An array of objects containing message listener callback
2632     * functions.
2633     * The containing objects need to be kept track of since
2634     * the callback functions may reference
2635     * variables in the containing object.
2636     * @private
2637     * @type Array
2638     */
2639     hwc._messageListenerContainingObjects = [];
2640     /**
2641     * An array of {@link hwc.MessageFilter} objects.
2642     * @private
2643     * @type {hwc.MessageFilter[]}
2644     */
2645     hwc._messageListenerFilters = []; // Array of
2646     MessageFilter
2647     /**
2648     * This is the main entry of message notification. The
2649     * native code should be
2650     * hardcoded to call this function
2651     * @private
2652     * @param {number} flag Will be one of: {@link
2653     * hwc.MSG_ADDED}, {@link hwc.MSG_REMOVED}, {@link hwc.MSG_UPDATED},
2654     * {@link hwc.MSG_REFRESH}
2655     * @param {number} msgId The message id that this
2656     * notification is about.
2657     */
```



```
2652     hwc._messageListenerNotification = function (flag,
msgId)
2653     {
2654         var i, filter, msg, containingObject;
2655
hwc.traceEnteringMethod("hwc._messageListenerNotification");
2656         try {
2657             if (hwc._messageListeners.length === 0)
2658             {
2659                 return;
2660             }
2661
2662             msg = hwc.getMessageByID(msgId);
2663             for (i = 0; i < hwc._messageListeners.length; i+
+)
2664             {
2665                 filter = hwc._messageListenerFilters[i];
2666                 if (filter !== null && filter !== undefined)
2667                 {
2668                     if( msg === null )
2669                     {
2670                         // a null message should pass no filter
2671                         continue;
2672                     }
2673                     if (filter.sender !== null && filter.sender !==
undefined)
2674                     {
2675                         if (msg.getSender().toLowerCase() !==
filter.sender.toLowerCase()) {
2676                             continue;
2677                         }
2678                     }
2679
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2680         if (filter.subject !== null &&
filter.subject !== undefined)
2681         {
2682             if (msg.getSubject() !== filter.subject)
{
2683                 continue;
2684             }
2685         }
2686
2687         if (filter.moduleId !== null &&
filter.moduleId !== undefined)
2688         {
2689             if (msg.getModuleId() !== filter.ModuleId)
{
2690                 continue;
2691             }
2692         }
2693
2694         if (filter.version !== null &&
filter.version !== undefined)
2695         {
2696             if (msg.getVersion() !== filter.version)
{
2697                 continue;
2698             }
2699         }
2700
2701         if (filter.isRead !== null && filter.isRead !==
undefined)
2702         {
2703             if (msg.getRead() !== filter.isRead) {
2704                 continue;
2705             }
2706         }
2707
```

```
2708         if (filter.processed !== null &&
filter.processed !== undefined)
2709             {
2710                 if (msg.getProcessed() !== filter.processed)
{
2711                     continue;
2712                 }
2713             }
2714         }
2715
2716         containingObject =
hwc._messageListenerContainingObjects[i];
2717         if (containingObject !== null &&
containingObject !== undefined)
2718             {
2719                 hwc._messageListeners[i].call(containingObject,
flag, msgId);
2720             }
2721         else
2722             {
2723                 hwc._messageListeners[i](flag, msgId);
2724             }
2725         }
2726     } finally {
2727         hwc.traceLeavingMethod("hwc._messageListenerNotification");
2728     }
2729 };
2730
2731 /**
2732  * Registers a message listener.
2733  *
2734  * @public
2735  * @memberOf hwc
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2736      * @param {hwc.MessageFilter} filters The message filter
that message events must pass to get passed to the {@link
anonymous.MessageListener}.

2737      * If no filter is desired, then null can be used for this
parameter.

2738      * @param {anonymous.MessageListener} MessageListener The
callback function for message changes.

2739      * @param {Object} [containingObject] The containing object
of the message listener. If a message listener callback function

2740      * references variables in its containing object, then the
containing object should be passed to this function.

2741      * @example

2742      * // soSomething is a global function called by the
listener callback.

2743      * var doSomething = function()

2744      * {

2745      *     alert("New message!");

2746      * }

2747      * // messageListener is the callback function passed to
hwc.addMessageListener.

2748      * var messageListener = function( flag, messageId )

2749      * {

2750      *     if( flag == hwc.MSG_ADDED )

2751      *     {

2752      *         doSomething();

2753      *     }

2754      * }

2755      * // We do not want to filter the message events the
listener will get invoked for, so pass null for the first
parameter.

2756      * hwc.addMessageListener( null, messageListener );

2757      *

2758      * @example

2759      * // someObject is an object that will contain the
listener callback as well as a variable referenced by the callback.

2760      * var someObject = {};
```

```
2761      * // doSomething is a function referenced by the callback
function.
2762      * someObject.doSomething = function()
2763      * {
2764      *     alert("New message!");
2765      * }
2766      * // messageListener is the callback that will be passed
to hwc.addMessageListener.
2767      * someObject.messageListener = function( flag,
messageId )
2768      * {
2769      *     if( flag == hwc.MSG_ADDED )
2770      *     {
2771      *         this.doSomething();
2772      *     }
2773      * }
2774      * // Create a filter so that not all message events will
invoke our callback function.
2775      * // Only events about messages with a subject of
"Subject" will trigger our callback function.
2776      * var filter = new hwc.MessageFilter( null, "Subject",
null, null, null, null);
2777      * // The callback function references a variable in its
containing object, so we need to pass in the containing object
2778      * // in addition to the filter and the callback
function.
2779      * hwc.addMessageListener( filter,
someObject.messageListener, someObject );
2780      */
2781      hwc.addMessageListener = function (filters,
MessageListener, containingObject)
2782      {
2783          hwc.traceEnteringMethod("hwc.addMessageListener");
2784          try {
2785              hwc._messageListenerFilters.push(filters);
2786              hwc._messageListeners.push(MessageListener);
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2787 hwc._messageListenerContainingObjects.push(containingObject);
2788         if (hwc._messageListeners.length === 1)
2789         {
2790             hwc.getDataFromContainer("startmsglistener");
2791         }
2792     } finally {
2793         hwc.traceLeavingMethod("hwc.addMessageListener");
2794     }
2795 };
2796
2797 /**
2798  * Removes the message listener. The two parameters passed
2799  * in to this function should match exactly the corresponding
2800  * parameters passed into {@link hwc.addMessageListener}
2801  * when the message listener was added.
2802  *
2803  * @public
2804  * @memberOf hwc
2805  * @param {anonymous.MessageListener} MessageListener The
2806  * callback for message changes.
2807  * @param {Object} [containingObject] If the containing
2808  * object was given to {@link hwc.addMessageListener} when the message
2809  * listener was added, then it also must be passed into
2810  * this function.
2811  * @example
2812  * // soSomething is a global function called by the
2813  * listener callback.
2814  * var doSomething = function()
2815  * {
2816  *     alert("New message!");
2817  * }
2818  * // messageListener is the callback function passed to
2819  * hwc.addMessageListener.
2820  * var messageListener = function( flag, messageId )
```

```
2814     * {
2815     *     if( flag == hwc.MSG_ADDED )
2816     *     {
2817     *         doSomething();
2818     *     }
2819     * }
2820     * // We do not want to filter the message events the
2821     * listener will get invoked for, so pass null for the first
2822     * parameter.
2823     * hwc.addMessageListener( null, messageListener );
2824     * // If we want to remove the listener at some other point,
2825     * use the following line of code:
2826     * hwc.removeMessageListener( messageListener );
2827     *
2828     * @example
2829     * // someObject is an object that will contain the
2830     * listener callback as well as a variable referenced by the callback.
2831     * var someObject = {};
2832     * // doSomething is a function referenced by the callback
2833     * function.
2834     * someObject.doSomething = function()
2835     * {
2836     *     alert("New message!");
2837     * }
2838     * // messageListener is the callback that will be passed
2839     * to hwc.addMessageListener.
2840     * someObject.messageListener = function( flag,
2841     * messageId )
2842     * {
2843     *     if( flag == hwc.MSG_ADDED )
2844     *     {
2845     *         this.doSomething();
2846     *     }
2847     * }
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2841      * // Create a filter so that not all message events will
invoke our callback function.
2842      * // Only events about messages with a subject of "SI<4>"
will trigger our callback function.
2843      * var filter = new hwc.MessageFilter( null, "SI<4>", null,
null, null, null);
2844      * // The callback function references a variable in its
containing object, so we need to pass in the containing object
2845      * // in addition to the filter and the callback
function.
2846      * hwc.addMessageListener( filter,
someObject.messageListener, someObject );
2847      * // If we want to remove the listener at some other point,
use the following line of code:
2848      * hwc.removeMessageListener( messageListener,
someObject );
2849      */
2850      hwc.removeMessageListener = function (MessageListener,
containingObject)
2851      {
2852          var i;
2853          hwc.traceEnteringMethod("hwc.removeMessageListener");
2854          try {
2855              if (hwc._messageListeners.length === 0) {
2856                  return;
2857              }
2858
2859              for (i = 0; i < hwc._messageListeners.length; i+
+)
2860              {
2861                  if (hwc._messageListeners[i]+" " ===
MessageListener+" " &&
2862                      hwc._messageListenerContainingObjects[i] ===
containingObject)
2863                  {
2864                      hwc._messageListeners.splice(i, 1);
2865                      hwc._messageListenerFilters.splice(i, 1);
```



```

2866         hwc._messageListenerContainingObjects.splice(i,
2867         1);
2868         if (hwc._messageListeners.length === 0)
2869         {
2870         hwc.getDataFromContainer("stopmsglistener");
2871         }
2872         return;
2873     }
2874     } finally {
2875     hwc.traceLeavingMethod("hwc.removeMessageListener");
2876     }
2877     };
2878
2879     /**
2880     * A constant indicating that a message needs to be
2881     * refreshed. Used in {@link anonymous.MessageListener} callback
2882     * functions.
2883     * @type number
2884     */
2885     hwc.MSG_REFRESH = 1;
2886     /**
2887     * A constant indicating that a message has been added.
2888     * Used in {@link anonymous.MessageListener} callback functions.
2889     * @type number
2890     */
2891     hwc.MSG_ADDED = 2;
2892     /**
2893     * A constant indicating that a message has been updated.
2894     * Used in {@link anonymous.MessageListener} callback functions.
2895     * @type number
2896     */
2897     hwc.MSG_UPDATED = 3;

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2894      /**
2895      * A constant indicating that a message has been removed.
2896      * Used in {@link anonymous.MessageListener} callback functions.
2897      * @type number
2898      */
2899      hwc.MSG_REMOVED = 4;
2900      /**
2901      * A constant indicating a message has normal priority.
2902      * Used in {@link hwc.Message}.
2903      * @type number
2904      */
2905      hwc.MSG_PRIORITY_NORMAL = 1;
2906      /**
2907      * A constant indicating a message has high priority. Used
2908      * in {@link hwc.Message}.
2909      * @type number
2910      */
2911      hwc.MSG_PRIORITY_HIGH = 3;
2912      /**
2913      * A sample {@link anonymous.MessageListener} callback
2914      * function.
2915      * @param {number} flag A number indicating which message
2916      * event occurred (will be one of MSG_* constants).
2917      * @param {number} msgId The message id of the affected
2918      * message.
2919      */
2920      hwc.sample_MessageListener = function (flag, msgId) {
2921      };
```

```

2922      * @public
2923      * @memberOf hwc
2924      * @param {hwc.MessageFilter} [messageFilter] A filter that
all returned messages will pass.
2925      * If you do not want to filter based on a certain attribute,
use null for that attribute when creating the filter.
2926      * If you do not want to filter at all, pass in null for
this parameter or do not pass in this parameter at all.
2927      * @param {boolean} [completeList] If this parameter is set
to true, then all messages will be returned.
2928      * If this parameter is set to false or if it is not set,
then if there is a default hybrid app only the messages belonging
2929      * to the default hybrid app will be returned (and if there
is no default hybrid app all messages will be returned).
2930      *
2931      * @returns {hwc.Message[]} An array of {@link hwc.Message}
objects - the received messages.
2932      * @example
2933      * // get all messages that have the subject "a
subject".
2934      * var filter = new hwc.MessageFilter( null, "a subject",
null, null, null, null );
2935      * var messages = hwc.getAllMessages(filter);
2936      *
2937      * @example
2938      * // Get all messages without filtering, but if there is a
default hybrid app only return its messages.
2939      * var messages = hwc.getAllMessages();
2940      *
2941      * @example
2942      * // Get all messages (without filtering) for all hybrid
apps, even if there is a default hybrid app.
2943      * var messages = hwc.getAllMessages( null, true );
2944      */
2945      hwc.getAllMessages = function (filters, completeList) {
2946          var filtersUrlString, messages, i, message,
formattedCompleteList, response, messageInstances;

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2947
2948     hwc.traceEnteringMethod("hwc.getAllMessages");
2949     formattedCompleteList = false;
2950     response = "";
2951     messageInstances = [];
2952
2953     if ( completeList )
2954     {
2955         formattedCompleteList = true;
2956     }
2957
2958     try {
2959         // Create filter url argument
2960         filtersUrlString = "";
2961         if( filters )
2962         {
2963             if( filters.sender !== undefined &&
filters.sender !== null )
2964             {
2965                 filtersUrlString = filtersUrlString +
"&filtermessagesender=" + encodeURIComponent(filters.sender);
2966             }
2967             if( filters.subject !== undefined &&
filters.subject !== null && filters.subject !== undefined)
2968             {
2969                 filtersUrlString = filtersUrlString +
"&filtermessagesubject=" + encodeURIComponent(filters.subject);
2970             }
2971             if( filters.moduleId !== undefined &&
filters.moduleId !== null )
2972             {
2973                 filtersUrlString = filtersUrlString +
"&filtermessagemoduleid=" + encodeURIComponent(filters.moduleId);
2974             }

```

```
2975         if( filters.version !== undefined &&
filters.version !== null )
2976         {
2977             filtersUrlString = filtersUrlString +
"&filtermessageversion=" + encodeURIComponent(filters.version);
2978         }
2979         if( filters.isRead !== undefined &&
filters.isRead !== null )
2980         {
2981             filtersUrlString = filtersUrlString +
"&filtermessageisread=" + encodeURIComponent(filters.isRead);
2982         }
2983         if( filters.processed !== undefined &&
filters.processed !== null )
2984         {
2985             filtersUrlString = filtersUrlString +
"&filtermessageisprocessed=" +
encodeURIComponent(filters.processed);
2986         }
2987     }
2988
2989     filtersUrlString += "&getcompletelist=" +
formattedCompleteList;
2990
2991     response = hwc.getDataFromContainer("getmessages",
filtersUrlString);
2992
2993     if (response !== null && response !== undefined &&
response !== "")
2994     {
2995         messages = JSON.parse(response);
2996         for (i=0; i<messages.length; i++){
2997             message = messages[i];
2998             messageInstances[i] = new
hwc.Message(message.id, new Date(message.milliseconds),
message.iconIndex, message.sender, message.isRead,
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
2999         message.isProcessed, message.priority,
message.subject, message.module, message.version);
3000     }
3001 }
3002
3003     }catch (ex){
3004         hwc.log("messages.getAll error:" + ex.message,
"ERROR", false);
3005     } finally {
3006         hwc.traceLeavingMethod("hwc.getAllMessages");
3007     }
3008
3009     return messageInstances;
3010 };
3011
3012 /**
3013  * Gets a {@link hwc.Message} object with the given message
ID.
3014  *
3015  * @public
3016  * @memberOf hwc
3017  * @param {number} msgId The message ID of the message to
get.
3018  *
3019  * @returns {hwc.Message} A message object, or null if no
message with given ID.
3020  * @example
3021  * // A message listener is one place that would likely
need to call hwc.getMessageByID.
3022  * var messageListener = function( flag, messageID )
3023  * {
3024  *     // Since the callback function only gets the
messageID, not the message itself, if we want
3025  *     // more information about the message we must call
hwc.getMessageByID.
```

```

3026      *      var message = hwc.getMessageByID( messageId );
3027      *      if( message.getSubject() == "a special subject" )
3028      *      {
3029      *          alert( "An event ocured for a special
message!" );
3030      *      }
3031      * }
3032      * hwc.addMessageListener( null, messageListener );
3033      */
3034      hwc.getMessageByID = function (msgId)
3035      {
3036          var response, messageInstance, message;
3037
3038          hwc.traceEnteringMethod("hwc.getMessageByID");
3039          response = "";
3040          messageInstance = null;
3041
3042          try {
3043              response = hwc.getDataFromContainer("getmessagebyid",
"&msgid=" + msgId);
3044
3045              if (response !== null && response !== undefined &&
response !== "")
3046              {
3047                  message = JSON.parse(response);
3048                  messageInstance = new hwc.Message(message.id, new
Date(message.milliseconds), message.iconIndex, message.sender,
message.isRead,
3049                  message.isProcessed, message.priority,
message.subject, message.module, message.version);
3050              }
3051          }catch (ex){
3052              hwc.log("messages.getMessageByID error:" + ex.message,
"ERROR", false);
3053          } finally {

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
3054         hwc.traceLeavingMethod("hwc.getMessageByID");
3055     }
3056
3057     return messageInstance;
3058 };
3059
3060 /**
3061  * Updates the message read status.
3062  *
3063  * @public
3064  * @memberOf hwc
3065  * @param {number} msgId The id of message to update the
3066  * read status for.
3067  * @param {boolean} status Whether the message will be set
3068  * to read (true) or unread (false).
3069  * @example
3070  * // set all messages as read
3071  * var messages = hwc.getAllMessages();
3072  * for( var index = 0; index < messages.length; index+
3073  * )
3074  * {
3075  *     hwc.updateMessageRead( messages[index].getMessageId(), true );
3076  * }
3077  */
3078 hwc.updateMessageRead = function (msgId, status)
3079 {
3080     var updateParms;
3081     hwc.traceEnteringMethod("hwc.updateMessageRead");
3082     try {
3083         updateParms = "&msgid=" + msgId + "&msgfield=read" +
3084         "&status=" + status;
3085         hwc.getDataFromContainer("updatemessage",
3086         updateParms);
```



```
3082         } catch (ex){
3083             hwc.log("Message.updateMsgRead error:" + ex.message,
3084                 "ERROR", false);
3085         } finally {
3086             hwc.traceLeavingMethod("hwc.updateMessageRead");
3087         }
3088     };
3089     /**
3090     * Updates the message processed status.
3091     *
3092     * @public
3093     * @memberOf hwc
3094     * @param {number} msgId The id of message to update the
3095     * processed status for.
3096     * @param {boolean} status Whether the message will be set
3097     * to processed (true) or unprocessed (false).
3098     * @example
3099     * // set all messages as processed
3100     * var messages = hwc.getAllMessages();
3101     * for( var index = 0; index < messages.length; index+
3102     + )
3103     * {
3104     *     hwc.updateMessageProcessed( messages[index].getMessageId(), true );
3105     * }
3106     */
3107     hwc.updateMessageProcessed = function (msgId, status)
3108     {
3109         var updateParms;
3110         hwc.traceEnteringMethod("hwc.updateMessageProcessed");
3111         try {
3112             updateParms = "&msgid=" + msgId +
3113                 "&msgfield=processed" + "&status=" + status;
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
3110         hwc.getDataFromContainer("updatemessage",
updateParms);
3111     } catch (ex){
3112         hwc.log("Message.updateMsgProcessed error:" +
ex.message, "ERROR", false);
3113     } finally {
3114         hwc.traceLeavingMethod("hwc.updateMessageProcessed");
3115     }
3116 };
3117
3118     /**
3119     * Removes (deletes) a message.
3120     *
3121     * @public
3122     * @memberOf hwc
3123     * @param {number} msgId The id of the message to be
removed.
3124     * @example
3125     * // remove all messages
3126     * var messages = hwc.getAllMessages();
3127     * for( var index = 0; index < messages.length; index+
+ )
3128     * {
3129     *
hwc.removeMessage( messages[index].getMessageId() );
3130     * }
3131     */
3132     hwc.removeMessage = function(msgId) {
3133         hwc.traceEnteringMethod("hwc.removeMessage");
3134         try {
3135             hwc.getDataFromContainer("removemessage", "&msgid=" +
msgId);
3136         } catch (ex){hwc.log("messages.remove error:" +
ex.message, "ERROR", false);}

```

```
3137         finally
3138     {hwc.traceLeavingMethod("hwc.removeMessage");}
3139
3140     /**
3141     * A constant indicating that a message was successfully
3142     * opened. This is a possible return value for {@link
3143     * hwc.openMessage}.
3144     * @type number
3145     */
3146     hwc.OPEN_MSG_SUCCESS = 0;
3147
3148     /**
3149     * A constant indicating that a message could not be opened
3150     * because no message with the given ID exists.
3151     * This is a possible return value for {@link
3152     * hwc.openMessage}.
3153     * @type number
3154     */
3155     hwc.OPEN_MSG_NOT_EXIST = 1;
3156
3157     /**
3158     * A constant indicating that a message could not be opened
3159     * because there was no associated hybrid app.
3160     * This is a possible return value for {@link
3161     * hwc.openMessage}.
3162     * @type number
3163     */
3164     hwc.OPEN_MSG_APP_NOT_EXIST = 2;
3165
3166     /**
3167     * A constant indicating that a message could not be opened
3168     * due to an unspecified error.
3169     * This is a possible return value for {@link
3170     * hwc.openMessage}.
3171     * @type number
3172     */
3173     hwc.OPEN_MSG_OTHER = 3;
3174 }
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
3164     /**
3165     * Launch the server initiated hybrid app associated with a
3166     * message. The hybrid app will be opened on top of the hybrid app
3167     * that is open when hwc.openMessage is called. When the
3168     * hybrid app that was opened with hwc.openMessage exits, it will exit
3169     * to the hybrid app that was open when hwc.openMessage was
3170     * called. It is possible to nest open hybrid apps, but it is
3171     * best not to have too many nested hybrid apps (eg:
3172     * recursively opening hybrid apps) because each open hybrid app
3173     * takes up device memory.
3174     *
3175     * @public
3176     * @memberOf hwc
3177     * @param {number} msgId The id of message to open.
3178     *
3179     * @returns {number} A number indicating the success or
3180     * failure of opening the message (will be one of {@link
3181     * hwc.OPEN_MSG_SUCCESS},
3182     * {@link hwc.OPEN_MSG_NOT_EXIST}, {@link
3183     * hwc.OPEN_MSG_APP_NOT_EXIST}, {@link hwc.OPEN_MSG_OTHER}).
3184     * @example
3185     * // get all messages, then open the first one
3186     * var messages = hwc.getAllMessages();
3187     * if( messages.length > 0 )
3188     * {
3189     *     hwc.openMessage( messages[0].getMessageId() );
3190     * }
3191     */
3192     hwc.openMessage = function(msgId) {
3193         var response;
3194
3195         hwc.traceEnteringMethod("hwc.openMessage");
3196
3197         try {
3198             response = hwc.getDataFromContainer("openmessage",
3199             "&msgid=" + msgId);
```

```

3191         return parseInt(response, 10);
3192     } catch (ex){
3193         hwc.log("messages.open error:" + ex.message, "ERROR",
false);
3194     } finally {
3195         hwc.traceLeavingMethod("hwc.openMessage");
3196     }
3197 };
3198
3199 /**
3200  * This function takes care of handling the XML HTTP
request to communicate with the HWC native code on the different
platforms.
3201  *
3202  * @private
3203  * @memberOf hwc
3204  *
3205  * @param {string} queryType A string indicating the type
of query being sent to the native code.
3206  * This parameter must match up with a constant defined in
the native code of the HWC.
3207  * @param {string} urlParams A string of parameters for the
query, in a format such that it can
3208  * be added directly to the url.
3209  *
3210  * @returns {string} The response text of the request.
3211  * @example
3212  * // This example is an excerpt from
hwc.getInstalledApps. There are many examples of how to use this
function in this file.
3213  * response = hwc.getDataFromContainer("getinstalledapps",
"&getcompletelist=true");
3214  * if (response != null && response != undefined &&
response != "")
3215  * {
3216  *     var apps = JSON.parse(response);

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
3217      *      for(var i=0; i<apps.length; i++) {
3218      *          var app = apps[i];
3219      *          installedApps[i] = new hwc.HybridApp(app.moduleId,
3220      *          app.version, app.displayName, app.iconIndex,
3221      *          hwc.createCustomIconObject(app.defaultCustomIcon, app.moduleId,
3222      *          app.version, hwc.DEFAULT_CUSTOM_ICON_INDEX),
3223      *          hwc.createCustomIconList(app.customIconList, app.moduleId,
3224      *          app.version));
3225      *      }
3226      *  }
3227      */
3228      hwc.getDataFromContainer = function( queryType,
3229      urlParams)
3230      {
3231      var response, xmlhttp;
3232      hwc.traceEnteringMethod("hwc.getDataFromContainer");
3233      response = "";
3234      if (urlParams === null || urlParams === undefined) {
3235      urlParams = "";
3236      }
3237      try {
3238      if (hwc.isWindowsMobile()) {
3239      xmlhttp = hwc.getXMLHTTPRequest();
3240      xmlhttp.open("GET", "/sup.amp?querytype=" +
3241      queryType + "&" + hwc.versionURLParam + "&" + urlParams, false );
3242      xmlhttp.send("");
3243      response = xmlhttp.responseText;
3244      }
3245      else if (hwc.isAndroid()) {
3246      response = _HWC.getData("http://localhost/sup.amp?
3247      querytype=" + queryType + "&" + hwc.versionURLParam + urlParams);
3248      }
3249      }
```

```
3244         else if (hwc.isBlackBerry()) {
3245             // CR661210 and NA3-2487
3246             if (hwc.isClosed()) {
3247                 return;
3248             }
3249
3250             xmlhttp = hwc.getXMLHttpRequest();
3251             xmlhttp.open("POST", "http://localhost/sup.amp?
querytype=" + queryType + "&" + hwc.versionURLParam + urlParams,
false);
3252             xmlhttp.send();
3253             response = xmlhttp.responseText;
3254         }
3255         else if (hwc.isIOS()) {
3256             xmlhttp = hwc.getXMLHttpRequest();
3257             xmlhttp.open("GET", "http://localhost/sup.amp?
querytype=" + queryType + "&" + hwc.versionURLParam + urlParams,
false);
3258             try
3259             {
3260                 xmlhttp.send("");
3261             }
3262             catch (ex)
3263             {
3264                 if (ex.message.search(/XMLHttpRequest Exception
101/) === -1)
3265                 {
3266                     throw ex;
3267                 }
3268             }
3269             response = xmlhttp.responseText;
3270         }
3271         return response;
3272     }
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
3273         catch (ex1)
3274         {
3275             hwc.log( "hwc.getDataFromContainer error: " +
3276                 ex1.message, "ERROR", false);
3277         } finally {
3278             hwc.traceLeavingMethod("hwc.getDataFromContainer");
3279         }
3280     };
3281     /**
3282     * This function takes care of handling the XML HTTP
3283     request to communicate with the HWC native code on different
3284     platforms.
3285     *
3286     * @private
3287     * @memberOf hwc
3288     * @param {string} queryType Indicates the type of query
3289     being sent to the native code.
3290     * This parameter must match up with a constant defined in
3291     the native code of the HWC.
3292     * @param {string} data Data to be sent with the
3293     request.
3294     *
3295     * @returns {string} The response text of the request.
3296     */
3297     hwc.postDataToContainer = function( queryType, data)
3298     {
3299         var response, xmlhttp;
3300         response = "";
3301         try
3302         {
3303             if (hwc.isWindowsMobile()) {
3304                 xmlhttp = hwc.getXMLHttpRequest();
```



```

3301         xmlhttp.open("POST", "/sup.amp?querytype=" +
queryType + "&" + hwc.versionURLParam, false);
3302         xmlhttp.send(data);
3303         response = xmlhttp.responseText;
3304     }
3305     else if (hwc.isAndroid()) {
3306         response = _HWC.postData("http://localhost/
sup.amp?querytype=" + queryType + "&" + hwc.versionURLParam, data);
3307     }
3308     else if (hwc.isBlackBerry()) {
3309         // CR661210 and NA3-2487
3310         if (hwc.isClosed()) {
3311             return;
3312         }
3313
3314         xmlhttp = hwc.getXMLHttpRequest();
3315         xmlhttp.open("POST", "http://localhost/sup.amp?
querytype=" + queryType + "&" + hwc.versionURLParam, false);
3316         xmlhttp.send(data);
3317         response = xmlhttp.responseText;
3318     }
3319     else if (hwc.isIOS()) {
3320         xmlhttp = hwc.getXMLHttpRequest();
3321         xmlhttp.open("POST", "http://localhost/sup.amp?
querytype=" + queryType + "&" + hwc.versionURLParam, false);
3322         try
3323         {
3324             xmlhttp.send(data);
3325         }
3326         catch (ex)
3327         {
3328             if (ex.message.search(/XMLHttpRequest Exception
101/) === -1)
3329                 {

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
3330         throw ex;
3331     }
3332 }
3333     response = xmlhttp.responseText;
3334 }
3335     return response;
3336 }
3337     catch (ex1)
3338     {
3339         hwc.log( "hwc.postDataToContainer error: " +
3340             ex1.message, "ERROR", false);
3341     };
3342
3343     var partialRequestUrl = null;
3344
3345     /**
3346     * Gets a URL that can be used to get resources from the
3347     HWC.
3348     *
3349     * @private
3350     * @memberOf hwc
3351     * @param {string} queryType The type of query
3352     * @param {string} urlParams Additional parameters to send
3353     with the request. Must be formatted such that it can be appended to
3354     the url
3355     * (eg: "firstParam=value1&secondParam=value2").
3356     *
3357     * @returns {string} A URL that can be used to access
3358     resources.
3359     */
3360     getRequestUrl = function ( queryType, urlParams )
3361     {
```

```

3358         // Lazy load to prevent platform identification
errors
3359         if (!partialRequestUrl)
3360         {
3361             partialRequestUrl = hwc.isWindowsMobile() ? "/"
sup.amp?querytype=" :
3362             hwc.isAndroid() ?
3363             ( window.location.protocol + "://" +
window.location.hostname + "/" +
window.location.pathname.split( '/' ) [1] + "/sup.amp/
querytype=" ) :
3364             hwc.isBlackBerry() || hwc.isIOS() ? "http://
localhost/sup.amp?querytype=" :
3365             "";
3366         }
3367
3368         return partialRequestUrl + queryType + "&" +
hwc.versionURLParam + (urlParams ? '&' : "") + urlParams;
3369     };
3370
3371     /**
3372     * Represents a Media Cache. This object gives the option
to use the cache when accessing .
3373     *
3374     * @classdesc
3375     * @public
3376     * @memberOf hwc
3377     * @static
3378     */
3379     hwc.MediaCache = {};
3380
3381     /**
3382     * hwc.MediaCache.Policy An object containing constants
representing the different caching policies.
3383     * @memberOf hwc.MediaCache
3384     */

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
3385     hwc.MediaCache.Policy = {};  
3386  
3387     /**  
3388     * hwc.MediaCache.Policy.SERVER_FIRST Use server first  
3389     * policy: requests will only be served from the cache if the server is  
3390     * unavailable.  
3391     * @type {string}  
3392     * @memberOf hwc.MediaCache  
3393     */  
3394     hwc.MediaCache.Policy.SERVER_FIRST = "ServerFirst";  
3395  
3396     /**  
3397     * hwc.MediaCache.Policy.CACHE_FIRST Use cache first  
3398     * policy: requests will be served from the cache if possible.  
3399     * @type {string}  
3400     * @memberOf hwc.MediaCache  
3401     */  
3402     hwc.MediaCache.Policy.CACHE_FIRST = "CacheFirst";  
3403  
3404     /**  
3405     * Creates a media cache URL for the resource. The cache  
3406     * first policy will be used if no policy is specified.  
3407     * @public  
3408     * @memberOf hwc.MediaCache  
3409     * @param {string} resourceUrl The URL to the resource  
3410     * @param {hwc.MediaCache.Policy} [policy] The optional  
3411     * cache policy to use.  
3412     * If set, it must be either {@link  
3413     * hwc.MediaCache.Policy.SERVER_FIRST} or {@link  
3414     * hwc.MediaCache.Policy.CACHE_FIRST}.  
3415     * Default policy is cache first.  
3416     * @returns {string} The URL that can be used to access the  
3417     * resource with the specified caching policy.  
3418     * @example
```

```

3412      * // This line creates a url that can be used to retrieve
the picture from the cache if possible, and from the server
otherwise.

3413      * var mediaCacheURL = hwc.MediaCache.getUrl( "http://
yourserver.com/Pictures/pentagon.jpg",
hwc.MediaCache.Policy.CACHE_FIRST );

3414      * // The following function adds a picture to the page.
Since the mediaCacheURL variable is used for the url, the picure will
be

3415      * // retrieved from the cache if possible.

3416      * var addPicFromMediaCache = function()

3417      * {

3418      *     // Create the image element.

3419      *     var image = document.createElement( "img" );

3420      *     // Set the source of the image to the media cache
URL.

3421      *     image.setAttribute( 'src', mediaCacheURL );

3422      *     // Add the image element to the page.

3423      *     document.body.appendChild( image );

3424      * }

3425      *

3426      * @example

3427      * // This line creates a url that can be used to retrieve
the picture from the server if it is available, or the cache
otherwise.

3428      * var mediaCacheURL_serverFirst =
hwc.MediaCache.getUrl( "http://yourserver.com/Pictures/
pentagon.jpg", hwc.MediaCache.Policy.SERVER_FIRST );

3429      * // The following function adds a picture to the page.
Since the mediaCacheURL_serverFirst variable is used for the url, the
picture will be gotten

3430      * // from the server if the server is available, and from
the cache otherwise.

3431      * var addPicFromMediaCache_ServerFirst = function()

3432      * {

3433      *     // Create the image element.

3434      *     var image = document.createElement( "img" );

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
3435      *      // Set the source of the image to the media cache
URL.
3436      *      image.setAttribute( 'src',
mediaCacheURL_serverFirst );
3437      *      // Add the image element to the page.
3438      *      document.body.appendChild( image );
3439      *  }
3440      *
3441      */
3442      hwc.MediaCache.getUrl = function ( returnUrl, policy )
{
3443          hwc.traceEnteringMethod("hwc.MediaCache.getUrl");
3444          try {
3445              policy = policy ? policy :
hwc.MediaCache.Policy.CACHE_FIRST;
3446              return getRequestUrl( "mediacache", "url=" +
encodeURIComponent( returnUrl)
3447                  + "&policy=" + policy + "&bustCache=" +
Math.random() );
3448          } finally {
3449              hwc.traceLeavingMethod("hwc.MediaCache.getUrl");
3450          }
3451      };
3452
3453      /**
3454      * Represents an E2E Trace. This object is used for
debugging and analysis.
3455      * @classdesc
3456      */
3457      hwc.e2eTrace = {};
3458
3459      hwc.e2eTrace.TraceLevel = {};
3460
3461      /**
```

```

3462      * A constant indicating a high level of detail for the
trace.
3463      * Use this level for functional analysis and detailed
functional logging and tracing.
3464      * @type string
3465      * @memberOf hwc.e2eTrace
3466      */
3467      hwc.e2eTrace.TraceLevel.HIGH = "HIGH";
3468
3469      /**
3470      * A constant indicating a low level of detail for the
trace.
3471      * Use this level for response-time-distribution analysis:
see how much time is spent on each server component to find
bottlenecks.
3472      * @type string
3473      * @memberOf hwc.e2eTrace
3474      */
3475      hwc.e2eTrace.TraceLevel.LOW = "LOW";
3476
3477      /**
3478      * A constant indicating a medium level of detail for the
trace.
3479      * Use this level for performance analysis (performance
traces are triggered on server-side).
3480      * @type string
3481      * @memberOf hwc.e2eTrace
3482      */
3483      hwc.e2eTrace.TraceLevel.MEDIUM = "MEDIUM";
3484
3485      /**
3486      * Gets whether the e2e tracing has been requested to be
started.
3487      * This function returns true between calls to {@link
hwc.e2eTrace#startTrace} and {@link hwc.e2eTrace#stopTrace}.
3488      * @memberOf hwc.e2eTrace

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
3489      * @returns {boolean} True if trace is enabled, false
otherwise.
3490      */
3491      hwc.e2eTrace.isTraceEnabled = function() {
3492
hwc.traceEnteringMethod("hwc.e2eTrace.isTraceEnabled");
3493      try {
3494          return
parseBoolean(hwc.getDataFromContainer("e2etrace",
"&method=istraceenabled"));
3495      } finally {
3496
hwc.traceLeavingMethod("hwc.e2eTrace.isTraceEnabled");
3497      }
3498      };
3499
3500      /**
3501      * Sets the passport e2eTrace level. This function must be
called before {@link hwc.e2eTrace#startTrace}.
3502      *
3503      * @memberOf hwc.e2eTrace
3504      * @param {string} The trace level. Must be one of {@link
hwc.e2eTrace.TraceLevel.LOW}, {@link
hwc.e2eTrace.TraceLevel.MEDIUM}, or
3505      * {@link hwc.e2eTrace.TraceLevel.HIGH}.
3506      */
3507      hwc.e2eTrace.setTraceLevel = function(level) {
3508
hwc.traceEnteringMethod("hwc.e2eTrace.setTraceLevel");
3509      try {
3510          hwc.getDataFromContainer("e2etrace",
"&method=settracelevel&level=" + level);
3511      } finally {
3512
hwc.traceLeavingMethod("hwc.e2eTrace.setTraceLevel");
3513      }
3514      };
```



```
3515
3516     /**
3517     * Starts tracing user actions and requests. Before this
3518     * function is called, the trace level must be set with {@link
3519     * hwc.e2eTrace#setTracelevel}.
3520     * @memberOf hwc.e2eTrace
3521     */
3522     hwc.e2eTrace.startTrace = function() {
3523         hwc.traceEnteringMethod("hwc.e2eTrace.startTrace");
3524         try {
3525             hwc.getDataFromContainer("e2etrace",
3526                 "&method=starttrace");
3527         } finally {
3528             hwc.traceLeavingMethod("hwc.e2eTrace.startTrace");
3529         }
3530     };
3531
3532     /**
3533     * Stops tracing user actions and requests.
3534     * @memberOf hwc.e2eTrace
3535     */
3536     hwc.e2eTrace.stopTrace = function() {
3537         hwc.traceEnteringMethod("hwc.e2eTrace.stopTrace");
3538         try {
3539             hwc.getDataFromContainer("e2etrace",
3540                 "&method=stoptrace");
3541         } finally {
3542             hwc.traceLeavingMethod("hwc.e2eTrace.stopTrace");
3543         }
3544     };
3545
3546     /**
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
3543      * Upload the Business Transaction XML (BTX) to the
server.

3544      * To upload, the SAP Solution Manager URL must be set in
SAP Control Center configuration.

3545      * @memberOf hwc.e2eTrace

3546      * @returns {boolean} True if the upload is successful,
false otherwise.

3547      */

3548      hwc.e2eTrace.uploadTrace = function() {

3549          hwc.traceEnteringMethod("hwc.e2eTrace.uploadTrace");

3550          try {

3551              return
parseBoolean(hwc.getDataFromContainer("e2etrace",
"&method=uploadtrace"));

3552          } finally {

3553              hwc.traceLeavingMethod("hwc.e2eTrace.uploadTrace");

3554          }

3555          };

3556

3557      /**

3558      * Represents the Performance Manager.

3559      * @classdesc

3560      * @memberOf hwc

3561      * @example

3562      * // Start performance collection.

3563      * if (hwc.perf.isEnabled())

3564      * {

3565      *     hwc.perf.stopInteraction();

3566      * }

3567      *

3568      * hwc.perf.startInteraction('someinteraction');

3569      *

3570      * hwc.perf.startInterval('IntervalName',
'CustomType'); // Start an optional interval.
```

```
3571      *
3572      * // Stop performance collection.  Logs will be
written.
3573      * if (hwc.perf.isEnabled())
3574      * {
3575      *   hwc.perf.stopInterval('IntervalName'); // Stop an
optional interval.
3576      *   hwc.perf.stopInteraction();
3577      * }
3578      */
3579      hwc.perf = {};
3580
3581      /**
3582      * Gets whether the performance agent is enabled.
3583      * @memberOf hwc.perf
3584      * @returns {boolean} True if the performance agent is
enabled, false otherwise.
3585      */
3586      hwc.perf.isEnabled = function() {
3587          hwc.traceEnteringMethod("hwc.perf.isEnabled");
3588          try {
3589              return parseBoolean(hwc.getDataFromContainer("perf",
"&method=isEnabled"));
3590          } finally {
3591              hwc.traceLeavingMethod("hwc.perf.isEnabled");
3592          }
3593      };
3594
3595      /**
3596      * Starts the interaction.
3597      * @memberOf hwc.perf
3598      * @param {string} interactionName The name of the
interaction.
3599      */
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
3600     hwc.perf.startInteraction = function(interactionName) {
3601
3602         hwc.traceEnteringMethod("hwc.perf.startInteraction");
3603         try {
3604             hwc.getDataFromContainer("perf",
3605                 "&method=startinteraction&interactionname=" +
3606                 encodeURIComponent(interactionName));
3607         } finally {
3608             hwc.traceLeavingMethod("hwc.perf.startInteraction");
3609         }
3610     };
3611
3612     /**
3613     * Stops the interaction.
3614     * @memberOf hwc.perf
3615     */
3616     hwc.perf.stopInteraction = function() {
3617         hwc.traceEnteringMethod("hwc.perf.stopInteraction");
3618         try {
3619             hwc.getDataFromContainer("perf",
3620                 "&method=stopinteraction");
3621         } finally {
3622             hwc.traceLeavingMethod("hwc.perf.stopInteraction");
3623         }
3624     };
3625
3626     /**
3627     * Starts an interval.
3628     * @memberOf hwc.perf
3629     * @param {string} intervalName The name of the
3630     * interval.
3631     * @param {string} intervalType The type of the interval.
3632     */
```

```
3627     */
3628     hwc.perf.startInterval = function(intervalName,
3629     intervalType) {
3630         hwc.traceEnteringMethod("hwc.perf.startInterval");
3631         try {
3632             hwc.getDataFromContainer("perf",
3633             "&method=startinterval&intervalname=" +
3634             encodeURIComponent(intervalName) + "&intervaltype=" +
3635             encodeURIComponent(intervalType));
3636         } finally {
3637             hwc.traceLeavingMethod("hwc.perf.startInterval");
3638         }
3639     };
3640
3641     /**
3642     * Stops the interval.
3643     * @memberOf hwc.perf
3644     * @param {string} intervalName The name of the
3645     interval.
3646     */
3647     hwc.perf.stopInterval = function(intervalName) {
3648         hwc.traceEnteringMethod("hwc.perf.stopInterval");
3649         try {
3650             hwc.getDataFromContainer("perf",
3651             "&method=stopinterval&intervalname=" +
3652             encodeURIComponent(intervalName));
3653         } finally {
3654             hwc.traceLeavingMethod("hwc.perf.stopInterval");
3655         }
3656     };
3657
3658     /**
3659     * Internal function to parse a boolean
3660     * @private
3661     */
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
3655     function parseBoolean(val)
3656     {
3657         return val === 'true';
3658     }
3659 }) (hwc);
3660
3661 /**
3662  * Used to group anonymous objects and callback functions used
3663  * as method parameters. Methods and fields in this
3664  * namespace cannot be instantiated. Used for API docs
3665  * generation only.
3666  * @namespace
3667  */
3668
3669 anonymous = (typeof anonymous === "undefined" || !
3670 anonymous) ? {} : anonymous; // SUP 'namespace'
3671
3672 /**
3673  * Callback function that will be invoked when the connection
3674  * state changes. Connection listeners can be added with {@link
3675  * hwc.addConnectionListener}.
3676  *
3677  * @name anonymous.ConnectionStateListener
3678  *
3679  * @param {number} event A number indicating the event that
3680  * occurred (will be {@link hwc.CONNECTED} or {@link
3681  * hwc.DISCONNECTED}).
3682  *
3683  * @param {number} errorCode An error code (0 indicating
3684  * success).
3685  *
3686  * @param {string} errorMessage Text of the error message.
3687  * Will be empty if there is no error.
3688  *
3689  * @function
3690  */
```

```

3682      * Callback function that will be invoked when events are
logged to the event log. Log listeners can be added with {@link
hwc.addLogListener}.

3683      *

3684      * @name anonymous.LogListener

3685      *

3686      * @param {number} milliseconds The date of the log message
represented in milliseconds.

3687      * @param {number} event A number that represents which
category this event falls under (It will be one of {@link
hwc.CONNECTION_ERROR},

3688      * {@link hwc.CONNECTION_OTHER}, {@link
hwc.CONNECTION_CONNECTED}, {@link hwc.CONNECTION_DISCONNECTED},
{@link hwc.CONNECTION_RETRIEVED_ITEMS}).

3689      * @param {string} optionalString The string carrying the
message of the log event.

3690      *

3691      * @function

3692      */

3693

3694      /**

3695      * Callback function that will be invoked on hybrid app
installation events. App installation listeners can be added with

3696      * {@link hwc.addAppInstallationListener}.

3697      *

3698      * @name anonymous.AppInstallationListener

3699      *

3700      * @param {number} event A number indicating the event (will
be either {@link hwc.INSTALLATION_BEGIN} or {@link
hwc.INSTALLATION_END}).

3701      * @param {string} moduleId The module ID of the hybrid app
the event is about.

3702      * @param {string} version The version of the hybrid app the
event is about.

3703      * @param {string} moduleName The display name of the hybrid
app the event is about.

3704      *

3705      * @function

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
3706     */
3707
3708     /**
3709     * Callback function that will be invoked when push
3710     * notifications are available.
3711     *
3712     * @name anonymous.PushNotificationListener
3713     *
3714     * @param {Array} notifications An array of notifications.
3715     *
3716     * @returns {number} A number indicating whether other push
3717     * notification listeners should be called after this one.
3718     * Must be either {@link hwc.NOTIFICATION_CANCEL} (if no more
3719     * listener callbacks should be called) or {@link
3720     * hwc.NOTIFICATION_CONTINUE}
3721     * (if more listener callbacks should be called).
3722     *
3723     * Callback function that will be invoked on hybrid app
3724     * installation events.
3725     * @name anonymous.AppInstallationListener
3726     * @param {Integer} event Installation flags
3727     * including, BEGIN(1), END(2), FAIL(3)
3728     * @param {String} moduleId Optional Module Id
3729     * @param {String} version Optional Module
3730     * version
3731     * @param {String} moduleName Optional Module display
3732     * name
3733     * @param {String} designerVersion Optional Version of
3734     * designer used to create app
3735     * @param {String} containerVersion Optional Version of
3736     * hybrid web container
3737     * @callback
3738     * @function
3739     */
3740
3741     /**
```



```

3732      * Callback function that will be invoked on hybrid app
installation events.
3733      * @name anonymous.AppInstallationListener
3734      * @param {Integer} event           Installation flags
including, BEGIN(1), END(2)
3735      * @param {String} moduleId        Optional Module Id
3736      * @param {String} version         Optional Module
version
3737      * @param {String} moduleName      Optional Module
display name
3738      * @callback
3739      * @function
3740      */
3741
3742      /**
3743      * Callback function that will be invoked on hybrid app
events.
3744      * Application listeners can be added with {@link
hwc.addAppListener}.
3745      *
3746      * @name anonymous.ApplicationListener
3747      *
3748      * @param {number} event A number indicating what event has
taken place (will be one of {@link hwc.APP_REFRESH},
3749      * {@link hwc.APP_ADDED}, {@link hwc.APP_UPDATED}, {@link
hwc.APP_REMOVED}).
3750      * @param {number} moduleId The module id of the hybrid app
the event is about.
3751      * @param {number} version module The version of the hybrid
app the event is about.
3752      *
3753      * @function
3754      */
3755
3756      /**
3757      * Callback function that will be invoked on message events.
Message listeners can be added with {@link hwc.addMessageListener}.

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
3758      *
3759      * @name anonymous.MessageListener
3760      *
3761      * @param {number} flag A number indicating which message
event occurred (will be one of {@link hwc.MSG_ADDED}, {@link
hwc.MSG_REMOVED},
3762      * {@link hwc.MSG_UPDATED}, {@link hwc.MSG_REFRESH}).
3763      * @param {number} msgId The message id of the affected
message.
3764      *
3765      * @function
3766      */
3767
```

hwc-comms.js

```
1      /**
2      * Sybase Hybrid App version 2.3.4
3      *
4      * API.js
5      * This file will not be regenerated, so it is possible to
modify it, but it
6      * is not recommended.
7      *
8      * The template used to create this file was compiled on Thu
Jun 07 14:57:11 EDT 2012
9      *
10     * Copyright (c) 2012 Sybase Inc. All rights reserved.
11     */
12
13     /**
14     * Holds all the Hybrid Web Container javascript
15     * @namespace
16     */
17     hwc = (typeof hwc === "undefined" || !hwc) ? {} : hwc;    //
SUP 'namespace'
```

```
18
19     /**
20     * Global Legacy Mapping
21     * Needed because called by generated HTML or hardcoded in
22     * workflow.js or XBWUtil.java or in container callbacks.
23
24     /**
25     * @deprecated Deprecated since version 2.2 - use
26     hwc.guid()
27     */
28     function guid()           { return hwc.guid(); }
29
30     /**
31     * @deprecated Deprecated since version 2.2 - use
32     hwc.getXMLHTTPRequest()
33     */
34     function getXMLHTTPRequest() { return
35     hwc.getXMLHTTPRequest(); }
36
37     /**
38     * @deprecated Deprecated since version 2.2 - use
39     hwc.log(sMsg, eLevel, notifyUser)
40     */
41     function logToWorkflow(sMsg, eLevel, notifyUser) { return
42     hwc.log(sMsg, eLevel, notifyUser); }
43
44     /**
45     * @deprecated Deprecated since version 2.2 - use
46     hwc.close()
47     */
48     function closeWorkflow()           { return hwc.close(); }
49
50     /**
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
45      * @deprecated Deprecated since version 2.2 - use
hwc.clearCacheItem( cachekey )
46      */
47      function clearCacheItem( cachekey ) { return
hwc.clearCacheItem( cachekey ); }
48
49      /**
50      * @deprecated Deprecated since version 2.2 - use
hwc.clearCache()
51      */
52      function clearCache()                { return
hwc.clearCache(); }
53
54      /**
55      * @deprecated Deprecated since version 2.2 - use
hwc.close()
56      */
57      function expireCredentials()         { return
hwc.expireCredentials(); }
58
59      /**
60      * @deprecated Deprecated since version 2.2 - use
hwc.showCertificatePicker()
61      */
62      function showCertificatePicker()     { return
hwc.showCertificatePicker(); }
63
64      /**
65      * @deprecated Deprecated since version 2.2 - use
hwc.saveLoginCertificate( certificate )
66      */
67      function saveLoginCertificate( certificate ) { return
hwc.saveLoginCertificate( certificate ); }
68
69      /**
70      * @deprecated Deprecated since version 2.2 - use
hwc.saveLoginCredentials( userName, password )
```

```

71     */
72     function saveLoginCredentials(userName, password) { return
hwc.saveLoginCredentials(userName, password); }
73
74     /**
75     * @deprecated Deprecated since version 2.2 - use
hwc.activationRequired()
76     */
77     function activationRequired()          { return
hwc.activationRequired(); }
78
79     /**
80     * @deprecated Deprecated since version 2.2 - use
hwc.showUrlInBrowser(url)
81     */
82     function showUrlInBrowser(url)        { return
hwc.showUrlInBrowser(url); }
83
84     /**
85     * @deprecated Deprecated since version 2.2 - use
hwc.markAsProcessed()
86     */
87     function markAsProcessed()           { return
hwc.markAsProcessed(); }
88
89     /**
90     * @deprecated Deprecated since version 2.2 - use
hwc.markAsActivated()
91     */
92     function markAsActivated()           { return
hwc.markAsActivated(); }
93
94     /**
95     * Delegate for data message processing details.
96     * In the custom case, the user is expected to provide their
own implemenation.

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
97      * In the default SUP HybridApp case, this updates values then
sets the next screen to navigate to.
98      * @param {string} incomingDataMessageValue The XML formatted
string for the incoming message
99      * @param {boolean} noUI true if this has no UI
100     * @param {boolean} loading If true, this is being called
while the application is loading
101     * @param {boolean} fromActivationFlow If true, this is being
called from within an activation flow
102     * @param {string} dataType If supplied, the data type of the
value display on target screen
103     */
104     function processDataMessage(incomingDataMessageValue, noUI,
loading, fromActivationFlow, dataType) {
105         if( typeof(hwc.processDataMessage) === 'function' ) {
106             return
hwc.processDataMessage(incomingDataMessageValue, noUI, loading,
fromActivationFlow, dataType);
107         }
108         else {
109             // get the users attention
110             hwc.log("Implementation required for
hwc.processDataMessage", "ERROR", true);
111             throw new Error("Implementation required for either
global processDataMessage or hwc.processDataMessage");
112         }
113     }
114
115     /**
116     * @deprecated Deprecated since version 2.2 - use
hwc.processDataMessage(incomingDataMessageValue, noUI, loading,
fromActivationFlow, dataType)
117     */
118     function processWorkflowMessage(incomingDataMessageValue,
noUI, loading, fromActivationFlow, dataType) {
119         return processDataMessage(incomingDataMessageValue, noUI,
loading, fromActivationFlow, dataType);
120     }
```

```
121
122     /**
123     * This function is invoked by the container when there is a
124     * native error to report.
125     * Use {@link hwc.setReportErrorFromNative} to set the
126     * callback function this function will call.
127     * This function is not intended to be called except by the
128     * container.
129     * @private
130     * @param {string} errString The string contains error
131     * message
132     */
133     function reportErrorFromNative(errString) {
134         var reportErrorCallback =
135         hwc.getReportErrorFromNativeCallback();
136         if( typeof reportErrorCallback === "function" )
137         {
138             reportErrorCallback( errString );
139         }
140     }
141
142     /**
143     * Container API
144     */
145     (function(hwc, window, undefined) {
146
147         /**
148         * A number representing the logging level. The logging
149         * level must be an integer from the range [1..4]
150         * with the higher numbers being more verbose.
151         *
152         * @type {number}
153         */
154         var requestedLoggingLevel,
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
149      /**
150         * A callback function used when {@link hwc.log} is
invoked with true for the notifyUser parameter.
151         * This callback should notify the user of the log message
in an appropriate manner.
152         *
153         * @type {anonymous.alertDialogCallbackFunction}
154         */
155         requestedAlertDialogCallback,
156     /**
157         * A callback function used when there is a native error to
report
158         *
159         * @type {anonymous.errorCallbackFunction}
160         */
161         reportErrorFromNativeCallback;
162
163     /**
164         * This object contains constants representing the status
of the hybrid app.
165         */
166     hwc.STATUS = {};
167
168     /**
169         * A constant indicating the hybrid app has been
closed.
170         * @constant
171         * @type {number}
172         * @member of hwc.STATUS
173         * @public
174         */
175     hwc.STATUS.CLOSED = 1;
176
177     /**
```



```
178     * A constant indicating the hybrid app is running.
179     * @constant
180     * @type {number}
181     * @member of hwc.STATUS
182     * @public
183     */
184     hwc.STATUS.RUNNING = 2;
185
186     /**
187     * A status representing the hybrid app, default is
    running.
188     *
189     * @type {number}
190     */
191     var status = hwc.STATUS.RUNNING;
192
193     /**
194     * This function sets the callback used by hwc.log when it
    is required to notify the user of a log item.
195     *
196     * @param {anonymous.alertDialogCallbackFunction}
    newAlertDialogCallback The alert dialog to use.
197     *
198     * @example
199     * customLogAlert = function( message )
200     * {
201     *     alert( "New log message: " + message );
202     * }
203     * hwc.setLoggingAlertDialog( customLogAlert );
204     * @memberOf hwc
205     * @public
206     */
207     hwc.setLoggingAlertDialog =
    function( newAlertDialogCallback )
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
208     {
209         requestedAlertDialogCallback =
newAlertDialogCallback;
210     };
211
212     /**
213     * This function gets the callback used by hwc.log when it
is required to notify the user of a log item.
214     *
215     * @memberOf hwc
216     * @public
217     * @returns {anonymous.alertDialogCallbackFunction} The
alert dialog callback function.
218     */
219     hwc.getLoggingAlertDialog = function()
220     {
221         return requestedAlertDialogCallback;
222     };
223
224     /**
225     * This function sets the logging level. The logging level
set with this function only persists as long as this javascript
context does.
226     * When the hybrid app is closed, the value set with this
function is lost.
227     * @memberOf hwc
228     * @public
229     * @param {number} newLoggingLevel The number representing
the new logging level.
230     * Must be an integer in the range [1..4]. The higher
numbers represent more verbose logging levels
231     * from 1 for ERROR level logging up to 4 for DEBUG level
logging.
232     * @example
233     * // Set the logging level to debug.
234     * hwc.setLoggingCurrentLevel( 4 );
```

```
235     */
236     hwc.setLoggingCurrentLevel =
function( newLoggingLevel )
237     {
238         requestedLoggingLevel = newLoggingLevel;
239     };
240
241     /**
242     * This function gets the logging level.
243     *
244     * @memberOf hwc
245     * @public
246     * @returns {number} A number representing the logging
level. Will be an integer in the range [1..4].
247     * The higher numbers represent more verbose logging
levels from 1 for ERROR level logging up to 4 for DEBUG level
logging.
248     * @example
249     * // Get the logging level
250     * var loggingLevel = hwc.getLoggingCurrentLevel();
251     */
252     hwc.getLoggingCurrentLevel = function() {
253         var logLevel;
254         if (requestedLoggingLevel === undefined) {
255             logLevel = hwc.getQueryVariable("loglevel");
256             requestedLoggingLevel = logLevel ?
parseInt(logLevel, 10) : 1;
257         }
258         return requestedLoggingLevel;
259     };
260
261     /**
262     * This function sets the callback function called when
there is a native error reported.
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
263      * Calling this function will replace any callback that
had been set previously.
264      * @memberOf hwc
265      * @public
266      * @param {function} callbackToSet The callback
function.
267      * @example
268      * var errorCallback = function( errorString )
269      * {
270      *     alert( "There was a native error: " +
errorString );
271      * }
272      *
hwc.setReportErrorFromNativeCallback( errorCallback );
273      */
274      hwc.setReportErrorFromNativeCallback =
function( callbackToSet )
275      {
276          reportErrorFromNativeCallback = callbackToSet;
277      };
278
279      /**
280      * This function returns the callback function that will
be called by {@link reportErrorFromNative}.
281      * This function is not intended to be called by any
function but {@link reportErrorFromNative}.
282      * @private
283      * @returns {function} The callback function.
284      */
285      hwc.getReportErrorFromNativeCallback = function()
286      {
287          return reportErrorFromNativeCallback;
288      };
289
290      /**
```

```
291      * This function looks in the query string on the URL for
the value corresponding to the given name.
292      *
293      * @memberOf hwc
294      * @public
295      * @param {string} variable The name of the variable in the
URL to retrieve the value for.
296      * @returns {string} The value corresponding to the given
name.
297      * @example
298      * // Get the pageToShow variable from the URL query
string
299      * var pageToShow =
hwc.getQueryVariable( "pageToShow" );
300      */
301      hwc.getQueryVariable = function(variable) {
302          var query, vars, i, pair;
303          query = window.location.search.substring(1);
304          vars = query.split("&");
305          for (i = 0; i < vars.length; i++) {
306              pair = vars[i].split("=");
307              if (pair[0] === variable) {
308                  return unescape(pair[1]);
309              }
310          }
311      };
312
313
314      /**
315      * This object contains constants representing the different
types of public native error codes.
316      * Error codes larger than 500 are reserved for server
communication errors which may occur as the result of online requests
and/or attachment downloads
317      *
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
318     * @namespace
319     */
320     hwc.NativeErrorCodes = {};
321     /**
322     * A constant indicating there was an unknown error.
323     *
324     * @constant
325     *
326     * @type {number}
327     * @memberOf hwc.NativeErrorCodes
328     * @public
329     */
330     hwc.NativeErrorCodes.UNKNOWN_ERROR = 1;
331     /**
332     * A constant indicating the attachment has not been
333     * downloaded.
334     *
335     * @type {number}
336     * @memberOf hwc.NativeErrorCodes
337     * @public
338     */
339     hwc.NativeErrorCodes.ATTACHMENT_NOT_DOWNLOADED = 100;
340     * A constant indicating there was an unknown MIME type.
341     *
342     * @type number
343     * @memberOf hwc.NativeErrorCodes
344     * @public
345     */
346     hwc.NativeErrorCodes.UNKNOWN_MIME_TYPE = 101;
347     /**
348     * A constant indicating there was a filename without an
349     * extension.
```

```
349     *
350     * @type {number}
351     * @memberOf hwc.NativeErrorCodes
352     * @public
353     */
354     hwc.NativeErrorCodes.FILENAME_NO_EXTENSION = 102;
355     /**
356     * A constant indicating a required parameter was not
357     * available.
358     * @type {number}
359     * @memberOf hwc.NativeErrorCodes
360     * @public
361     */
362     hwc.NativeErrorCodes.REQUIRED_PARAMETER_NOT_AVAILABLE =
363     103;
364     /**
365     * A constant indicating there was no certificate selected by
366     * the user.
367     * @type {number}
368     * @memberOf hwc.NativeErrorCodes
369     * @public
370     */
371     hwc.NativeErrorCodes.CERTIFICATE_NOT_SELECTED = 104;
372     /**
373     * A constant indicating the attachment type is not
374     * supported.
375     * @type {number}
376     * @memberOf hwc.NativeErrorCodes
377     * @public
378     */
379     hwc.NativeErrorCodes.UNSUPPORTED_ATTACHMENT_TYPE = 105;
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
379     /**
380     * A constant indicating there was an SSO certificate manager
exception.
381     *
382     * @type {number}
383     * @memberOf hwc.NativeErrorCodes
384     * @public
385     */
386     hwc.NativeErrorCodes.SSOCERT_EXCEPTION = 106;
387     /**
388     * A constant indicating a failure to save a credential.
389     *
390     * @type {number}
391     * @memberOf hwc.NativeErrorCodes
392     * @public
393     */
394     hwc.NativeErrorCodes.FAIL_TO_SAVE_CREDENTIAL = 107;
395     /**
396     * A constant indicating a failure to save a certificate.
397     *
398     * @type {number}
399     * @memberOf hwc.NativeErrorCodes
400     * @public
401     */
402     hwc.NativeErrorCodes.FAIL_TO_SAVE_CERTIFICATE = 108;
403     /**
404     * A constant indicating the device is not connected.
405     *
406     * @type {number}
407     * @memberOf hwc.NativeErrorCodes
408     * @public
409     */
```



```
410     hwc.NativeErrorCodes.DEVICE_NOT_CONNECTED = 109;
411     /**
412     * A constant indicating the response is too large for a
413     * javascript variable.
414     * @type {number}
415     * @memberOf hwc.NativeErrorCodes
416     * @public
417     */
418     hwc.NativeErrorCodes.RESPONSE_TOO_LARGE = 110;
419     /**
420     * A constant indicating that opening the URL failed.
421     *
422     * @type {number}
423     * @memberOf hwc.NativeErrorCodes
424     * @public
425     */
426     hwc.NativeErrorCodes.NAVIGATION_ERROR = 111;
427     /**
428     * A constant indicating an invalid common name was passed
429     * while requesting a certificate from Afaria.
430     * @type {number}
431     * @memberOf hwc.NativeErrorCodes
432     * @public
433     */
434     hwc.NativeErrorCodes.INVALID_COMMON_NAME = 112;
435
436
437
438     /**
439     * A utility function for use in generating a GUID
440     *
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
441     * @private
442     * @returns {string} The generated GUID.
443     */
444     function S4() {
445         return ((1+Math.random()*0x10000|
446         0).toString(16).substring(1);
447
448     /**
449     * This function generates a GUID (globally unique
450     identifier).
451     *
452     * @memberOf hwc
453     * @public
454     * @returns {string} The generated GUID.
455     * @example
456     * var globallyUniqueName = hwc.guid();
457
458     hwc.guid = function() {
459         return (S4()+S4()+"-"+S4()+"-"+S4()+"-"+S4()+"-"+S4()+
460         +S4()+S4());
461     };
462
463     /**
464     * Reliably returns an XMLHttpRequest object regardless of
465     what platform this code is being executed on.
466     *
467     * @memberOf hwc
468     * @public
469     * @returns {object} An XMLHttpRequest object.
470     * @example
471     * var request = hwc.getXMLHttpRequest();
472
473     hwc.getXMLHttpRequest = function getXMLHttpRequest() {
```

```
471      //
472      /*
473      SMPONP-9439
474      Avoid this endless loop:
475      hwc.log()
476      at hwc.traceEnteringMethod()
477      at hwc.getXMLHttpRequest()
478      at hwc.postDataToContainer()
479      at hwc.log()
480      */
481      //hwc.traceEnteringMethod("hwc.getXMLHttpRequest");
482      try {
483          var xmlHttpRequest;
484          if (window.XMLHttpRequest) {
485              xmlHttpRequest = new XMLHttpRequest();
486          }
487          else { // code for IE6, IE5
488              xmlHttpRequest = new
ActiveXObject("Microsoft.XMLHTTP");
489          }
490          return xmlHttpRequest;
491      } finally {
492          //hwc.traceLeavingMethod("hwc.getXMLHttpRequest");
493      }
494      };
495
496
497      /***** Hybrid App NATIVE FUNCTIONS
498      *****/
499      /**
500      * Sets the title of the screen.
501      *
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
502     * @memberOf hwc
503     * @public
504     * @param {string} screenTitle The screen title to use.
505     * @example
506     * hwc.setScreenTitle_CONT( "Custom Screen Title" );
507     */
508     hwc.setScreenTitle_CONT = function(screenTitle) {
509         hwc.traceEnteringMethod("hwc.setScreenTitle_CONT");
510         try {
511             if (hwc.isWindows()) {
512                 document.title = screenTitle;
513             }
514             else {
515                 if (hwc.isIOS() || hwc.isAndroid()) {
516                     hwc.getDataFromContainer("setscreentitle",
517 "&title=" + encodeURIComponent(screenTitle));
518                 }
519                 else {
520                     hwc.postDataToContainer("setscreentitle",
521 "&title=" + encodeURIComponent(screenTitle));
522                 }
523             } finally {
524                 hwc.traceLeavingMethod("hwc.setScreenTitle_CONT");
525             }
526         };
527     /**
528     * This class represents a collection of menu items.
529     * @memberOf hwc
530     * @public
531     * @classdesc
532     * @example
```

```
533     * // This is the function we'll use as a callback for the
534     * first menu item.
535     * {
536     *     alert( "You clicked the first menu item!" );
537     * }
538     *
539     * // This is the function we'll use as a callback for the
540     * second menu item.
541     * var callback2 = function()
542     * {
543     *     alert( "You clicked the second menu item!" );
544     * }
545     * // This function creates and adds a menu item
546     * collection.
547     * var addMenuItems = function()
548     * {
549     *     var menuItemCollection = new
550     * hwc.MenuItemCollection();
551     *     menuItemCollection.addItem("menu item 1",
552     * "callback()");
553     *     menuItemCollection.addItem("menu item 2",
554     * "callback2()");
555     *     hwc.addItemCollection( menuItemCollection );
556     * }
557     */
558     hwc.MenuItemCollection = function() {
559     this.menuItems = [];
560     this.subMenuName = null;
561     this.okAction = null;
562     };
563     /**
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
561      * This function adds a menu item to the collection.
562      *
563      * @memberOf hwc.MenuItemCollection
564      * @public
565      * @param {string} title The display text for the menu
item.
566      * @param {anonymous.genericCallbackFunction} callback The
function to call when the menu item is clicked.
567      * @param {boolean} [isDefault] Determines if the menu item is
selected by default on BlackBerry.
568      * If more than one menu item is added to the same collection
with true for this parameter, the
569      * last menu item added with true for this parameter will be
selected by default on Blackberry.
570      * @example
571      * var callbackFunctionName = function()
572      * {
573      *     alert( "Menu item clicked!" );
574      * }
575      * var menuItemCollection = new hwc.MenuItemCollection();
576      * menuItemCollection.addMenuItem("menu item name",
"callbackFunctionName()", true);
577      *
578      */
579      hwc.MenuItemCollection.prototype.addMenuItem =
function(title, callback, isDefault) {
580      hwc.traceEnteringMethod("hwc.MenuItemCollection.addMenuItem");
581      try {
582          this.menuItems.push( { "name" : title, "action" :
callback, "default" : isDefault ? "true" : "false" } );
583      } finally {
584      hwc.traceLeavingMethod("hwc.MenuItemCollection.addMenuItem");
585      }
586      };
```

```
587
588     /**
589     * This function sets the sub menu name to use on Windows
590     * Mobile.
591     *
592     * @memberOf hwc.MenuItemCollection
593     * @public
594     * @param {string} name The sub menu name to use.
595     * @example
596     * var callbackFunctionName = function()
597     * {
598     *     alert( "Menu item clicked!" );
599     * }
600     * var menuItemCollection = new hwc.MenuItemCollection();
601     * menuItemCollection.setSubMenuName( "Custom Menu" );
602     * menuItemCollection.addMenuItem("menu item name",
603     * "callbackFunctionName()");
604     */
605     hwc.MenuItemCollection.prototype.setSubMenuName =
606     function(name) {
607         this.subMenuName = name;
608     };
609
610     /**
611     * This function sets the OK action to use on WM.
612     *
613     * @memberOf hwc.MenuItemCollection
614     * @public
615     * @param {anonymous.genericCallbackFunction} callback The
616     * function to call when the OK button is pressed.
617     *
618     * @example
619     * var callbackFunctionName = function()
620     * {
621     *     alert( "Menu item clicked!" );
622     * }
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
617     * }
618     * var okActionFunction = function()
619     * {
620     *     alert( "A OKAY!" );
621     * }
622     * var menuItemCollection = new hwc.MenuItemCollection();
623     * menuItemCollection.setOKAction( "okActionFunction()" );
624     * menuItemCollection.addItem("menu item name",
625     * "callbackFunctionName()");
626     */
627     hwc.MenuItemCollection.prototype.setOKAction =
628     function(callback) {
629         this.okAction = callback;
630     };
631     /**
632     * This function converts the menu item collection to a JSON
633     * string. This function
634     * is used as a helper for {@link
635     * hwc.addItemCollection}.
636     *
637     * @memberOf hwc.MenuItemCollection
638     * @public
639     * @returns {string} The JSON string representing this menu
640     * item collection.
641     * @example
642     * var callbackFunctionName = function()
643     * {
644     *     alert( "Menu item clicked!" );
645     * }
646     * var menuItemCollection = new hwc.MenuItemCollection();
647     * var jsonMenuItemCollection =
648     * menuItemCollection.stringify();
649     */
```



```
645     hwc.MenuItemCollection.prototype.stringify = function()
646     {
647         return JSON.stringify({
648             "menuitems" : this.menuItems,
649             "submenuname" : this.subMenuName,
650             "OK" : this.okAction
651         });
652     };
653
654     /**
655     * This function adds a menu item collection to the menu items
656     * for the screen.
657     *
658     * @memberOf hwc
659     * @public
660     * @param {hwc.MenuItemCollection} collection The collection
661     * of menu items to add to the screen.
662     * @example
663     * var callbackFunctionName = function()
664     * {
665     *     alert( "Menu item clicked!" );
666     * }
667     * var menuItemCollection = new hwc.MenuItemCollection();
668     * menuItemCollection.addMenuItem("menu item name",
669     * "callbackFunctionName()");
670     * hwc.addMenuItemCollection( menuItemCollection );
671     */
672     hwc.addMenuItemCollection = function(collection) {
673         hwc.traceEnteringMethod("hwc.addMenuItemCollection");
674         try {
675             if (isBlackBerry() || isWindowsMobile() ||
676                 isAndroid()) {
677                 var request = "menuitems=" +
678                     encodeURIComponent(collection.stringify());
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
674             hwc.postDataToContainer("addallmenuitems",
request);
675         }
676     } finally {
677
678         hwc.traceLeavingMethod("hwc.addItemCollection");
679     }
680
681     /**
682     * Allows the user to add a menuitem with the specified name
and with the specified
683     * callback, which will be invoked when the menuitem is
clicked. This function should
684     * only be used in hybrid apps generated with the Unwired
Workspace designer.
685     *
686     * @memberOf hwc
687     * @private
688     *
689     * @param {string} menuItemName The specified menuitem
name.
690     * @param {string} functionName The string representing the
call to the {@link anonymous.genericCallbackFunction} callback
function.
691     * @param {string} subMenuName The specific sub-menu name for
Windows Mobile.
692     * @param {string} screenToShow The screen about to be
shown.
693     * @param {string} [menuItemKey] The menuItem's key.
694     * @example
695     * var callbackFunction = function()
696     * {
697     *     alert( "Menu Item Clicked!" );
698     * }
699     * hwc.addItem_COLLECTION( "Custom Menu Item",
"callbackFunction()", "Custom Sub Menu", "Start" );
```

```
700     */
701     hwc.addItem_CONT = function(menuItemName, functionName,
subMenuName, screenToShow, menuItemKey) {
702         var div, menuStr, idxOfMenuItemName, comma, request;
703         hwc.traceEnteringMethod("hwc.addItem_CONT");
704         try {
705             //first add the item to sup_menuitems
706             div = document.getElementById(screenToShow +
"ScreenDiv");
707             menuStr = div.getAttribute("sup_menuitems");
708             idxOfMenuItemName = menuStr.indexOf(menuItemName);
709             if (idxOfMenuItemName !== -1) {
710                 return;
711             }
712             comma = (menuStr.length > 0) ? "," : "";
713             menuStr = menuStr + comma + menuItemName + "," +
menuItemKey;
714             try {
715                 div.setAttribute("sup_menuitems", menuStr); //has
no affect on Windows Mobile
716             }
717             catch (e) {
718             }
719
720             request = "menuItemname=" +
encodeURIComponent(menuItemName);
721             request += ("&onmenuclick=" +
encodeURIComponent(functionName) + "()");
722             if (hwc.isWindowsMobile()) {
723                 request += "&submenuname=";
724                 if (subMenuName) {
725                     request += encodeURIComponent(subMenuName);
726                 }
727             } else {
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
728         if (resources) {
729             request +=
encodeURIComponent(resources.getString("MENU"));
730         }
731         else {
732             request += "Menu";
733         }
734     }
735     hwc.postDataToContainer("addMenuItem",
request);
736     }
737     else if (hwc.isAndroid() || hwc.isBlackBerry()) {
738         hwc.postDataToContainer("addMenuItem",
request);
739     }
740     } finally {
741         hwc.traceLeavingMethod("hwc.addMenuItem_CONT");
742     }
743 };
744
745
746 /**
747  * This function removes all menu items that were added by the
hybrid app.
748  * Note: This API does not support on iOS platform.
749  * @memberOf hwc
750  * @public
751  * @example
752  * hwc.removeAllMenuItems();
753  */
754 hwc.removeAllMenuItems = function() {
755     hwc.traceEnteringMethod("hwc.removeAllMenuItems");
756     try {
```

```
757         if (hwc.isAndroid() || hwc.isWindowsMobile() ||
hwc.isBlackBerry() ) {
758             hwc.getDataFromContainer("removeallmenuitems");
759         }
760     } finally {
761         hwc.traceLeavingMethod("hwc.removeAllMenuItems");
762     }
763 };
764
765 /**
766  * This function sets the activation required state of this
hybrid app to true. After calling this
767  * function, the current hybrid app will need to be
activated.
768  * @memberOf hwc
769  * @public
770  * @example
771  * hwc.activationRequired();
772  */
773 hwc.activationRequired = function() {
774     hwc.traceEnteringMethod("hwc.activationRequired");
775     try {
776 hwc.getDataFromContainer("requiresactivation");
777     } finally {
778         hwc.traceLeavingMethod("hwc.activationRequired");
779     }
780 };
781
782 /**
783  * This function sets the activation required state for the
current hybrid app to false. After calling this
784  * function, the current hybrid app will not need to be
activated.
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
785     * @memberOf hwc
786     * @public
787     * @example
788     * hwc.markAsActivated();
789     */
790     hwc.markAsActivated = function() {
791         hwc.traceEnteringMethod("hwc.markAsActivated");
792         try {
793             hwc.getDataFromContainer("markasactivated");
794         } finally {
795             hwc.traceLeavingMethod("hwc.markAsActivated");
796         }
797     };
798
799     /**
800     * Allows the user to set the processed state to true for the
801     * current message.
802     * @memberOf hwc
803     * @public
804     * @example
805     * hwc.markAsProcessed()
806     */
807     hwc.markAsProcessed = function() {
808         hwc.traceEnteringMethod("hwc.markAsProcessed");
809         try {
810             hwc.getDataFromContainer("markasprocessed");
811         } finally {
812             hwc.traceLeavingMethod("hwc.markAsProcessed");
813         }
814     };
815     /**
```

```
816     * Allows the user to set the credentials to the expired state
for the current hybrid app.
817     * @memberOf hwc
818     * @public
819     * @example
820     * hwc.expireCredentials();
821     */
822     hwc.expireCredentials = function() {
823         hwc.traceEnteringMethod("hwc.expireCredentials");
824         try {
825             hwc.getDataFromContainer("expirecredentials");
826         } finally {
827             hwc.traceLeavingMethod("hwc.expireCredentials");
828         }
829     };
830
831     /**
832     * This function clears the contents of the on-device request
result cache for the current hybrid app.
833     * @memberOf hwc
834     * @public
835     * @example
836     * hwc.clearCache();
837     */
838     hwc.clearCache = function() {
839         hwc.traceEnteringMethod("hwc.clearCache");
840         try {
841             hwc.getDataFromContainer("clearrequestcache");
842         } finally {
843             hwc.traceLeavingMethod("hwc.clearCache");
844         }
845
846     };
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
847
848     /**
849     * This function clears an item from the contents of the on-
device request result cache for the current hybrid app.
850     *
851     * @memberOf hwc
852     * @public
853     * @param {string} cachekey The key for the cache item to be
removed. This is the same key that was passed to
hwc.doOnlineRequest.
854     * @example
855     * // The cache key is set when calling
hwc.doOnlineRequest_CONT
856     * hwc.doOnlineRequest( .. .. .. .. .. .,
cacheKey, .. );
857     * // At some later point if we want to clear the cache for the
above request, we use the following code:
858     * hwc.clearCacheItem( cacheKey );
859     */
860     hwc.clearCacheItem = function( cachekey ) {
861         var request;
862         hwc.traceEnteringMethod("hwc.clearCacheItem");
863         try {
864             request = "cachekey=" +
encodeURIComponent(cachekey);
865
866             hwc.postDataToContainer("clearrequestcacheitem",
request);
867         } finally {
868             hwc.traceLeavingMethod("hwc.clearCacheItem");
869         }
870     };
871
872
873     /**
```



```

874      * Allows the user to log a message to the device trace log
      which can be remotely retrieved from the server.

875      * Whether the message actually gets logged will depend on how
      the log level that the administrator has selected

876      * for this device user compares with the log level of this
      message.

877      * The logging level and alert dialog callback can be set with
      {@link hwc.setLoggingCurrentLevel} and {@link
      setLoggingAlertDialog}.

878      *

879      * @memberOf hwc

880      * @public

881      * @param {string} sMsg The message to be logged.

882      * @param {string} eLevel The error level for this message.
      This parameter must be one of: "ERROR", "WARN", "INFO" or "DEBUG".

883      * @param {boolean} notifyUser Whether the logging alert
      callback will be invoked. This parameter is independent of the

884      * logging level (the logging alert callback will always be
      invoked if this is true, and never if this is false).

885      * @example

886      * var logAlert = function( message )
887      * {
888      *     alert( "New log message: " + message );
889      * }

890      * hwc.setLoggingAlertDialog( logAlert );

891      * hwc.setLoggingCurrentLevel( 3 );

892      * // The following will be logged, and the logging alert
      dialog will be invoked.

893      * hwc.log( "info message notify", "INFO", true );

894      * // The following will be logged, but the logging alert
      dialog will not be invoked.

895      * hwc.log( "info message", "INFO", false );

896      * // The following will not be logged, but the logging alert
      dialog will be invoked.

897      * hwc.log( "debug message notify", "DEBUG", true );

898      * // The following will not be logged, and the logging alert
      dialog will not be invoked.

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
899     * hwc.log( "debug message", "DEBUG", false );
900     *
901     */
902     hwc.log = function log(sMsg, eLevel, notifyUser) {
903         var msgLogLevel;
904         if( !sMsg ) {
905             return;
906         }
907         if (notifyUser && hwc.getLoggingAlertDialog()) {
908             (hwc.getLoggingAlertDialog())(sMsg);
909         }
910
911         switch (eLevel) {
912             case "ERROR":
913                 msgLogLevel = 1;
914                 break;
915             case "WARN":
916                 msgLogLevel = 2;
917                 break;
918             case "INFO":
919                 msgLogLevel = 3;
920                 break;
921             case "DEBUG":
922                 msgLogLevel = 4;
923                 break;
924             default:
925                 msgLogLevel = 1;
926         }
927         if((sMsg === "") || (msgLogLevel >
hwc.getLoggingCurrentLevel()) || (hwc.isWindows())) {
928             return;
929         }
```

```
930
931     if (hwc.isAndroid()) {
932         _HWC.log(sMsg, msgLogLevel);
933     } else {
934         hwc.postMessageToContainer("logtoworkflow", "loglevel=" +
msgLogLevel + "&logmessage=" + encodeURIComponent(sMsg));
935     }
936
937 };
938
939 /**
940  * This function opens a form on the device that allows the
user to specify the credentials for the use of
941  * certificate-based authentication. If the user picks a
certificate, then that certificate is saved in the
942  * credentials cache.
943  * @memberOf hwc
944  * @public
945  * @example
946  * hwc.showCertificatePicker();
947  */
948 hwc.showCertificatePicker = function() {
949     hwc.traceEnteringMethod("hwc.showCertificatePicker");
950     try {
951         hwc.getDataFromContainer("showcertpicker");
952     } finally {
953         hwc.traceLeavingMethod("hwc.showCertificatePicker");
954     }
955 };
956
957
958 /**
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
959     * This function saves login credentials from a certificate to
the credential cache.
960     * The common name is used for the username and the signed
certificate is used for the password.
961     *
962     * @memberOf hwc
963     * @public
964     * @param {object} certificate The values
certificate.subjectCN and certificate.signedCertificate must be
defined.
965     * @example
966     * var certInfo = {};
967     * certInfo.subjectCN = "sampleCommonName";
968     * certInfo.signedCertificate = "samplePassword";
969     * hwc.saveLoginCertificate( certInfo );
970     */
971     hwc.saveLoginCertificate = function(certificate) {
972         hwc.traceEnteringMethod("hwc.saveLoginCertificate");
973         try {
974             hwc.saveLoginCredentials(certificate.subjectCN,
certificate.signedCertificate, true);
975         } finally {
976             hwc.traceLeavingMethod("hwc.saveLoginCertificate");
977         }
978     };
979
980     /**
981     * This function saves login credentials to the credential
cache.
982     *
983     * @memberOf hwc
984     * @public
985     * @param {string} userName The user name to save
986     * @param {string} password The password to save
```

```

987      * @example
988      * hwc.saveLoginCredentials( "sampleUserName",
"samplePassword" );
989      */
990      hwc.saveLoginCredentials = function(userName, password) {
991          var requestData;
992          hwc.traceEnteringMethod("hwc.saveLoginCredentials");
993          try {
994              requestData = "supusername=" +
encodeURIComponent(userName) + "&suppassword=" +
encodeURIComponent(password);
995
996              if (hwc.isAndroid()) {
997                  _HWC.saveCredentials( userName, password );
998              } else {
999                  hwc.postDataToContainer("savecredential",
requestData);
1000              }
1001
1002          } finally {
1003              hwc.traceLeavingMethod("hwc.saveLoginCredentials");
1004          }
1005      };
1006
1007      /**
1008      * This function opens the supplied URL in a browser. The
browser opens on top of the hybrid app - the
1009      * context of the hybrid app is undisturbed.
1010      *
1011      * @memberOf hwc
1012      * @public
1013      * @param {string} url The URL to be shown in a browser.
1014      * @example
1015      * hwc.showUrlInBrowser( "http://www.google.com" );

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1016     */
1017     hwc.showUrlInBrowser = function showUrlInBrowser(url)
1018     {
1019         var idxOfColon;
1020         hwc.traceEnteringMethod("hwc.showUrlInBrowser");
1021         try {
1022             url = hwc.trimSpaces(url, true);
1023             idxOfColon = url.indexOf(":");
1024             if (idxOfColon === -1 || (idxOfColon > 7)) {
1025                 url = "http://" + url;
1026             }
1027
1028             if (hwc.isWindowsMobile() || hwc.isAndroid() ||
hwc.isIOS() || hwc.isBlackBerry()) {
1029                 hwc.getDataFromContainer("showInBrowser", "&url="
+ encodeURIComponent(url));
1030             } else {
1031                 window.open(url);
1032             }
1033         } finally {
1034             hwc.traceLeavingMethod("hwc.showUrlInBrowser");
1035         }
1036     };
1037
1038     /**
1039     * Shows the given file contents in a content-appropriate way.
The type of the content is
1040     * supplied by either the MIME type or the filename, at least
one of which must be supplied.
1041     * The content itself should be presented as a base64-encoded
string. Not all file types may
1042     * be supported on all platforms.
1043     *
1044     * @memberOf hwc
```

```

1045     * @public
1046     * @param {string} contents The base-64 encoded version of the
binary content of the attachment to be displayed.
1047     * @param {string} mimeType The MIME type of the file.
1048     * @param {string} fileName The name of the file.
1049     * @param {anonymous.genericCallbackFunction}
waitDialogCallbackString The callback function used to close a wait
dialog once the attachment
1050     * is done opening.
1051     * @example
1052     * var openAttachmentBase64StringPng = function()
1053     * {
1054     *     // How you want get the base 64 encoding of the file is
up to you. This string represents a small png image.
1055     *     var data =
"iVBORw0KGgoAAAANSUUhEUgAAACAAAAAgCAYAAABzenr0AAAAAXNSR0IArs4c6QAAAA
RnQU1BAACxjwv8YQUAAAJcEhZcwAADsMAAA7DACdvdqGQAAAA0SURBVFH7dAxEQAAC
AMx3CAT6eVQwZKh8/dSmc7n6jN
+bQcIECBAGAABAGQIECBAGACBBb3SkJeQ67u1AAAAElFTkSuQmCC";
1056     *     hwc.showProgressDialog();
1057     *     // Don't have to pass the filename because we are
passing the MIME type.
1058     *     hwc.showAttachmentContents_CONT( data, "image/png",
null, "hwc.hideProgressDialog()" );
1059     * }
1060     *
1061     * @example
1062     * var openAttachmentBase64StringTxt = function()
1063     * {
1064     *     // How you want get the base 64 encoding of the file is
up to you. This string represents a short text file.
1065     *     var data =
"VGhpcyBpcyBwYXJ0IG9mIGEgaHlicmlkIGFwcC4=";
1066     *     // Don't have to pass the MIME type because we are
passing the filename.
1067     *     hwc.showAttachmentContents_CONT( data, null,
"attach.txt" );
1068     * }

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1069     */
1070     hwc.showAttachmentContents_CONT = function(contents,
mimeType, fileName, waitDialogCallbackString) {
1071         var request;
1072
1073         hwc.traceEnteringMethod("hwc.showAttachmentContents_CONT");
1074         try {
1075             request = "callback=" + waitDialogCallbackString;
1076
1077             if (hwc.isWindowsMobile()) {
1078                 contents = contents.replace(/=/g, "~");
1079                 request += "&Attachmentdata=" + contents;
1080             } else {
1081                 request += "&Attachmentdata=" +
encodeURIComponent(contents);
1082             }
1083             if (contentType) {
1084                 request += "&mimetype=" +
encodeURIComponent(mimeType);
1085             }
1086             if (fileName) {
1087                 request += "&filename=" +
encodeURIComponent(fileName);
1088             }
1089             hwc.postDataToContainer("showattachment", request);
1090         } finally {
1091             hwc.traceLeavingMethod("hwc.showAttachmentContents_CONT");
1092         }
1093     };
1094
1095     /**
1096     * Shows the given file contents in a content-appropriate way.
The type of the content is
```



```
1097     * supplied by either the MIME type or the filename, at least
1098     * one of which must be supplied.
1099     * The content itself will be a unique key supplied earlier to
1100     * a call to doAttachmentDownload.
1101     * @memberOf hwc
1102     * @public
1103     * @param {string} uniqueKey The unique key for the
1104     * attachment.
1105     * @param {string} mimeType The MIME type of the file.
1106     * @param {string} fileName The name of the file.
1107     * @param {string} waitDialogCallbackString string with the
1108     * value for the 'callback=' parameter.
1109     */
1110     hwc.showAttachmentFromCache_CONT = function(uniqueKey,
1111     mimeType, fileName, waitDialogCallbackString) {
1112         var request;
1113         hwc.traceEnteringMethod("hwc.showAttachmentFromCache_CONT");
1114         try {
1115             request = "callback=" + waitDialogCallbackString;
1116             request += "&uniquekey=" +
1117             encodeURIComponent(uniqueKey);
1118             if (mimeType) {
1119                 request += "&mimetype=" +
1120                 encodeURIComponent(mimeType);
1121             }
1122             if (fileName) {
1123                 request += "&filename=" +
1124                 encodeURIComponent(fileName);
1125             }
1126             hwc.postDataToContainer("showattachment", request);
1127         } finally {
1128             hwc.traceLeavingMethod("hwc.showAttachmentFromCache_CONT");
1129         }
1130     };
1131 }
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1123     }
1124   };
1125
1126   /**
1127    * Shows a local attachment.
1128    *
1129    * @memberOf hwc
1130    * @public
1131    * @param {string} key The key of the attachment. This is the
1132    * path to the file, with the root being the
1133    * folder that manifest.xml is located.
1134    * @example
1135    * hwc.showLocalAttachment( "html/images/samplePic.gif" );
1136    */
1137   hwc.showLocalAttachment = function showLocalAttachment(key)
1138   {
1139     hwc.traceEnteringMethod("hwc.showLocalAttachment");
1140     try {
1141       if (hwc.isWindowsMobile() || hwc.isAndroid() ||
1142         hwc.isIOS()) {
1143         hwc.getDataFromContainer("showlocalattachment",
1144           "&key=" + encodeURIComponent(key));
1145       } else if (hwc.isBlackBerry()) {
1146         if (key.indexOf("file://") > -1){
1147           window.location = key;
1148         } else {
1149           window.location = "http://localhost/" +
1150             key;
1151         }
1152       } else {
1153         window.open(key);
1154       }
1155     } finally {
```

```

1152         hwc.traceLeavingMethod("hwc.showLocalAttachment");
1153     }
1154 };
1155
1156 /**
1157  * Internal function to allow the user to cause an operation/
1158  * object query to be invoked. This function should probably
1159  * only be used by designer generated javascript.
1160  *
1161  * @memberOf hwc
1162  * @private
1163  *
1164  * @param {string} credInfo Credential info in the format
1165  * "supusername=usernameValue&suppassword=passwordValue"
1166  * @param {string} serializeDataMessageToSend The data
1167  * message, already serialized. This parameter should be obtained by
1168  * calling serializeToString
1169  * on the result from
1170  * hwc.getMessageValueCollectionForOnlineRequest.
1171  * @param {boolean} hasFileMessageValue Whether the data
1172  * message to send has a file message value. This parameter should be
1173  * obtained by calling
1174  * getHasFileMessageValue on the result from
1175  * hwc.getMessageValueCollectionForOnlineRequest.
1176  * @param {number} timeout Specifies the time, in seconds, to
1177  * wait before giving up waiting for a response.
1178  * @param {string} cacheTimeout Specifies the time, in
1179  * seconds, since the last invocation with the same input parameter
1180  * values to use the same
1181  * response as previously retrieved without making a new call
1182  * to the server. If this parameter is NEVER, the cache content will
1183  * never expire.
1184  * @param {string} errorMessage Specifies the string to
1185  * display if an online request fails.
1186  * @param {anonymous.errorCallbackFunction} errorCallback
1187  * Name of the function to be called if an online request fails. If this
1188  * parameter is null,
1189  * 'reportRMIError' will be used.

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1174     * @param {string} cacheKey String used as the key for this
request in the on-device request result cache.

1175     * @param {string} cachePolicy Specifies cache lookup policy
used by container. If this parameter is 'serverfirst' (ignoring case)
then the cache policy

1176     * used for this online request will be to check the server
before the cache. If this parameter is any other value then a cache
first policy will be used.

1177     * If this parameter is absent and cache is enabled, the
container uses default cache lookup policy to get data from cache if
it is not expired.

1178     * @param {boolean} asynchronous Specifies whether container
will make the request in synchronous or asynchronous mode.

1179     * If this parameter is absent, the container makes the
request to the server in synchronous mode.

1180     */

1181     hwc.doOnlineRequest_CONT = function( credInfo,
1182         serializeDataMessageToSend,
1183         hasFileMessageValue,
1184         timeout, cacheTimeout,
1185         errorMessage, errorCallback,
1186         cacheKey, cachePolicy,
1187         asynchronous) {

1188

1189         hwc.traceEnteringMethod("hwc.doOnlineRequest_CONT");

1190

1191         try {

1192             var request, xmlhttp, response, encodedMessage, url,
funcCall, responseDataType;

1193             request = "xmlWorkflowMessage=" +
encodeURIComponent(serializeDataMessageToSend);

1194

1195             if (credInfo) {

1196                 request += ("&" + credInfo);

1197             }

1198             request += ("&cachekey=" +
encodeURIComponent(cacheKey));
```

```
1199         if (timeout) {
1200             request += ("&rmitimeout=" + timeout);
1201         }
1202         if (cacheTimeout) {
1203             request += ("&RequestExpiry=" + cacheTimeout);
1204         }
1205         if (hasFileMessageValue) {
1206             request += ("&parse=true");
1207         }
1208         if (errorMessage) {
1209             if( hwc.isBlackBerry() ) {
1210                 encodedMessage =
1211 encodeURIComponent(escape(errorMessage));
1212             } else {
1213                 // This is a temporary fix for a bug in the
1214                 // container that calls
1215                 // encodeURIComponent on the whole query string
1216                 // for Android. See
1217                 // IR 676161-2.
1218                 encodedMessage =
1219 encodeURIComponent(errorMessage);
1220             }
1221             request += ("&onErrorMsg=" + encodedMessage);
1222         }
1223         if (!errorCallback) {
1224             errorCallback = "hwc.reportRMIError";
1225         }
1226         if (cachePolicy) {
1227             request += ("&cachePolicy=" + cachePolicy);
1228         }
1229         if (asynchronous) {
1230             request += ("&asynchronous=" + asynchronous);
1231         }
1232         request += ("&onErrorCallback=" + errorCallback);
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1229
1230         if (hwc.isWindowsMobile() || hwc.isWindows()) {
1231             //make xmlhttp request to load the rmi response
from server
1232             xmlhttp = hwc.getXMLHttpRequest();
1233
1234             if (hwc.isWindowsMobile()) {
1235                 xmlhttp.open("POST", "/sup.amp?querytype=rmi&"
+ hwc.versionURLParam, true );
1236
1237                 xmlhttp.onreadystatechange = function() {
1238                     if (xmlhttp.readyState === 4) {
1239                         if (xmlhttp.status === 200 ||
xmlhttp.status === 0) {
1240                             response =
xmlhttp.responseText;
1241                             var responseData =
xmlhttp.getResponseHeader("OnlineRequest-Response-Data-Type");
1242                             processDataMessage(response, null,
null, null, responseData);
1243                         }
1244                     }
1245                 };
1246
1247                 try {
1248                     xmlhttp.send(request);
1249                 }
1250                 catch (excep1) {
1251                     hwc.log("Error: Unable to retrieve the
message from the server", "ERROR", true);
1252                 }
1253             }
1254             else { // hwc.isWindows()
1255                 xmlhttp.open("POST", "rmi.xml", false );
1256                 xmlhttp.send(request);
```

```
1257
1258         //Win32 returns 200 for OK, WM returns 0 for
1259         OK
1260         if (xmlhttp.status === 200 || xmlhttp.status ===
1261         0) {
1262             response = xmlhttp.responseText;
1263             processDataMessage(response);
1264         }
1265         else {
1266             hwc.log("Error: Unable to retrieve the
1267             message from the server", "ERROR", true);
1268         }
1269     }
1270     else if (hwc.isAndroid()) {
1271         url = 'http://localhost/sup.amp?querytype=rmi&' +
1272         hwc.versionURLParam;
1273         funcCall = "_HWC.postData('" + url + "', '" +
1274         request + "')";
1275         // method processDataMessage invoked by native
1276         container.
1277         // funcCall = "processDataMessage(" + funcCall +
1278         ")";
1279         setTimeout(funcCall, 5);
1280     }
1281     else { //BB and iPhone
1282         xmlhttp = hwc.getXMLHttpRequest();
1283         xmlhttp.open("POST", "http://localhost/sup.amp?
1284         querytype=rmi&" + hwc.versionURLParam, true);
1285     }
1286     if (hwc.isBlackBerry()) {
1287         xmlhttp.onreadystatechange = function() {
1288             if (xmlhttp.readyState === 4) {
1289                 if (xmlhttp.status === 200) {
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1284         response =
xmlhttp.responseText;

1285         responseData =
xmlhttp.getResponseHeader("OnlineRequest-Response-Data-Type");

1286         processDataMessage(response, null,
null, null, responseData);
1287     }
1288 }
1289 };
1290 }
1291     try {
1292         xmlhttp.send(request);
1293     }
1294     catch (excep2) {
1295         hwc.log("Error: Unable to retrieve the message
from the server", "ERROR", true);
1296     }
1297 }
1298 } finally {
1299     hwc.traceLeavingMethod("hwc.doOnlineRequest_CONT");
1300 }
1301 };
1302
1303 /**
1304  * Allows the user to cause an operation/object query to be
invoked. This function should probably only be used by
1305  * designer generated javascript.
1306  *
1307  * @memberOf hwc
1308  * @private
1309  *
1310  * @param {string} credInfo Credential info in the format
"supusername=usernameValue&suppassword=passwordValue"
```



```

1311     * @param {string} serializeDataMessageToSend The data
message, already serialized. This parameter should be obtained by
calling serializeToString

1312     * on the result from
hwc.getMessageValueCollectionForOnlineRequest.

1313     * @param {string} attachmentKey The specified key of the
result will not be returned in the data message but will instead be
stored on the

1314     * device for later access via {@link
hwc.showAttachmentFromCache_CONT}.

1315     * @param {string} requestGUID Represents a unique key that
can be used to store/access the cached key value from the request
results.

1316     * @param {callback function} downloadCompleteCallback A
function that will be invoked when the attachment has been downloaded
to the device

1317     * and is ready to be accessed.

1318     */

1319     hwc.doAttachmentDownload_CONT = function(credInfo,
serializeDataMessageToSend, attachmentKey, requestGUID,
downloadCompleteCallback) {

1320     hwc.traceEnteringMethod("hwc.doAttachmentDownload_CONT");

1321         try {

1322             var request, xmlhttp;

1323             request = "xmlWorkflowMessage=" +
encodeURIComponent(serializeDataMessageToSend);

1324

1325             if (credInfo) {

1326                 request += ("&" + credInfo);

1327             }

1328             request += ("&attachmentkey=" + attachmentKey);

1329             request += ("&uniquekey=" + requestGUID);

1330             request += ("&ondownloadcomplete=" +
downloadCompleteCallback);

1331             if (hwc.isWindowsMobile() || hwc.isWindows()) {

1332                 xmlhttp = hwc.getXMLHttpRequest();

1333                 xmlhttp.open("POST", "/sup.amp?
querytype=downloadattachment&" + hwc.versionURLParam, true );

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1334         xmlhttp.onreadystatechange = function() {
1335             if (xmlhttp.readyState === 4) {
1336                 if (xmlhttp.status === 200) {
1337                     window[downloadCompleteCallback].call(this,
1338                     decodeURIComponent(requestGUID), xmlhttp.responseText);
1339                 }
1340             };
1341         try {
1342             xmlhttp.send(request);
1343         }
1344         catch (e3) {}
1345     }
1346     else if (hwc.isAndroid()) {
1347         hwc.postDataToContainer("downloadattachment",
1348         request);
1349     }
1350     else {
1351         xmlhttp = hwc.getXMLHttpRequest();
1352         xmlhttp.open("POST", "http://localhost/sup.amp?
1353         querytype=downloadattachment&" + hwc.versionURLParam, true);
1354         if (hwc.isBlackBerry()) {
1355             xmlhttp.onreadystatechange = function() {
1356                 if (xmlhttp.readyState === 4) {
1357                     if (xmlhttp.status === 200) {
1358                         window[downloadCompleteCallback].call(this,
1359                         decodeURIComponent(requestGUID), xmlhttp.responseText);
1360                     }
1361                 }
1362             };
1363         }
1364     }
1365     try {
```

```
1362         xmlhttp.send(request);
1363     }
1364     catch (e1) {}
1365     }
1366     } finally {
1367 hwc.traceLeavingMethod("hwc.doAttachmentDownload_CONT");
1368     }
1369 };
1370
1371 /**
1372  * Allows the user to cause an operation/object query to be
1373  * invoked. Will close the hybrid app application when finished. This
1374  * function should probably only be used by
1375  * designer generated javascript.
1376  *
1377  * @memberOf hwc
1378  * @private
1379  *
1380  * @param {string} credInfo Credential info in the format
1381  * "supusername=usernameValue&suppassword=passwordValue"
1382  * @param {string} serializeDataMessageToSend The data
1383  * message, already serialized. This parameter should be obtained by
1384  * calling toString
1385  * on the result from
1386  * hwc.getMessageValueCollectionForOnlineRequest.
1387  * @param {boolean} hasFileMessageValue Whether the data
1388  * message to send has a file message value. This parameter should be
1389  * obtained by calling
1390  * getHasFileMessageValue on the result from
1391  * hwc.getMessageValueCollectionForOnlineRequest.
1392  */
1393 hwc.doSubmitWorkflow_CONT = function(credInfo,
1394     serializeDataMessageToSend, hasFileMessageValue) {
1395     hwc.traceEnteringMethod("hwc.doSubmitWorkflow_CONT");
1396     try {
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1387         var request = "xmlWorkflowMessage=" +
encodeURIComponent(serializeDataMessageToSend);
1388
1389         if (credInfo) {
1390             request += ("&" + credInfo);
1391         }
1392         if (hasFileMessageValue) {
1393             request += ("&parse=true");
1394         }
1395
1396         hwc.postMessageToContainer("submit", request);
1397     } finally {
1398         hwc.traceLeavingMethod("hwc.doSubmitWorkflow_CONT");
1399     }
1400 };
1401
1402 /**
1403  * Internal function to allow the user to cause an operation/
object query to be invoked. This function should probably only be
used by
1404  * designer generated javascript.
1405  *
1406  * @memberOf hwc
1407  * @private
1408  *
1409  * @param {string} credInfo Credential info in the format
"supusername=usernameValue&suppassword=passwordValue"
1410  * @param {string} serializeDataMessageToSend The data
message, already serialized. This parameter should be obtained by
calling serializeToString
1411  * on the result from
hwc.getMessageValueCollectionForOnlineRequest.
1412  */
1413     hwc.doActivateWorkflow_CONT = function(credInfo,
serializeDataMessageToSend) {
```

```
1414     var request, xmlhttp;
1415     hwc.traceEnteringMethod("hwc.doActivateWorkflow_CONT");
1416     try {
1417         request = "xmlWorkflowMessage=" +
encodeURIComponent(serializeDataMessageToSend);
1418
1419         if (credInfo) {
1420             request += ("&" + credInfo);
1421         }
1422
1423         hwc.postDataToContainer("activate", request);
1424     } finally {
1425         hwc.traceLeavingMethod("hwc.doActivateWorkflow_CONT");
1426     }
1427 };
1428
1429 /**
1430  * This function should probably only be used by designer
generated javascript.
1431  *
1432  * @memberOf hwc
1433  * @private
1434
1435  * @param {string} credInfo Credential info in the format
"supusername=usernameValue&suppassword=passwordValue"
1436  * @param serializeDataMessageToSend The data message,
already serialized. This parameter should be obtained by calling
serializeToString
1437  * on the result from
hwc.getMessageValueCollectionForOnlineRequest.
1438  */
1439     hwc.doCredentialsSubmit_CONT = function(credInfo,
serializeDataMessageToSend ) {
1440         hwc.traceEnteringMethod("hwc.doCredentialsSubmit_CONT");
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1441     try {
1442         var request = "xmlWorkflowMessage=" +
encodeURIComponent(serializeDataMessageToSend);
1443
1444         if (credInfo) {
1445             request += ("&" + credInfo);
1446         }
1447
1448         hwc.postDataToContainer("credentials", request);
1449     } finally {
1450 hwc.traceLeavingMethod("hwc.doCredentialsSubmit_CONT");
1451     }
1452 };
1453
1454 /**
1455  * This function shows a progress dialog with spinner. The
dialog created by this function will block all
1456  * user input until {@link hwc.hideProgressDialog} is
called. It is important to be sure that
1457  * {@link hwc.hideProgressDialog} will be called after a call
to this function.
1458  *
1459  * @memberOf hwc
1460  * @public
1461  * @param {string} [message] The message to show on the
progress dialog. This message is displayed on Android
1462  * platforms only - other platforms show only a spinner.
1463  * @example
1464  * var showProgress = function()
1465  * {
1466  *     hwc.showProgressDialog( "a message" );
1467  *     setTimeout( hideProgress, 10000 );
1468  * }
```

```
1469     *
1470     * var hideProgress = function()
1471     * {
1472     *     hwc.hideProgressDialog();
1473     * }
1474     */
1475     hwc.showProgressDialog = function(message) {
1476         hwc.traceEnteringMethod("hwc.showProgressDialog");
1477         try {
1478             hwc.getDataFromContainer("showprogressdialog",
1479 "&message=" + message);
1480         } finally {
1481             hwc.traceLeavingMethod("hwc.showProgressDialog");
1482         }
1483     };
1484     /**
1485     * This function hides the progress dialog displaying the
1486 spinner. This function should be used to hide
1487 * the progress dialog after a call to {@link
1488 hwc.showProgressDialog}. If this function is called while there
1489 * is no progress dialog, then nothing will happen.
1490 * @memberOf hwc
1491 * @public
1492 * @example
1493 * var showProgress = function()
1494 * {
1495 *     hwc.showProgressDialog( "a message" );
1496 *     setTimeout( hideProgress, 10000 );
1497 * }
1498 * var hideProgress = function()
1499 * {
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1499     *     hwc.hideProgressDialog();
1500     * }
1501     */
1502     hwc.hideProgressDialog = function() {
1503         hwc.traceEnteringMethod("hwc.hideProgressDialog");
1504         try {
1505             hwc.getDataFromContainer("hideprogressdialog");
1506         } finally {
1507             hwc.traceLeavingMethod("hwc.hideProgressDialog");
1508         }
1509     };
1510
1511
1512     /**
1513     * Displays an alert dialog to the user. This function blocks
1514     * until it receives a response from the user.
1515     * @memberOf hwc
1516     * @public
1517     * @param {string} message The message to display
1518     * @param {string} [title] The title doesn't actually get
1519     * displayed.
1520     * @example
1521     * hwc.showAlertDialog( "This is a fancy alert dialog", "With
1522     * a Title" );
1523     */
1524     hwc.showAlertDialog = function(message, title) {
1525         if(hwc.isIOS()){
1526             // For ios client, creating an IFRAME element for the
1527             // alert message, so as to hide the
1528             // title bar in the alert box
1529             var iframe = document.createElement("IFRAME");
1530             iframe.setAttribute("src", 'data:text/plain');
1531             document.documentElement.appendChild(iframe);
```



```
1529 window.frames[window.frames.length-1].alert(message);
1530         iframe.parentNode.removeChild(iframe);
1531     }
1532     else{
1533         alert(message);
1534     }
1535 };
1536
1537 /**
1538  * Shows a confirm dialog to the user. This function blocks
1539  * until it receives a response from the user.
1540  *
1541  * @memberOf hwc
1542  * @public
1543  * @param {string} message The message to display in the
1544  * dialog.
1545  * @param {string} [title] The title doesn't actually get
1546  * displayed.
1547  * @returns {boolean} The user's choice from the confirm
1548  * dialog.
1549  * @example
1550  * var userConfirm = hwc.showConfirmDialog( "Are you sure you
1551  * want to see an alert message?", "Confirm Alert" );
1552  * if( userConfirm )
1553  * {
1554  *     alert( "This is what you wanted." );
1555  * }
1556  */
1557 hwc.showConfirmDialog = function(message, title) {
1558     return confirm(message);
1559 };
1560
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1557  /**
1558   * This function closes the hybrid app.
1559   * @memberOf hwc
1560   * @public
1561   * @example
1562   * hwc.close();
1563   */
1564  hwc.close = function() {
1565      workflowMessage = "";
1566      hwc.supUserName = "";
1567      hwc.traceEnteringMethod("hwc.close");
1568      try {
1569          if (hwc.isWindowsMobile()) {
1570              if (typeof(hwc.setWindowBlankScreen) ===
1571                  'function')
1572                  {
1573                      hwc.setWindowBlankScreen();
1574                  }
1575                  hwc.getDataFromContainer("close");
1576              }
1577              else if (hwc.isIOS()) {
1578                  hwc.getDataFromContainer("close");
1579              }
1580              else if (hwc.isAndroid()) {
1581                  hwc.log("Closing Hybrid App" , "INFO");
1582                  _HWC.close();
1583              }
1584              else {
1585                  window.close();
1586              }
1587          }
```

```
1588         status = hwc.STATUS.CLOSED;
1589     } finally {
1590         hwc.traceLeavingMethod("hwc.close");
1591     }
1592 };
1593
1594 /**
1595  * This function checks if the hybrid app has been closed.
1596  * @returns {boolean} true if hybrid app is closed, otherwise
1597  * false.
1598  * @memberOf hwc
1599  * @public
1600  * @example
1601  * hwc.isClosed();
1602  */
1603 hwc.isClosed = function() {
1604     return status === hwc.STATUS.CLOSED;
1605 };
1606 })(hwc, window);
1607
1608 /**
1609  * A callback function invoked when {@link hwc.log} is invoked
1610  * with true for the notifyUser parameter.
1611  * This callback should notify the user of the log message in
1612  * an appropriate manner.
1613  *
1614  * @name anonymous.alertDialogCallbackFunction
1615  *
1616  * @param {string} message The message that the user should be
1617  * notified of.
1618  * @function
1619  */
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
1618
1619  /**
1620   * A callback function invoked if there is an error.
1621   *
1622   * @name anonymous.errorCallbackFunction
1623   *
1624   * @param {string} errorMessage The message describing the
1625   * error.
1626   *
1627   * @function
1628   */
1629  /**
1630   * A generic callback function that takes no parameters. Used
1631   * to execute code when a certain event occurs.
1632   *
1633   * @name anonymous.genericCallbackFunction
1634   *
1635   * @function
1636   */
```

hwc-utils.js

```
1  /**
2   * Sybase Hybrid App version 2.3.4
3   *
4   * Utils_CONT.js - container maintained aspect
5   *
6   * This file will not be regenerated, so it is possible to
7   * modify it, but it
8   *
9   * is not recommended.
10  *
11  * The template used to create this file was compiled on Thu
12  * Jun 07 14:57:11 EDT 2012
13  *
14  */
```

```

11     * Copyright (c) 2012 Sybase Inc. All rights reserved.
12     */
13     /**
14     * The namespace for the Hybrid Web Container javascript
15     * @namespace */
16     hwc = (typeof hwc === "undefined" || !hwc) ? {} : hwc;    //
SUP 'namespace'
17
18     /**
19     * Container Utilities
20     */
21     (function(hwc, window, undefined) {
22
23         /***** PUBLIC CONSTANTS
24         *****/
25         /** @private */
26         hwc.versionURLParam = "version=2.2";
27
28         /***** PUBLIC API *****/
29
30         /**
31         * The version number sent with the HTTP messages to the
32         native code.
33         * Used for internal versioning only
34         * @private
35         * @returns {String} the version string
36         */
37         hwc.getVersionURLParam = function() {
38             return hwc.versionURLParam;
39         };
40

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
41     /**
42     * Internal worker for initial HybridApp loading.
43     * Returns the response message or empty.
44     * @private
45     */
46     hwc.onHybridAppLoad_CONT = function() {
47         var response = hwc.getTransformData();
48         processDataMessage(response, false, true);
49     };
50
51     /**
52     * Returns the transform data for the hybridapp. Only a
53     * server-initiated app will have this data.
54     * @example
55     * TODO: Add an example
56     * @returns the transform data.
57     * @public
58     * @memberOf hwc
59     */
60     hwc.getTransformData = function() {
61         var xmlhttp;
62         hwc.traceEnteringMethod("hwc.getTransformData");
63         try {
64             if (hwc.isWindows()) {
65                 xmlhttp = hwc.getXMLHTTPPreRequest();
66                 xmlhttp.open("GET", "transform.xml", false);
67                 xmlhttp.send("");
68                 if (xmlhttp.status === 200 || xmlhttp.status === 0)
69                 { //Win32 returns 200 for OK, WM returns 0 for OK
70                     return xmlhttp.responseText;
71                 }
72             }
73         }
74     }
75 }
```

```
72         else
73         {
74             return
hwc.getDataFromContainer("loadtransformdata");
75         }
76     } finally {
77         hwc.traceLeavingMethod("hwc.getTransformData");
78     }
79 };
80
81 /**Internal worker for adding a single menu item.
82  * @private
83  * @param {string} menuStr Information of the menu.
84  */
85 hwc.addNativeMenuItem_CONT = function (menuStr ) {
86     hwc.postDataToContainer("addallmenuitems",
"menuitems=" + encodeURIComponent(menuStr));
87 };
88
89 /**Internal worker setting credential information.
90  * @private
91  * @param {string} credInfo Information of the credential.
92  */
93 hwc.handleCredentialChange_CONT = function(credInfo) {
94     var requestData = credInfo ? credInfo : "";
95     if (requestData) {
96         if (!hwc.isWindows()) {
97             hwc.postDataToContainer("formredirect",
requestData);
98         }
99     }
100 };
101
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
102     /**
103     * Removes spaces from the specified string.
104     * @private
105     * @param {string} str The specified string
106     * @param {boolean} leftAndRightOnly When true removes
leading and trailing spaces
107     * @returns {string} The trimmed string
108     * @memberOf hwc
109     */
110     hwc.trimSpaces = function(str, leftAndRightOnly) {
111         if (leftAndRightOnly) {
112             return str.replace(/^\s+|\s+$/g, "");
113         }
114         return str.replace(/\s+/g, '');
115     };
116
117     /** @private
118     * @memberOf hwc
119     * @param {string} value The string to be parsed.
120     */
121     hwc.parseBoolean = function(value) {
122         if (value) {
123             return hwc.trimSpaces(value, true).toLowerCase() ===
"true";
124         }
125         else {
126             return false;
127         }
128     };
129
130     /**
131     * Extract the error message from a URL string. The parameter
name of the error message should be "onErrorMsg".
```



```

132     *
133     * @param {String} errString The error string URL
134     * @returns {String} error message
135     * @memberOf hwc
136     * @public
137     */
138     hwc.getOnErrorMessageFromNativeError = function
getOnErrorMessageFromNativeError(errString) {
139     hwc.traceEnteringMethod("hwc.getOnErrorMessageFromNativeError");
140         try {
141             if( hwc.isBlackBerry() ) {
142                 return
unescape(hwc.getURLParamFromNativeError("onErrorMsg", errString));
143             } else {
144                 // This is a temporary fix for a bug in the
container that calls
145                 // encodeURIComponent on the whole query string
for Android. See
146                 // IR 676161-2.
147                 return
hwc.getURLParamFromNativeError("onErrorMsg", errString);
148             }
149         } finally {
150     hwc.traceLeavingMethod("hwc.getOnErrorMessageFromNativeError");
151         }
152     };
153
154     /**
155     * Extract the error call back method name from a URL string.
The parameter name of the error call back method should be
"onErrorCallback".
156     * @param {String} errString The error string URL
157     * @returns {String} the error callback method name
158     * @memberOf hwc

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
159     * @public
160     */
161     hwc.getCallbackFromNativeError = function
getCallbackFromNativeError(errString) {
162         hwc.traceEnteringMethod("hwc.getCallbackFromNativeError");
163         try {
164             return
hwc.getURLParamFromNativeError("onErrorCallback", errString);
165         } finally {
166             hwc.traceLeavingMethod("hwc.getCallbackFromNativeError");
167         }
168     };
169
170     /**
171     * Extract an error code from a URL string. The parameter name
of the error code should be "errorCode".
172     * @example
173     * TODO: CONFIRM THE RETURN DATATYPE
174     * @param {String} errString The error string URL
175     * @returns {String} error code
176     * @memberOf hwc
177     * @public
178     */
179     hwc.getCodeFromNativeError = function
getCodeFromNativeError( errString ) {
180         hwc.traceEnteringMethod("hwc.getCodeFromNativeError");
181         try {
182             return hwc.getURLParamFromNativeError("errorCode",
errString);
183         } finally {
184             hwc.traceLeavingMethod("hwc.getCodeFromNativeError");
185         }
```

```
186     };
187
188     /**
189     * Extract a native message from a URL string. The parameter
190     * name of the native message should be "nativeErrMsg".
191     * @param {String} errString The error string URL
192     * @returns {String} the native message
193     * @memberOf hwc
194     * @public
195     */
196     hwc.getNativeMessageFromNativeError = function
197     getNativeMessageFromNativeError( errString ) {
198         hwc.traceEnteringMethod("hwc.getNativeMessageFromNativeError");
199         try {
200             return hwc.getURLParamFromNativeError("nativeErrMsg",
201             errString);
202         } finally {
203             hwc.traceLeavingMethod("hwc.getNativeMessageFromNativeError");
204         }
205     };
206
207     /**
208     * Extract a parameter value from a URL string with a given
209     * parameter name.
210     * @param {String} paramName The parameter name
211     * @param {String} url The containing URL of the parameter
212     * @returns {String} The parameter value
213     * @memberOf hwc
214     * @public
215     */
216     hwc.getURLParamFromNativeError = function
217     getURLParamFromNativeError( paramName, url ) {
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
213         var idxofS, idxofE, pName, pValue, paramSection, ret,
paramSectionsAmp, ampSections, idxofA;
214
215 hwc.traceEnteringMethod("hwc.getURLParamFromNativeError");
216         try {
217             if( hwc.isBlackBerry() ) {
218                 paramSection = url;
219             } else {
220                 // This is a temporary fix for a bug in the
container that calls
221                 // encodeURIComponent on the whole query string
for Android. See
222                 // IR 676161-2.
223                 paramSection = decodeURIComponent(url);
224             }
225             idxofA = paramSection.indexOf("&");
226             if (idxofA > 0) { //there is one or more parameters
in the & section
227                 paramSectionsAmp =
paramSection.substring(idxofA + 1);
228                 ampSections = paramSection.split("&");
229                 if (ampSections.length === 1) {
230                     idxofE = paramSectionsAmp.indexOf("=");
231                     pName = paramSectionsAmp.substring(0,
idxofE);
232                     if (pName.toLowerCase() ===
paramName.toLowerCase()) {
233                         pValue =
paramSectionsAmp.substring(idxofE + 1);
234                         ret = decodeURIComponent( pValue);
235                         return ret;
236                     }
237                 } else { //multiple parameters in the &
section
238                     for (idxofS in ampSections) {
```

```

239                                     idxofE =
ampSections[idxofS].indexOf("=");

240                                     pName =
ampSections[idxofS].substring(0, idxofE);

241                                     if (pName.toLowerCase() ===
paramName.toLowerCase()) {

242                                     pValue =
ampSections[idxofS].substring(idxofE + 1);

243                                     ret =
decodeURIComponent( pValue) ;

244                                     return ret;

245                                     }

246                                     }

247                                     }

248                                     //ok did not find paramName in & section look
for it at the start

249                                     idxofE = paramSection.indexOf("=");

250                                     pName = paramSection.substring(0, idxofE);

251                                     if (pName.toLowerCase() ===
paramName.toLowerCase()) {

252                                     pValue = paramSection.substring(idxofE + 1,
idxofA);

253                                     ret = decodeURIComponent( pValue );

254                                     return ret;

255                                     }

256                                     } else { //only one param

257                                     idxofE = paramSection.indexOf("=");

258                                     pName = paramSection.substring(0, idxofE);

259                                     if (pName.toLowerCase() ===
paramName.toLowerCase()) {

260                                     pValue = paramSection.substring(idxofE +
1);

261                                     ret = decodeURIComponent( pValue );

262                                     return ret;

263                                     }

264                                     }

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
265         return pValue;
266     } finally {
267
268         hwc.traceLeavingMethod("hwc.getURLParamFromNativeError");
269     };
270
271     /**
272     * Log the behavior of entering a JavaScript method.
273     * @param {String} methodName The target method name.
274     * @private
275     */
276     hwc.traceEnteringMethod = function (methodName) {
277         if (hwc.getLoggingCurrentLevel() >= 4) { hwc.log("entering
278         " + methodName + "()", "DEBUG", false); }
279     }
280
281     /**
282     * Log the behavior of leaving a JavaScript method.
283     * @param {String} methodName The target method name.
284     * @private
285     */
286     hwc.traceLeavingMethod = function (methodName) {
287         if (hwc.getLoggingCurrentLevel() >= 4) { hwc.log("exiting
288         " + methodName + "()", "DEBUG", false); }
289     }
290 })(hwc, window);
```

PlatformIdentification.js

```
1     /*
2     * Sybase Hybrid App version 2.3.4
3     */
```

```

4      * PlatformIdentification.js
5      * This file will not be regenerated, so it is possible to
modify it, but it
6      * is not recommended.
7      *
8      * Original file date: 2012-Oct-22
9      * Copyright (c) 2012 Sybase Inc. All rights reserved.
10     */
11
12     /**
13      * The namespace for the Hybrid Web Container javascript
14      * @namespace
15      */
16     hwc = (typeof hwc === "undefined" || !hwc) ? {} :
hwc;
17     // SUP 'namespace'
18
19     (function(hwc, window, undefined) {
20         // private variables
21         // platform identifiers are all calculated once and
cached
22         var _isIOS = false, _isIPad = false, _isIOS5 = false,
_isIOS6 = false, _isIOS7 = false, _isIOS4 = false,
23         _isBB = false, _isBB5 = false, _isBB5Touch = false,
_isBB6NonTouch = false, _isBB7 = false,
24         _isAndroid = false, _isAndroid3 = false,
25         _isWindows = false, _isWinMobile = false;
26
27         // public API
28         /**
29          * Returns true if the hybrid app application is being run
on an iOS (e.g. iPhone, iPad) platform.
30          * @desc Platform
31          * @memberOf hwc
32          * @public

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
32     * @returns {boolean} True if the hybrid app application is
being run on an iOS (e.g. iPhone, iPad) platform.
33     */
34     hwc.isiOS = function() { return _isiOS; };
35     /**
36     * Returns true if the hybrid app application is being run
on an iPad.
37     * @desc Platform
38     * @memberOf hwc
39     * @public
40     * @returns {boolean} True if the hybrid app application is
being run on an iPad.
41     */
42     hwc.isIPad = function() { return _isIPad; };
43     /**
44     * Returns true if the hybrid app application is being run
on iOS5
45     * @desc Platform
46     * @memberOf hwc
47     * @public
48     * @returns {boolean} True if the hybrid app application is
being run on iOS5
49     */
50     hwc.isiOS5 = function() { return _isiOS5; };
51     /**
52     * Returns true if the hybrid app application is being run
on iOS6
53     * @memberOf hwc
54     * @public
55     * @returns {boolean} True if the hybrid app application is
being run on iOS6
56     */
57     hwc.isiOS6 = function() { return _isiOS6; };
58     /**
```



```

59      * Returns true if the hybrid app application is being run
on iOS7
60      * @public
61      * @return {boolean} True if the hybrid app application is
being run on iOS7
62      * @memberOf hwc
63      * @public
64      */
65      hwc.isIOS7 = function() { return _isIOS7; };
66
67      /**
68      * Returns true if the hybrid app application is being run
on iOS4
69      * @feature Platform
70      * @return {Boolean} True if the hybrid app application is
being run on iOS4
71      * @memberOf hwc
72      * @public
73      */
74      hwc.isIOS4 = function() { return _isIOS4; };
75
76      /**
77      * Returns true if the hybrid app application is being run
on a BlackBerry platform.
78      * @desc Platform
79      * @memberOf hwc
80      * @public
81      * @returns {boolean} True if the hybrid app application is
being run on a BlackBerry platform.
82      */
83      hwc.isBlackBerry = function() { return
_isBB; };
84      /**
85      * Returns true if the hybrid app application is being run
on a BlackBerry 5.0 OS

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
86      * @desc Platform
87      * @memberOf hwc
88      * @public
89      * @returns {boolean} True if the hybrid app application is
being run on a BlackBerry 5.0 OS
90      */
91      hwc.isBlackBerry5 = function() { return
_isBB5; };
92      /**
93      * Returns true if the hybrid app application is being run
on a BlackBerry 7.x OS
94      * @desc Platform
95      * @memberOf hwc
96      * @public
97      * @returns {boolean} True if the hybrid app application is
being run on a BlackBerry 7.x OS
98      */
99      hwc.isBlackBerry7 = function() { return
_isBB7; };
100     /**
101     * Returns true if the hybrid app application is being run
on a BlackBerry 5.0 OS with a touch screen
102     * @desc Platform
103     * @memberOf hwc
104     * @public
105     * @returns {boolean} True if the hybrid app application is
being run on a BlackBerry 5.0 OS with a touch screen
106     */
107     hwc.isBlackBerry5WithTouchScreen = function() { return
_isBB5Touch; };
108     /**
109     * Returns true if the hybrid app application is being run
on a BlackBerry 6.0 OS without a touch screen
110     * @desc Platform
111     * @memberOf hwc
112     * @public
```

```

113     * @returns {boolean} True if the hybrid app application is
being run on a BlackBerry 6.0 OS without a touch screen
114     */
115     hwc.isBlackBerry6NonTouchScreen = function() { return
_isBB6NonTouch; };
116
117     /**
118     * Returns true if the hybrid app application is being run
on a Windows Mobile platform.
119     * @desc Platform
120     * @memberOf hwc
121     * @public
122     * @returns {boolean} True if the hybrid app application is
being run on a Windows Mobile platform.
123     */
124     hwc.isWindowsMobile = function() { return
_isWinMobile; };
125     /**
126     * Returns true if the hybrid app application is being run
on a Windows platform.
127     * @desc Platform
128     * @memberOf hwc
129     * @public
130     * @returns {boolean} True if the hybrid app application is
being run on a Windows platform.
131     */
132     hwc.isWindows = function() { return
_isWindows; };
133
134     /**
135     * Returns true if the hybrid app application is being run
on an Android 3.0 OS
136     * @desc Platform
137     * @memberOf hwc
138     * @public

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
139     * @returns {boolean} True if the hybrid app application is
being run on an Android 3.0 OS
140     */
141     hwc.isAndroid3 = function() { return _isAndroid3; };
142     /**
143     * Returns true if the hybrid app application is being run
on an Android platform.
144     * @desc Platform
145     * @memberOf hwc
146     * @public
147     * @returns {boolean} True if the hybrid app application is
being run on an Android platform.
148     */
149     hwc.isAndroid = function() { return _isAndroid; };
150
151
152     /**
153     * @private
154     * @returns {boolean} True if this code is running on a
Blackberry 5 OS
155     */
156     function _isBlackBerry5() {
157         var ua = navigator.userAgent;
158         if (ua.indexOf("BlackBerry 9800") >= 0) {
159             return false;
160         }
161         if (ua.match(/5\.[0-9]\.[0-9]/i) !== null) {
162             return true;
163         }
164         return false;
165     }
166
167     /**
168     * @private
```

```
169      * @returns {boolean} True if this code is running on a
Blackberry 5 OS with a touch screen
170      */
171      function _isBlackBerry5WithTouchScreen() {
172          if (isBlackBerry5()) {
173              var ua = navigator.userAgent;
174              if (ua.length > 12 && ua.substring(0, 12) ===
"BlackBerry95") {
175                  return true;
176              }
177          }
178          return false;
179      }
180
181      /**
182      * @private
183      * @returns {boolean} True if this code is running on a
Blackberry 6 OS with no touch screen
184      */
185      function _isBlackBerry6NonTouchScreen() {
186          if (navigator.userAgent.match(/Version\/6./i)) {
187              var ua = navigator.userAgent;
188              if ((ua.indexOf('9780') > 0) || (ua.indexOf('9700')
> 0) || (ua.indexOf('9650') > 0) || (ua.indexOf('9300') > 0) ||
(ua.indexOf('9330') > 0)) {
189                  return true;
190              }
191          }
192          return false;
193      }
194
195      /**
196      * @private
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
197      * @returns {boolean} True if this code is running on a
Blackberry & OS
198      */
199      function _isBlackBerry7() {
200          if (navigator.userAgent.match(/Version\/7\.[0-9]\.
[0-9]/i) !== null){
201              return true;
202          }
203          else {
204              return false;
205          }
206      }
207
208
209      /**
210      * Execute once to identify and cache
211      */
212      {
213          // apple products
214          _isIOS = ((navigator.platform.indexOf("i") ===
0));
215          if( _isIOS ) {
216              _isIOS4 = (navigator.userAgent.match(/OS 4_[0-9_]+
like Mac OS X/i) !== null);
217              _isIOS5 = (navigator.userAgent.match(/OS 5_[0-9_]+
like Mac OS X/i) !== null);
218              _isIOS6 = (navigator.userAgent.match(/OS 6_[0-9_]+
like Mac OS X/i) !== null);
219              _isIOS7 = (navigator.userAgent.match(/OS 7_[0-9_]+
like Mac OS X/i) !== null);
220              _isIPad = (navigator.userAgent.match(/iPad/i) !==
null);
221          }
222
223          // BlackBerry
```

```

224     _isBB = (navigator.platform === "BlackBerry");
225     if( _isBB ) {
226         _isBB5 = _isBlackBerry5();
227         _isBB7 = _isBlackBerry7();
228         _isBB5Touch =
_isBlackBerry5WithTouchScreen();
229         _isBB6NonTouch =
_isBlackBerry6NonTouchScreen();
230     }
231
232     // Android
233     _isAndroid = (navigator.userAgent.indexOf("Android")
> -1);
234     if( _isAndroid ) {
235         _isAndroid3 = (navigator.userAgent.indexOf("3.0") >
-1);
236     }
237
238     // Windows
239     _isWinMobile = (navigator.platform === "WinCE");
240     _isWindows = ( (navigator.platform === "Win32") ||
(navigator.platform === "Win64") || (navigator.platform ===
"MacIntel") ||
241         ( !_isAndroid &&
(navigator.platform.indexOf("Linux") === 0) ) );
242
243     //alert("Platform Identified: Win=" + _isWindows + ",
BB=" + _isBB);
244     }
245     })(hwc, window);
246
247
248
249     /**
250     * Returns true if the hybrid app application is being run on
an iOS (e.g. iPhone, iPad) platform.

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
251     * @private
252     * @returns {boolean} True if the hybrid app application is
being run on an iOS (e.g. iPhone, iPad) platform.
253     */
254     function isIOS() { return hwc.isIOS(); }
255
256     /**
257     * Returns true if the hybrid app application is being run on
iOS5
258     * @private
259     * @returns {boolean} True if the hybrid app application is
being run on iOS5
260     */
261     function isIOS5() { return hwc.isIOS5(); }
262
263     /**
264     * Returns true if the hybrid app application is being run on
an iPad.
265     * @private
266     * @returns {boolean} True if the hybrid app application is
being run on an iPad.
267     */
268     function isIPad() { return hwc.isIPad(); }
269
270     /**
271     * Returns true if the hybrid app application is being run on
a BlackBerry platform.
272     * @private
273     * @returns {boolean} True if the hybrid app application is
being run on a BlackBerry platform.
274     */
275     function isBlackBerry() { return hwc.isBlackBerry(); }
276
277     /**
```



```

278     * Returns true if the hybrid app application is being run on
a BlackBerry 5.0 OS
279     * @private
280     * @returns {boolean} True if the hybrid app application is
being run on a BlackBerry 5.0 OS
281     */
282     function isBlackBerry5() { return hwc.isBlackBerry5(); }
283
284     /**
285     * Returns true if the hybrid app application is being run on
a BlackBerry 5.0 OS with a touch screen
286     * @private
287     * @returns {boolean} True if the hybrid app application is
being run on a BlackBerry 5.0 OS with a touch screen
288     */
289     function isBlackBerry5WithTouchScreen() { return
hwc.isBlackBerry5WithTouchScreen(); }
290
291     /**
292     * Returns true if the hybrid app application is being run on
a BlackBerry 6.0 OS without a touch screen
293     * @private
294     * @returns {boolean} True if the hybrid app application is
being run on a BlackBerry 6.0 OS without a touch screen
295     */
296     function isBlackBerry6NonTouchScreen() { return
hwc.isBlackBerry6NonTouchScreen(); }
297
298     /**
299     * Returns true if the hybrid app application is being run on
a BlackBerry 7.x OS
300     * @private
301     * @returns {boolean} True if the hybrid app application is
being run on a BlackBerry 7.x OS
302     */
303     function isBlackBerry7() { return hwc.isBlackBerry7(); }

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
304
305     /**
306     * Returns true if the hybrid app application is being run on
a Windows Mobile platform.
307     * @private
308     * @returns {boolean} True if the hybrid app application is
being run on a Windows Mobile platform.
309     */
310     function isWindowsMobile() { return
hwc.isWindowsMobile(); }
311
312     /**
313     * Returns true if the hybrid app application is being run on
a Windows platform.
314     * @private
315     * @returns {boolean} True if the hybrid app application is
being run on a Windows platform.
316     */
317     function isWindows() { return hwc.isWindows(); }
318
319     /**
320     * Returns true if the hybrid app application is being run on
an Android platform.
321     * @private
322     * @returns {boolean} True if the hybrid app application is
being run on an Android platform.
323     */
324     function isAndroid() { return hwc.isAndroid(); }
325
326     /**
327     * Returns true if the hybrid app application is being run on
an Android 3.0 OS
328     * @private
329     * @returns {boolean} True if the hybrid app application is
being run on an Android 3.0 OS
330     */
```

```

331     function isAndroid3() { return hwc.isAndroid3(); }
332

```

Plugins/AppLog/applog.js

```

1     /*
2         * Sybase Hybrid App version 2.3.4
3         * Sybase PhoneGap AppLog plugin
4         *
5         * applog.js
6         * This file will not be regenerated, so it is possible to
modify it, but it
7         * is not recommended.
8         *
9         * Copyright (c) 2013 Sybase Inc. All rights reserved.
10        */
11
12    /**
13        * The namespace for AppLog plugin
14        * @namespace
15        */
16    AppLog = (typeof AppLog === "undefined" || !AppLog) ? {} :
AppLog; // 'namespace'
17
18    (function(AppLog, window, undefined) {
19        /**
20            * Constant indicating the operation failed with unknown
error. Used in {@link anonymous.AppLogErrorCallbackParameter}.
21            * @type number
22            */
23        AppLog.ERR_UNKNOWN = -1;
24
25        /**
26            * Constant indicating an app log entry is associated with
an unknown event. Used in {@link AppLog.LogEntry}.

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
27      * @type number
28      */
29      AppLog.STATUS_EVENT_UNKNOWN          = 1;
30
31      /**
32      * Constant indicating an app log entry is associated with
33      * starting the client connection to the SUP server. Used in {@link
34      * AppLog.LogEntry}.
35      * @type number
36      */
37      AppLog.STATUS_EVENT_STARTUP          = 2;
38
39      /**
40      * Constant indicating an app log entry is associated with
41      * shutting down the client connection to the SUP server. Used in {@link
42      * AppLog.LogEntry}.
43      * @type number
44      */
45      AppLog.STATUS_EVENT_SHUTDOWN        = 3;
46
47      /**
48      * Constant indicating an app log entry is associated with
49      * restarting the client connection to the SUP server. Used in {@link
50      * AppLog.LogEntry}.
51      * @type number
52      */
53      AppLog.STATUS_EVENT_RESTART         = 4;
54
55      /**
56      * Constant indicating an app log entry is associated with
57      * the client successfully connecting to the SUP server. Used in {@link
58      * AppLog.LogEntry}.
59      * @type number
60      */
61      AppLog.STATUS_EVENT_CONNECTED       = 5;
```

```
54
55     /**
56     * Constant indicating an app log entry is associated with
57     * the client losing connection to the SUP server. Used in {@link
58     * AppLog.LogEntry}.
59     * @type number
60     */
61     AppLog.STATUS_EVENT_DISCONNECTED = 6;
62
63     /**
64     * Constant indicating an app log entry is associated with
65     * the client going into flight mode. Used in {@link AppLog.LogEntry}.
66     * @type number
67     */
68     AppLog.STATUS_EVENT_FLIGHT_MODE = 7;
69
70     /**
71     * Constant indicating an app log entry is associated with
72     * the client going out of network. Used in {@link AppLog.LogEntry}.
73     * @type number
74     */
75     AppLog.STATUS_EVENT_OUT_OF_NETWORK = 8;
76
77     /**
78     * Constant indicating an app log entry is associated with
79     * the client waiting to connect to the SUP server. Used in {@link
80     * AppLog.LogEntry}.
81     * @type number
82     */
83     AppLog.STATUS_EVENT_WAITING_TO_CONNECT = 9;
84
85     /**
86     * Constant indicating an app log entry is associated with
87     * the client losing connection to the SUP server due to roaming. Used
88     * in {@link AppLog.LogEntry}.
89     * @type number
90     */
91     AppLog.STATUS_EVENT_ROAMING = 10;
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
81      * @type number
82      */
83      AppLog.STATUS_EVENT_DISCONNECTED_ROAMING = 10;
84
85      /**
86      * Constant indicating an app log entry is associated with
the client losing connection to the SUP server due to low storage.
Used in {@link AppLog.LogEntry}.
87      * @type number
88      */
89      AppLog.STATUS_EVENT_DISCONNECTED_LOW_STORAGE = 11;
90
91      /**
92      * Constant indicating an app log entry is associated with
the client starting registration. Used in {@link AppLog.LogEntry}.
93      * @type number
94      */
95      AppLog.STATUS_EVENT_REGISTRATION_STARTED = 12;
96
97      /**
98      * Constant indicating an app log entry is associated with
the client receiving a notification. Used in {@link
AppLog.LogEntry}.
99      * @type number
100     */
101     AppLog.STATUS_EVENT_NOTIFICATION_RECEIVED = 13;
102
103     /**
104     * Constant indicating an app log entry is associated with
a default app being set from the server. Used in {@link
AppLog.LogEntry}.
105     * @type number
106     */
107     AppLog.STATUS_EVENT_SET_DEFAULT_ITEM = 14;
108
```

```

109      /**
110      * Constant indicating an app log entry is associated with
111      * a default app being unset from the server. Used in {@link
112      * AppLog.LogEntry}.
113      * @type number
114      */
115      AppLog.STATUS_EVENT_UNSET_DEFAULT_ITEM = 15;
116
117      /**
118      * This object represents a log entry.
119      *
120      * @classdesc
121      * @public
122      * @memberOf AppLog
123      * @param {number} logDate The date the log entry was
124      * recorded, in milliseconds since January 1, 1970, 00:00:00 GMT.
125      * @param {number} event The event ID of the log entry (will
126      * be one of the AppLog status events, or possibly a custom value).
127      * @param {string} msg The message of the log entry.
128      */
129      AppLog.LogEntry = function ( logDate, event, msg ) {
130
131          this.date = logDate;
132          this.statusCode = event;
133          this.message = msg;
134      };
135
136      /**
137      * Call this function to get an array of {@link
138      * AppLog.LogEntry} objects. There will be one
139      *
140      * @param {number} object for each line in the app
141      * log.
142      *
143      * @public
144      * @memberOf AppLog

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
137      * @param {anonymous.getLogEntriesSuccessCallback}
successCB The callback function that will receive the asynchronous
138      * callback with the log entries.
139      * @param {anonymous.getLogEntriesErrorCallback} errorCallback
The callback function that will be invoked on errors.
140      *
141      * @example
142      * // A global function called with the log entries.
143      * function onLogEntriesSuccessCallback(data) {
144      *     for ( var i = 0; i < data.length; i++ )
145      *     {
146      *         var logEntry = data[ i ];
147      *         alert('Log entry ' + ( i + 1 ) + ':\n' +
148      *             'Date (ms): ' + logEntry.date + '\n' +
149      *             'Status code: ' + logEntry.statusCode + '\n'
+
150      *             'Message: ' + logEntry.message
151      *             );
152      *     }
153      * }
154      *
155      * // A global function called if there is an error
retrieving log entries.
156      * function onLogEntriesFailureCallback(error) {
157      *     alert('Error retrieving log entries: ' + error);
158      * }
159      *
160      * // Get the log entries
161      * AppLog.getLogEntries(onLogEntriesSuccessCallback,
onLogEntriesFailureCallback);
162      */
163      AppLog.getLogEntries = function( successCB, errorCallback ) {
164          try
165          {
```



```

166         cordova.exec( successCB, errorCallback, "AppLog",
"getLogEntries", [] );
167     }
168     catch (ex)
169     {
170         setTimeout( errorCallback({errorCode : AppLog.ERR_UNKNOWN,
description : ex.message}), 0 );
171     }
172 }
173
174 /**
175  * Registers a log listener.
176  *
177  * @public
178  * @memberOf AppLog
179  * @param
{anonymous.startOrStopLogListenerSuccessCallback} successCB A
callback function that will be invoked
180  * if the log listener is successfully registered.
181  * @param {anonymous.startOrStopLogListenerErrorCallback}
errorCB A callback function that will be invoked if there
182  * is an error registering the log listener.
183  * @param {anonymous.logListener} logListener The callback
to register. This will be invoked when new entries are added to the
log.
184  * @param {Object} [containingObject] Object containing the
definition for logListener. If a log listener callback function
185  * references variables in its containing object, then the
containing object should be passed to this function.
186  *
187  * @example
188  * // This example shows how to use this function with a
globally-scoped logListener.
189  * // A global function called by the log listener.
190  * var doSomething = function()
191  * {

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
192     *     alert("this gets displayed when there is a new log
entry.");
193     * }
194     *
195     * // The log listener callback function that will be
passed to AppLog.startLogListener.
196     * // This function will be invoked whenever there is a new
log entry.
197     * var logListener = function( date, statusCode,
message )
198     * {
199     *     doSomething();
200     * }
201     *
202     * function onStartLogListenerSuccessCallback() {
203     *     // Do something here after listener has been added
204     * }
205     *
206     * function onStartLogListenerFailureCallback(error) {
207     *     // React to error here
208     * }
209     *
210     * // Add the log listener.
211     *
AppLog.startLogListener( onStartLogListenerSuccessCallback,
212     *
onStartLogListenerFailureCallback,
213     *
logListener );
214     *
215     * @example
216     * // This example shows how to use this function with a
logListener contained in an object.
217     * // logListenerManager is an object that will contain the
listener callback as well
218     * // as a function that will be invoked from the listener
callback function.
```

```
219     * var logListenerManager = {};
220     *
221     * // This is a function that is called from the listener
callback.
222     * logListenerManager.doSomething = function()
223     * {
224     *     alert("this gets displayed when there is a new log
entry.");
225     * }
226     *
227     * // This is the listener callback that will be passed to
AppLog.startLogListener.
228     * // Since a variable is referenced from the containing
object, the containing object
229     * // will need to be passed to AppLog.startLogListener.
230     * logListenerManager.listener = function( date,
statusCode, message )
231     * {
232     *     this.doSomething();
233     * }
234     *
235     * function onStartLogListenerSuccessCallback() {
236     *     // Do something here after listener has been added
237     * }
238     *
239     * function onStartLogListenerFailureCallback(error) {
240     *     // React to error here
241     * }
242     *
243     * // Pass both the listener callback and the containing
object.
244     *
AppLog.startLogListener( onStartLogListenerSuccessCallback,
245     *
onStartLogListenerFailureCallback,
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
246         *                               logListenerManager.listener,
247         *                               logListenerManager );
248     */
249     AppLog.startLogListener = function( successCB, errorCallback,
logListener, containingObject ) {
250         try
251         {
252             if ( !logListener || ( typeof logListener !==
"function" ) )
253             {
254                 throw new Error( "AppLog.startLogListener Error:
logListener is not a function" );
255             }
256
257             var newListener = new
AppLog.logListenerCallBack( logListener, containingObject );
258             AppLog.logListeners_internal.push( newListener );
259             if ( AppLog.logListeners_internal.length === 1 )
260             {
261                 cordova.exec( successCB, errorCallback, "AppLog",
"startLogListener", [ ] );
262             }
263             else
264             {
265                 setTimeout( successCB(), 0 );
266             }
267         }
268         catch (ex)
269         {
270             setTimeout( errorCallback( {errorCode : AppLog.ERR_UNKNOWN,
description : ex.message} ), 0 );
271         }
272     }
273
```

```
274      /**
275      * Removes a log listener. This function should be called
276      * with identical parameters that were used
277      *
278      * when adding the log listener with {@link
279      * AppLog.startLogListener}.
280      *
281      * @param
282      * {anonymous.startOrStopLogListenerSuccessCallback} successCB A
283      * callback function that will be invoked
284      * if the log listener is successfully removed.
285      *
286      * @param {anonymous.startOrStopLogListenerErrorCallback}
287      * errorCB A callback function that will be invoked if there
288      * is an error removing the log listener.
289      *
290      * @param {anonymous.logListener} logListener The callback
291      * that was added with {@link AppLog.startLogListener}.
292      *
293      * @param {Object} [containingObject] Object containing the
294      * definition for logListener.
295      *
296      * @public
297      * @memberOf AppLog
298      *
299      * @example
300      * // This example shows how to use this function with a
301      * globally-scoped logListener.
302      * // A global function called by the log listener.
303      * var doSomething = function()
304      * {
305      *     alert("this gets displayed when there is a new log
306      * entry.");
307      * }
308      *
309      * // The log listener callback function that will be
310      * passed to AppLog.startLogListener.
311      * // This function will be invoked whenever there is a new
312      * log entry.
313      * var logListener = function( date, statusCode,
314      * message )
315      * {
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
299     *     doSomething();
300     * }
301     *
302     * function onStartLogListenerSuccessCallback() {
303     *     // Do something here after listener has been added
304     * }
305     *
306     * function onStartLogListenerFailureCallback(error) {
307     *     // React to error here
308     * }
309     *
310     * function onStopLogListenerSuccessCallback() {
311     *     // Do something here after listener has been
removed
312     * }
313     *
314     * function onStopLogListenerFailureCallback(error) {
315     *     // React to error here
316     * }
317     *
318     * // Add the log listener.
319     *
AppLog.startLogListener( onStartLogListenerSuccessCallback,
320     *
onStartLogListenerFailureCallback,
321     *
logListener );
322     *
323     * // At some other point if we want to remove the listener,
we use the following line.
324     *
AppLog.stopLogListener( onStopLogListenerSuccessCallback,
325     *
onStopLogListenerFailureCallback,
326     *
logListener );
```

```
327      *
328      * @example
329      * // This example shows how to use this function with a
logListener contained in an object.
330      * // logListenerManager is an object that will contain the
listener callback as well
331      * // as a function that will be invoked from the listener
callback function.
332      * var logListenerManager = {};
333      *
334      * // This is a function that is called from the listener
callback.
335      * logListenerManager.doSomething = function()
336      * {
337      *     alert("this gets displayed when there is a new log
entry.");
338      * }
339      *
340      * // This is the listener callback that will be passed to
AppLog.startLogListener.
341      * // Since a variable is referenced from the containing
object, the containing object
342      * // will need to be passed to AppLog.startLogListener.
343      * logListenerManager.listener = function( date,
statusCode, message )
344      * {
345      *     this.doSomething();
346      * }
347      *
348      * function onStartLogListenerSuccessCallback() {
349      *     // Do something here after listener has been added
350      * }
351      *
352      * function onStartLogListenerFailureCallback(error) {
353      *     // React to error here
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
354     * }
355     *
356     * function onStopLogListenerSuccessCallback() {
357     *     // Do something here after listener has been
removed
358     * }
359     *
360     * function onStopLogListenerFailureCallback(error) {
361     *     // React to error here
362     * }
363     *
364     * // Pass both the listener callback and the containing
object.
365     *
AppLog.startLogListener( onStartLogListenerSuccessCallback,
366     *
onStartLogListenerFailureCallback,
367     *
logListenerManager.listener,
368     *
logListenerManager );
369     *
370     * // At some other point if we want to remove the listener,
we use the following line.
371     *
AppLog.stopLogListener( onStopLogListenerSuccessCallback,
372     *
onStopLogListenerFailureCallback,
373     *
logListenerManager.listener,
374     *
logListenerManager );
375     */
376     AppLog.stopLogListener = function( successCB, errorCB,
logListener, containingObject ) {
377         try
378         {
379             if ( !logListener || ( typeof logListener !==
"function" ) )
380             {
```



```
381         throw new Error( "AppLog.stopLogListener Error:  
logListener is not a function" );  
382     }  
383  
384     if ( AppLog.logListeners_internal.length > 0 )  
385     {  
386         var foundListener = false;  
387         for ( var i = 0; i <  
AppLog.logListeners_internal.length; i++ )  
388         {  
389             var listener =  
AppLog.logListeners_internal[ i ];  
390             if ( listener.listener === logListener &&  
391                 listener.containingObject ===  
containingObject )  
392             {  
393                 foundListener = true;  
394                 AppLog.logListeners_internal.splice(i,  
1);  
395             }  
396         }  
397  
398         if ( !foundListener )  
399         {  
400             throw new Error( "AppLog.stopLogListener Error:  
logListener was not found. Nothing removed" );  
401         }  
402  
403         if ( AppLog.logListeners_internal.length ===  
0 )  
404         {  
405             cordova.exec( null, null, "AppLog",  
"stopLogListener", [ ] );  
406         }  
407         else
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
408         {
409             setTimeout( successCB(), 0 );
410         }
411     }
412     else
413     {
414         throw new Error( "AppLog.stopLogListener Error:
There are no registered listeners" );
415     }
416 }
417 catch (ex)
418 {
419     setTimeout( errorCallback( {errorCode : AppLog.ERR_UNKNOWN,
description : ex.message} ), 0 ) ;
420 }
421 }
422
423 /**
424  * @private
425  * @param {anonymous.logListener} logListener The callback
to register. This will be invoked when new entries are added to the
log.
426  * @param {Object} [containingObject] Object containing the
definition for logListener. If a log listener callback function
427  */
428 AppLog.logListenerCallBack = function( logListener,
containingObject ) {
429     this.containingObject = containingObject;
430     this.listener = logListener;
431 }
432 AppLog.logListeners_internal = [];
433
434 /**
435  * @private
```

```
436      * @param {AppLog.LogEntry} logEntry Object for each line
in the app log.
437      */
438      AppLog.logListener_internal = function( logEntry ) {
439          if (AppLog.logListeners_internal.length === 0)
440          {
441              return;
442          }
443
444          // The incoming date is number of millisecond, we need to
change it to real JavaScript Date type.
445          var dateInJS = new Date(logEntry.date);
446
447          for (var i = 0; i < AppLog.logListeners_internal.length;
i++)
448          {
449              var logCallBack =
AppLog.logListeners_internal[ i ];
450              var containingObject =
logCallBack.containingObject;
451              var callbackFunction = logCallBack.listener;
452              if (containingObject !== null && containingObject !==
undefined)
453              {
454                  callbackFunction.call(containingObject, dateInJS,
logEntry.statusCode, logEntry.message);
455              }
456              else
457              {
458                  callbackFunction(dateInJS, logEntry.statusCode,
logEntry.message);
459              }
460          }
461      }
462  }) (AppLog, window);
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
463
464  /**
465  * Used to group anonymous objects and callback functions used
  as method parameters. Methods and fields in this
466  * namespace cannot be instantiated. Used for API docs
  generation only.
467  * @namespace
468  */
469  anonymous = (typeof anonymous === "undefined" || !
  anonymous) ? {} : anonymous;
470
471  /**
472  * Callback function that will be invoked with all the entries
  in the app log. There will be one
473  * {@link AppLog.LogEntry} object for each line in the app
  log.
474  * Log entries can be retrieved with {@link
  AppLog.getLogEntries}.
475  *
476  * @name anonymous.getLogEntriesSuccessCallback
477  * @param {AppLog.LogEntry[]} data An array of
  AppLog.LogEntry objects.
478  * @function
479  */
480
481  /**
482  * Callback function that will be invoked when {@link
  AppLog.getLogEntries} fails.
483  *
484  * @name anonymous.getLogEntriesErrorCallback
485  * @param {anonymous.AppLogErrorCallbackParameter} data The
  error object.
486  * @function
487  */
488
489  /**
```

```

490     * Callback function that will be invoked upon successfully
starting a log listener via {@link AppLog.startLogListener},
491     * or upon successfully removing a log listener via {@link
AppLog.stopLogListener}.
492     *
493     * @name anonymous.startOrStopLogListenerSuccessCallback
494     * @function
495     */
496
497     /**
498     * Callback function that will be invoked upon failure to
start a log listener via {@link AppLog.startLogListener},
499     * or upon failure to removing a log listener via {@link
AppLog.stopLogListener}.
500     *
501     * @name anonymous.startOrStopLogListenerErrorCallback
502     * @param {anonymous.AppLogErrorCallbackParameter} data The
error object.
503     * @function
504     */
505
506     /**
507     * Object used in {@link
anonymous.getLogEntriesErrorCallback} and {@link
anonymous.startOrStopLogListenerErrorCallback} functions.
508     *
509     * @class
510     * @name anonymous.AppLogErrorCallbackParameter
511     * @property {number} errorCode Predefined error code
512     * @property {string} description The description of the
error
513     */
514
515     /**

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
516      * Callback function that will be invoked when events are
logged to the app log. Log listeners can be added with {@link
AppLog.startLogListener}.
517      *
518      * @name anonymous.logListener
519      *
520      * @param {Date} date The date of the log entry.
521      * @param {number} event The event ID of the log entry (will
be one of the AppLog status events, or possibly a custom value).
522      * @param {string} message The string carrying the message of
the log entry.
523      * @function
524      */
525
```

Plugins/HttpsProxy/dataajs-https-proxy.js

```
1      // it is depending on Dataajs.js and httpsproxy.js
2      /*
3      * Sybase Hybrid App version 2.3.4
4      * Sybase dataajs integration with PhoneGap HTTPS proxy
5      *
6      * dataajs-https-proxy.js
7      * This file will not be regenerated, so it is possible to
modify it, but it
8      * is not recommended.
9      *
10     * Copyright (c) 2013 Sybase Inc. All rights reserved.
11     */
12
13
14     /**
15     * The namespace for HTTP(S) proxy
16     * @namespace
17     */
```

```

18     httpsConnection = (typeof httpsConnection === "undefined"
|| !httpsConnection) ? {} : httpsConnection;    // 'namespace'
19
20     (function(httpsConnection, window, undefined) {
21
22         /**
23          * Generate an OData HttpClient object over https proxy of
native platform.
24          *
25          * This object will re-direct all odata request to the http
proxy because even with HTTP connection, there are
26          * are some known issue by default setting since the
application in device is cross server accessing the odata service.
27          * See: http://datajs.codeplex.com/discussions/396112 for
details of the issue.
28          *
29          * Call this method normally on HTML page load event to
replace the default odata HTTP client.
30          * @memberOf httpsConnection
31          * @public
32          * @example
33          * // Call datajs api without certificate, users could
call just as normal by passing
34          * // URL as first argument
35          * var length = 0;
36          * var updateUri = server + "/example.svc/
Categories(1)";
37          *
38          * OData.read(server + "/example.svc/Categories",
39          * function (data, response) {
40          *     alert("length " + data.results.length);
41          *     length = data.results.length;
42          *     if ( length > 0 )
43          *     {
44          *         var updateRequest = {

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
45      *           requestUri: updateUri,
46      *           method: "PUT",
47      *           data:
48      *   {
49      *           Picture: new Date().getTime(),
50      *           Description: "Update Record",
51      *           CategoryName: "Updated Category",
52      *           CategoryID: 1
53      *       }
54      *   };
55
56      *   OData.request(updateRequest,
57      *       function (data, response) {
58      *           alert("Response " +
59      *               JSON.stringify(response));
60      *       },
61      *       function (err) {
62      *           alert("Error occurred " +
63      *               err.message);
64      *       }
65      *   ),
66      *   function (err) {
67      *       alert("Error occurred " + err.message);
68      *   });
69
70      * // However, to specify certificate source in the method
71      * // call, users need to pass in
72      * // the request object instead of URL,
73      * // and add the field "certificateSource" to the request
74      * // object.
75      * var length = 0;
```



```
74      * var updateUri = server + "/example.svc/  
Categories(1)";  
75      *  
76      * OData.read({ requestUri: server + "/example.svc/  
Categories", certificateSource : cert},  
77      * function (data, response) {  
78      *     alert("length " + data.results.length);  
79      *     length = data.results.length;  
80      *     if ( length > 0 )  
81      *     {  
82      *         var updateRequest = {  
83      *             requestUri: updateUri,  
84      *             certificateSource : cert,  
85      *             method: "PUT",  
86      *             data:  
87      *             {  
88      *                 Picture: new Date().getTime(),  
89      *                 Description: "Update Record",  
90      *                 CategoryName: "Updated Category",  
91      *                 CategoryID: 1  
92      *             }  
93      *         };  
94      *  
95      *         OData.request(updateRequest,  
96      *             function (data, response) {  
97      *                 alert("Response " +  
JSON.stringify(response));  
98      *             },  
99      *             function (err) {  
100      *                 alert("Error occurred " +  
err.message);  
101      *             }  
102      *         );  
103      *     }  
};
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
104     * },
105     * function (err) {
106     *     alert("Error occurred " + err.message);
107     * });
108     */
109     httpsConnection.generateODataHttpClient = function () {
110         if ( httpsConnection.sendRequest && !
OData.defaultHttpClient.httpsConnectionWrapper) {
111             OData.defaultHttpClient = {
112                 httpsConnectionWrapper: true,
113                 request: function (request, success, error)
{
114                     var url, requestHeaders, requestBody,
statusCode, statusText, responseHeaders;
115                     var responseBody, requestTimeout,
requestUserName, requestPassword, requestCertificate;
116                     var client, result;
117
118                     url = request.requestUri;
119                     requestHeaders = request.headers;
120                     requestBody = request.body;
121
122                     var successCB = function( data ) {
123                         var response = {
124                             requestUri: url,
125                             statusCode: data.status,
126                             statusText: data.statusText,
127                             headers: data.headers,
128                             body: (data.responseText ?
data.responseText : data.responseBase64)
129                         };
130
131                     if (response.statusCode >= 200 &&
response.statusCode <= 299) {
```

```
132         if ( success ) {
133             success(response);
134         }
135     } else {
136         if ( error ) {
137             error({ message: "HTTP request
failed", request: request, response: response });
138         }
139     }
140 };
141
142     var errorCallback = function( data ) {
143         if ( error ) {
144             error({message: data});
145         }
146     };
147
148     if ( request.timeoutMS ) {
149         requestTimeout = request.timeoutMS /
1000;
150     }
151
152     if ( request.certificateSource ) {
153         requestCertificate =
request.certificateSource;
154     }
155
156     if ( request.user ) {
157         requestUserName = request.user;
158     }
159
160     if ( request.password ) {
161         requestPassword = request.password;
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
162         }
163
164         client =
165         httpsConnection.sendRequest(request.method || "GET", url,
166         requestHeaders, requestBody, successCB, errorCallback, requestUserName,
167         requestPassword, requestTimeout, requestCertificate );
168
169
170         result = {};
171         result.abort = function () {
172             client.abort();
173         }
174         if ( error ) {
175             error({ message: "Request
176             aborted" });
177         }
178         };
179         return result;
180     }
181 }
```

Plugins/HttpsProxy/https-proxy.js

```
1  /*
2  * Sybase Hybrid App version 2.3.4
3  * Sybase PhoneGap HTTPS proxy
4  *
5  * https-proxy.js
6  * This file will not be regenerated, so it is possible to
7  * modify it, but it
8  * is not recommended.
9  *
```

```
9      * Copyright (c) 2013 Sybase Inc. All rights reserved.
10     */
11
12     /**
13      * Holds PhoneGap HTTP(S) proxy JavaScript
14      * @namespace
15      */
16     (function (window, undefined) {
17         if (!window.HttpsConnection) {
18             window.HttpsConnection = {};
19         }
20
21         var HttpsConnection = window.HttpsConnection;
22
23         /**
24          * Constant definitions for registration methods
25          */
26
27         /**
28          * Constant indicating the operation failed with unknown
29          * error. Used in {@link anonymous.sendRequestErrorCBParameter}
30          * @type number
31          */
32         HttpsConnection.ERR_UNKNOWN = -1;
33
34         /**
35          * Constant indicating the operation has invalid parameter.
36          * Used in {@link anonymous.sendRequestErrorCBParameter}
37          * @type number
38          */
39         HttpsConnection.ERR_INVALID_PARAMETER_VALUE = -2;
40
41         /**
42          * Constant indicating the operation failed because of
43          * missing parameter. Used in {@link
44          * anonymous.sendRequestErrorCBParameter}
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
39      * @type number
40      */
41      HttpsConnection.ERR_MISSING_PARAMETER = -3;
42      /**
43      * Constant indicating there is no such cordova action for
44      the current service. Used in {@link
45      anonymous.sendRequestErrorCBParameter}
46      * @type number
47      */
48      HttpsConnection.ERR_NO_SUCH_ACTION = -100;
49      /**
50      * Constant indicating certificate from file keystore is
51      not supported on current platform. Used in {@link
52      anonymous.sendRequestErrorCBParameter}
53      * @type number
54      */
55      HttpsConnection.ERR_FILE_CERTIFICATE_SOURCE_UNSUPPORTED = -101;
56      /**
57      * Constant indicating certificate from system keystore is
58      not supported on current platform. Used in {@link
59      anonymous.sendRequestErrorCBParameter}
60      * @type number
61      */
62      HttpsConnection.ERR_SYSTEM_CERTIFICATE_SOURCE_UNSUPPORTED
63      = -102;
64      /**
65      * Constant indicating certificate from Afaria server is
66      not supported on current platform. Used in {@link
67      anonymous.sendRequestErrorCBParameter}
68      * @type number
69      */
70      HttpsConnection.ERR_AFARIA_CERTIFICATE_SOURCE_UNSUPPORTED
71      = -103;
72      /**
```

```

63      * Constant indicating the certificate with given alias
could not be found. Used in {@link
anonymous.sendRequestErrorCBParameter}

64      * @type number

65      */

66      HttpsConnection.ERR_CERTIFICATE_ALIAS_NOT_FOUND = -104;

67      /**

68      * Constant indicating the certificate file could not be
found. Used in {@link anonymous.sendRequestErrorCBParameter}

69      * @type number

70      */

71      HttpsConnection.ERR_CERTIFICATE_FILE_NOT_EXIST = -105;

72      /**

73      * Constant indicating incorrect certificate file format.
Used in {@link anonymous.sendRequestErrorCBParameter}

74      * @type number

75      */

76      HttpsConnection.ERR_CERTIFICATE_INVALID_FILE_FORMAT =
-106;

77      /**

78      * Constant indicating failed in getting certificate. Used
in {@link anonymous.sendRequestErrorCBParameter}

79      * @type number

80      */

81      HttpsConnection.ERR_GET_CERTIFICATE_FAILED = -107;

82      /**

83      * Constant indicating the provided certificate failed
validation on server side. Used in {@link
anonymous.sendRequestErrorCBParameter}

84      * @type number

85      */

86      HttpsConnection.ERR_CLIENT_CERTIFICATE_VALIDATION =
-108;

87      /**

88      * Constant indicating the server certificate failed
validation on client side. Used in {@link
anonymous.sendRequestErrorCB}

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
89      * @type number
90      */
91      HttpsConnection.ERR_SERVER_CERTIFICATE_VALIDATION =
-109;
92      /**
93      * Constant indicating the server request failed. Used in
{@link anonymous.sendRequestErrorCB}
94      * @type number
95      */
96      HttpsConnection.ERR_SERVER_REQUEST_FAILED = -110;
97      /**
98      * Constant indicating timeout error while connecting to
the server. Used in {@link anonymous.sendRequestErrorCB}
99      * @type number
100     */
101     HttpsConnection.ERR_HTTP_TIMEOUT = -120;
102
103     /**
104     * Create certificate source description object for
certificates from a keystore file.
105     * <b> Not supported on Blackberry platform </b>
106     * @class
107     * @memberOf HttpsConnection
108     * @public
109     * @param {string} Path Path of the keystore file. For iOS
client, it first tries to load the
110     *         relative file path from application's
Documents folder; if it fails, then tries
111     *         to load the file path from application's
main bundle. In addition, before trying
112     *         to load the certificate from file system,
iOS client first checks whether the
113     *         specified certificate key already exists
in the key store, if so, it just loads
114     *         the existing certificate from key store,
instead of loading the certificate from
```



```

115         *                               file system.
116         * @param {string} Password Password of the keystore.
117         * @param {string} CertificateKey An unique key that will
118         * be used to locate the certificate.
119         */
119         HttpsConnection.CertificateFromFile = function (Path,
120         Password, CertificateKey) {
121             this.Source = "FILE";
122             this.Path = Path;
123             this.Password = Password;
124             this.CertificateKey = CertificateKey;
125         };
126     /**
127     * Create certificate source description object for
128     * certificates from Afaria.
129     * @class
130     * @memberOf HttpsConnection
131     * @public
132     * @param {string} CN Common Name (CN) for CA/SCEP
133     * protocol. For iOS, the retrieved certificate is
134     * stored in the key store with the common
135     * name as the certificate key, the
136     * following requests for the same common
137     * name will just load the saved certificate
138     * from key store, instead of sending a
139     * new request to Afaria server.
140     * @param {string} [ChallengeCode] Challenge code for CA/
141     * SCEP protocol.
142     */
143     HttpsConnection.CertificateFromAfaria = function (CN,
144     ChallengeCode) {
145         this.Source = "AFARIA";
146         this.CN = CN;
147         this.ChallengeCode = ChallengeCode;
148     };

```

```
142
143     /**
144         * Create certificate source description object for
certificates from system keystore (Keystore in BB, Keychain in iOS
and Android).
145         * The certificateKey is not used on the BB platform. BB
will prompt the user to select a certificate if a certificate was not
already
146         * used for the server connection.
147         * @class
148         * @memberOf HttpsConnection
149         * @public
150         * @param {string} CertificateKey An unique key that will
be used to locate the certificate. Not used in BB platform.
151         */
152         HttpsConnection.CertificateFromStore = function
(CertificateKey) {
153             this.Source = "SYSTEM";
154             this.CertificateKey = CertificateKey;
155         };
156
157         HttpsConnection.MSG_MISSING_PARAMETER = "Missing a
required parameter: ";
158         HttpsConnection.MSG_INVALID_PARAMETER_VALUE = "Invalid
Parameter Value for parameter: ";
159
160     /**
161         * @private
162         * @param {Object} [certSource] Certificate description
object. It can be one of {@link
HttpsConnection.CertificateFromFile},
163         * {@link HttpsConnection.CertificateFromStore}, or {@link
HttpsConnection.CertificateFromAfaria}.
164         * @param {anonymous.sendRequestErrorCB} errorCallback Callback
method upon failure.
165         */
```

```
166     httpsConnection.validateCertSource = function(certSource,
errorCB) {
167         if (!certSource) {
168             // The certificate is not present, so just ignore
it.
169             return true;
170         }
171
172         // errorCB required.
173         // First check this one. We may need it to return
errors
174         if (errorCB && (typeof errorCB !== "function")) {
175             console.log("HttpsConnection Error: errorCB is not
a function");
176             return false;
177         }
178
179         try {
180             // First check whether it is an object
181             if (typeof certSource !== "object") {
182                 errorCB({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_INVALID_PARAMETER_VALUE + "certSource"});
183                 return false;
184             }
185
186             if (certSource.Source === "FILE") {
187                 if (!certSource.Path) {
188                     errorCB({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_MISSING_PARAMETER + "keystore path"});
189                     return false;
190                 }
191
192                 if (typeof certSource.Path !== "string") {
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
193         errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_INVALID_PARAMETER_VALUE + "keystore path"});
194         return false;
195     }
196
197     if (!certSource.Password) {
198         errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_MISSING_PARAMETER + "keystore password"});
199         return false;
200     }
201
202     if (typeof certSource.Password !== "string")
{
203         errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_INVALID_PARAMETER_VALUE + "keystore
password"});
204         return false;
205     }
206
207     if (!certSource.CertificateKey) {
208         errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_MISSING_PARAMETER + "certificate key"});
209         return false;
210     }
211
212     if (typeof certSource.CertificateKey !==
"string") {
213         errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_INVALID_PARAMETER_VALUE + "certificate key"});
214         return false;
215     }
216     } else if (certSource.Source === "SYSTEM") {
```

```
217         if (!certSource.CertificateKey) {
218             errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_MISSING_PARAMETER + "certificate key"});
219             return false;
220         }
221
222         if (typeof certSource.CertificateKey !==
"string") {
223             errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_INVALID_PARAMETER_VALUE + "certificate key"});
224             return false;
225         }
226     } else if (certSource.Source === "AFARIA") {
227         if (!certSource.CN) {
228             errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_MISSING_PARAMETER + "common name"});
229             return false;
230         }
231
232         if (typeof certSource.CN !== "string") {
233             errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_INVALID_PARAMETER_VALUE + "common name"});
234             return false;
235         }
236
237         if (!certSource.ChallengeCode) {
238             errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_MISSING_PARAMETER + "Afaría challenge code"});
239             return false;
240         }
241     }
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
242         if (typeof certSource.ChallengeCode !==
"string") {
243             errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_INVALID_PARAMETER_VALUE + "Afaria challenge
code"});
244             return false;
245         }
246     } else {
247         errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_INVALID_PARAMETER_VALUE + "certSource"});
248         return false;
249     }
250
251     return true;
252 } catch (ex) {
253     errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_INVALID_PARAMETER_VALUE + "certSource"});
254 }
255 };
256
257 /**
258  * Send a HTTP(S) request to a remote server.
259  * @memberOf HttpsConnection
260  * @public
261  * @param {string} method Standard HTTP request method
name.
262  * @param {string} url The http url with format http(s)://
[user:password]@hostname[:port]/path.
263  * @param {Object} header HTTP header to be sent to server.
This is an Object. Can be null.
264  * @param {string} requestBody Data to be sent to server
with the request. Itâ€™s a string value. Can be null.
265  * @param {anonymous.sendRequestSuccessCB} successCB
Callback method upon success.
```

```

266      * @param {anonymous.sendRequestErrorCB} errorCallback Callback
method upon failure.

267      * @param {string} [user] User ID for basic
authentication.

268      * @param {string} [password] User password for basic
authentication.

269      * @param {number} [timeout] Timeout setting in
seconds.

270      * @param {Object} [certSource] Certificate description
object. It can be one of {@link
HttpsConnection.CertificateFromFile},

271      * {@link HttpsConnection.CertificateFromStore}, or {@link
HttpsConnection.CertificateFromAfaria}.

272      * @returns {anonymous.abort} A JavaScript function object
to cancel the operation.

273      * @example

274      * // To send a post request to server, call the method

275      * HttpsConnection.sendRequest("POST", "http://
www.google.com", null, "THIS IS THE BODY", function (data) {

276      *     alert("Status: " +
JSON.stringify(data.status));

277      *     alert("Headers: " +
JSON.stringify(data.headers));

278      *     alert("Response: " +
JSON.stringify(data.response));

279      *     }, function (data) {

280      *     alert("Failed: " + JSON.stringify(data));});

281      * // To send a post request to server with headers, call
the method

282      * HttpsConnection.sendRequest("POST", url, {HeaderName :
"Header value"}, "THIS IS THE BODY", successCB, errorCallback);

283      * // To send a post request to server with basic
authentication, call the method

284      * HttpsConnection.sendRequest("POST", url, headers, "THIS
IS THE BODY", successCB, errorCallback, "username", "password");

285      * // To send a post request to server with mutual
authentication, call the method

286      * HttpsConnection.sendRequest("POST", "https://
hostname", headers, "THIS IS THE BODY", successCB, errorCallback, null,

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
287         *      null, 0, new CertificateFromFile("/mnt/sdcard/
my.keystore", "password", "mykey"));
288         */
289         HttpsConnection.sendRequest = function (method, url,
header, requestBody, successCB, errorCallback, user, password, timeout,
certSource){
290
291             // errorCallback required.
292             // First check this one. We may need it to return
errors
293             if (!errorCB || (typeof errorCallback !== "function")) {
294                 console.log("HttpsConnection Error: errorCallback is not
a function");
295                 // if error callback is invalid, throw an exception
to notify the caller
296                 throw new Error("HttpsConnection Error: errorCallback is
not a function");
297                 return;
298             }
299
300             // method required
301             if (!method) {
302                 console.log("HttpsConnection Error: method is
required");
303                 errorCallback({errorCode :
HttpsConnection.ERR_MISSING_PARAMETER, description :
HttpsConnection.MSG_MISSING_PARAMETER + "method"});
304                 return;
305             }
306
307             // We only support GET, POST, HEAD, PUT, DELETE
method
308             if (method !== "GET" && method !== "POST" && method !==
"HEAD" && method !== "PUT" && method !== "DELETE") {
309                 console.log("Invalid Parameter Value for parameter:
" + method);
```



```
310         errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_INVALID_PARAMETER_VALUE + "method"});
311         return;
312     }
313
314     // url required
315     if (!url) {
316         console.log("HttpsConnection Error: url is
required");
317         errorCallback({errorCode :
HttpsConnection.ERR_MISSING_PARAMETER, description :
HttpsConnection.MSG_MISSING_PARAMETER + "url"});
318         return;
319     }
320
321     // successCB required
322     if (!successCB) {
323         console.log("HttpsConnection Error: successCB is
required");
324         errorCallback({errorCode :
HttpsConnection.ERR_MISSING_PARAMETER, description :
HttpsConnection.MSG_MISSING_PARAMETER + "successCB"});
325         return;
326     }
327
328     if (typeof successCB !== "function") {
329         console.log("HttpsConnection Error: successCB is
not a function");
330         errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_INVALID_PARAMETER_VALUE + "successCB"});
331         return;
332     }
333
334     if (user && typeof user !== "string") {
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
335         errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_INVALID_PARAMETER_VALUE + "user"});
336         return;
337     }
338
339     if (password && typeof password !== "string") {
340         errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_INVALID_PARAMETER_VALUE + "password"});
341         return;
342     }
343
344     if (timeout && typeof timeout !== "number") {
345         errorCallback({errorCode :
HttpsConnection.ERR_INVALID_PARAMETER_VALUE, description :
HttpsConnection.MSG_INVALID_PARAMETER_VALUE + "timeout"});
346         return;
347     }
348
349     if (!HttpsConnection.validateCertSource(certSource,
errorCB)) {
350         return;
351     }
352
353     try {
354         var client = new HttpsConnection.Client(method,
url, header, requestBody, successCB, errorCallback, user, password,
timeout, certSource);
355         return client.send();
356     } catch (ex){
357         errorCallback({errorCode : HttpsConnection.ERR_UNKNOWN,
description : ex.message});
358     }
359 };
360
```

```

361         /**
362         * Send a HTTP(S) GET request to a remote server.
363         * @memberOf HttpsConnection
364         * @public
365         * @param {string} url The http url with format http(s)://
[user:password]@hostname[:port]/path.
366         * @param {Object} header HTTP header to be sent to server.
This is an Object. Can be null.
367         * @param {anonymous.sendRequestSuccessCB} successCB
Callback method upon success.
368         * @param {anonymous.sendRequestErrorCB} [errorCB]
Callback method upon failure.
369         * @param {string} [user] User ID for basic
authentication.
370         * @param {string} [password] User password for basic
authentication.
371         * @param {number} [timeout] Timeout setting in
seconds.
372         * @param {Object} [certSource] Certificate description
object. It can be one of {@link
HttpsConnection.CertificateFromFile},
373         * {@link HttpsConnection.CertificateFromStore}, or {@link
HttpsConnection.CertificateFromAfararia}.
374         * @returns {anonymous.abort} A JavaScript function object
to cancel the operation.
375         * @example
376         * // To send a get request to server, call the method
377         * HttpsConnection.get("http://www.google.com", null,
function (data) {
378             *             alert("Status: " +
JSON.stringify(data.status));
379             *             alert("Headers: " +
JSON.stringify(data.headers));
380             *             if (data.responseText){
381             *                 alert("Response: " +
JSON.stringify(data.responseText));
382             *             }
383             *             },

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
384         *         function (error) {
385             *             alert("Failed: " +
JSON.stringify(error));
386         *         }};

387         * // To send a get request to server with headers, call
the method

388         * HttpsConnection.get(url, {HeaderName : "Header value"},
successCB, errorCallback);

389         * // To send a get request to server with basic
authentication, call the method

390         * HttpsConnection.get(url, headers, successCB, errorCallback,
"username", "password");

391         * // To send a get request to server with mutual
authentication, call the method

392         * HttpsConnection.get("https://hostname", headers,
successCB, errorCallback, null, null, 0,

393         *         new CertificateFromFile("/mnt/sdcard/my.p12",
"password", "mykey"));

394         */

395         HttpsConnection.get = function (url, header, successCB,
errorCB, user, password, timeout, certSource){

396             return HttpsConnection.sendRequest("GET", url, header,
null, successCB, errorCallback, user, password, timeout, certSource);

397         };

398

399         /**

400         * Delete cached certificate from keychain. iOS client
will always try the cached certificate first if it is available
before requesting the certificate from

401         * afaria server or loading the certificate from file
system. In case the cached certificate is no longer valid, use this
method to delete it from keychain

402         * <b> Only supported by iOS platform </b>

403         * @memberOf HttpsConnection

404         * @public

405         * @param {anonymous.sendRequestSuccessCB} successCB
Callback method upon success.

406         * @param {anonymous.sendRequestErrorCB} [errorCB]
Callback method upon failure.
```

```

407      * @param {string} certificateKey The key of the
certificate to be deleted.
408      */
409      HttpsConnection.deleteCertificateFromStore = function
(successCB, errorCallback, certificateKey) {
410          cordova.exec(successCB, errorCallback, "HttpsProxy",
"deleteCertificateFromStore", [certificateKey]);
411      };
412
413
414      /**
415      * @private
416      * @param {string} method Standard HTTP request method
name.
417      * @param {string} url The http url with format http(s)://
[user:password]@hostname[:port]/path.
418      * @param {Object} header HTTP header to be sent to server.
This is an Object. Can be null.
419      * @param {string} requestBody Data to be sent to server
with the request. Itâ€™s a string value. Can be null.
420      * @param {anonymous.sendRequestSuccessCB} successCB
Callback method upon success.
421      * @param {anonymous.sendRequestErrorCB} errorCallback Callback
method upon failure.
422      * @param {string} [user] User ID for basic
authentication.
423      * @param {string} [password] User password for basic
authentication.
424      * @param {number} [timeout] Timeout setting in
seconds.
425      * @param {Object} [certSource] Certificate description
object. It can be one of {@link
HttpsConnection.CertificateFromFile},
426      * {@link HttpsConnection.CertificateFromStore}, or {@link
HttpsConnection.CertificateFromAfaria}.
427      * @returns {anonymous.abort} A JavaScript function object
to cancel the operation.
428      */

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
429     httpsConnection.Client = function ( method, url, header,
requestBody, successCB, errorCallback, user, password, timeout,
certSource )
430     {
431         //ios plugin parameter does not support object type,
convert Header and CertSource to JSON string
432         if (device.platform === "iOS" || (device.platform &&
device.platform.indexOf("iP") === 0 ) )
433         {
434             if (header) {
435                 header = JSON.stringify(header);
436             }
437             if (certSource) {
438                 certSource = JSON.stringify(certSource);
439             }
440         }
441
442         this.Method = method;
443         this.Url = url;
444         this.Header = header;
445         this.RequestBody = requestBody;
446         this.SuccessCB = successCB;
447         this.ErrorCB = errorCallback;
448         this.User = user;
449         this.Password = password;
450         this.Timeout = timeout;
451         this.CertSource = certSource;
452         this.IsAbort = false;
453
454         this.abort = function ()
455         {
456             this.IsAbort = true;
457         };
458     }
```

```
459         this.send = function ()
460         {
461             var args = [this.Method, this.Url, this.Header,
this.RequestBody, this.User, this.Password, this.Timeout,
this.CertSource];
462
463             var me = this;
464
465             var successCallBack = function(data)
466             {
467                 if (me.IsAbort === true)
468                 {
469                     return;
470                 }
471
472                 successCB(data);
473             };
474
475             var errorCallback = function(data)
476             {
477                 if (me.IsAbort === true)
478                 {
479                     return;
480                 }
481
482                 errorCallback(data);
483             };
484
485             cordova.exec(successCallBack, errorCallback,
"HttpsProxy", "sendRequest", args);
486
487             return this.abort;
488         };
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
489     };
490
491     })(this);
492
493     /**
494     * Used to group anonymous objects and callback functions used
495     * as method parameters. Methods and fields in this
496     * namespace cannot be instantiated. Used for API docs
497     * generation only.
498     * @namespace
499     */
500     anonymous = (typeof anonymous === "undefined" || !
501     anonymous) ? {} : anonymous; // SUP 'namespace'
502
503     /**
504     * Callback function that will be invoked
505     * HttpsConnection.get()/sendRequest() succeeded.
506     * @name anonymous.sendRequestSuccessCB
507     * @param {anonymous.sendRequestSuccessCBParameter} data The
508     * response data object.
509     * @function
510     */
511
512     /**
513     * Callback function that will be invoked
514     * HttpsConnection.get()/sendRequest() failed.
515     * @name anonymous.sendRequestErrorCB
516     * @param {anonymous.sendRequestErrorCBParameter} data The
517     * error object.
518     * @function
519     */
```



```

517     /**
518     * Object used in {@link anonymous.sendRequestSuccessCB}
function.
519     * @class
520     * @name anonymous.sendRequestSuccessCBParameter
521     * @property {number} status The HTTP status code
522     * @property {object} headers An object that contains
headerKey = value pairs.
523     * @property {string} [responseText] The text response. This
parameter is present only if the response is a text response.
524     * @property {string} [responseBase64] Base64 encoded
representation of the binary response. This parameter is included
only
525     *                                     if the response is a binary
response.
526     * @property {object} [clientError] An optional object that
contains the authentication error. It is an object of {@link
anonymous.sendRequestErrorCBParameter}.
527     */
528
529     /**
530     * Object used in {@link anonymous.sendRequestErrorCB}
function.
531     * @class
532     * @name anonymous.sendRequestErrorCBParameter
533     * @property {number} errorCode Predefined error code
534     * @property {string} description The description of the
error
535     * @property {number} [nativeErrorCode] The native error code
reported from Afaria, device, etc (optional)
536     */
537
538
539     /**
540     * JavaScript function to abort the HTTP(S) request
541     *
542     * @name anonymous.abort

```

```
543      *
544      * @function
545      */
546
```

SUPStorage.js

```
1      /*
2      * Sybase Hybrid App version 2.3.4
3      *
4      * SUPStorage.js
5      * This file will not be regenerated, so it is possible to
modify it, but it
6      * is not recommended.
7      *
8      * Copyright (c) 2012 Sybase Inc. All rights reserved.
9      */
10
11     /**
12     * The namespace for the Hybrid Web Container javascript
13     * @namespace
14     */
15     hwc = (typeof hwc === "undefined" || !hwc) ? {} :
hwc;
// SUP 'namespace'
16
17
18     /**
19     * Access the storage functions, which allow you to specify a
cache that stores results from online requests.
20     *
21     * These functions give you the ability to:
22     * Name the cached result sets
23     * Enumerate the cached result sets
24     * Read, delete, and modify cached contents individually for
each cached result set
```

```

25     * Cached result sets must be stored as strings (before
deserialization to an xmlWorkflowMessage structure).
26     */
27     (function(hwc, window, undefined) {
28
29     /**
30     * Creates a SUPStorage with the specified storeName. Provides
encrypted storage of name value pairs. Results from online requests
are one example.
31     * Strings stored in SUPStorage are encrypted and persisted to
survive multiple invocations of the mobile workflow application.
32     * @desc Storage
33     * @memberOf hwc
34     * @constructor
35     * @param {string} store the store name
36     *
37     * @example
38     * var store1 = new hwc.SUPStorage("one");
39     */
40     hwc.SUPStorage = function(store) {
41         this.bForSharedStorage = false;
42         this.store = store ? store : "";
43     };
44
45     /**
46     * Gets the number of available keys in this object. The keys
themselves may be
47     * retrieved using key().
48     * @desc Storage
49     * @memberOf hwc.SUPStorage
50     * @public
51     * @example
52     * // Create the SUP Storage
53     * var store = new hwc.SUPStorage ("one");

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
54     * store.setItem ("foo", "bar"); // add an item.
55     * store.setItem ("foo1", "bar"); // add an item.
56     * store.setItem ("foo2", "bar"); // add an item.
57     * var result = store.length; // result = 3
58     */
59     hwc.SUPStorage.prototype.length = function() {
60         var response;
61         hwc.traceEnteringMethod("hwc.SUPStorage.length");
62         try {
63             if (hwc.isWindowsMobile() || hwc.isIOS()) {
64                 response =
hwc.getDataFromContainer("workflowstorage",
"&command=length&shared=" + this.bForSharedStorage +
65                 "&store=" +
encodeURIComponent(this.store));
66                 return parseInt(response, 10);
67             }
68             else {
69                 if (this.bForSharedStorage) {
70                     return
_SharedStorage.length(hwc.versionURLParam);
71                 }
72                 else {
73                     return SUPStorage.length(this.store);
74                 }
75             }
76         } finally {
77             hwc.traceLeavingMethod("hwc.SUPStorage.length");
78         }
79     };
80
81     /**
82     * Returns the key at the supplied index. Keys are guaranteed
to remain
```

```

83     * at the same index until a modification is made.
84     *
85     * @desc Storage
86     * @public
87     * @memberOf hwc.SUPStorage
88     * @param {Integer} index 0-based index to the key. Must be
less than the value retrieved
89     *     by .length.
90     * @returns {string} The key, or null if the index is
invalid.
91     * @example
92     * // Create the SUP Storage
93     * var store = new hwc.SUPStorage ("one");
94     * store.setItem ("foo", "bar"); // add an item.
95     * var result = store.key (0); // will returns "foo".
96     */
97     hwc.SUPStorage.prototype.key = function(index) {
98         var key, isExist;
99         hwc.traceEnteringMethod("hwc.SUPStorage.key");
100        try {
101            if (null === index) {
102                return null;
103            }
104
105            if (hwc.isWindowsMobile() || hwc.isIOS()) {
106                key = hwc.getDataFromContainer("workflowstorage",
"&command=key&shared=" + this.bForSharedStorage +
107                    "&store=" + encodeURIComponent(this.store) +
"&index=" + encodeURIComponent(index));
108
109                if (key === null || typeof key === 'undefined' ||
key === "") {
110                    isExist =
hwc.getDataFromContainer("workflowstorage",
"&command=exist&shared=" + this.bForSharedStorage +

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
111         "&store=" + encodeURIComponent(this.store) +
112         "&index=" + encodeURIComponent(index));
113
114         //WM returns empty string if an item does not
115         //exist or if the value is empty string
116         //call exist to distinguish this
117         if (isExist == "true") {
118             key = "";
119         }
120         else {
121             key = null;
122         }
123     } else {
124         if (this.bForSharedStorage) {
125             key = _SharedStorage.key(index,
126             hwc.versionURLParam);
127         }
128         else {
129             key = SUPStorage.key(this.store, index);
130         }
131     }
132     if (key === null || typeof key === 'undefined') {
133         return null;
134     } else {
135         return key + "";
136     }
137 } finally {
138     hwc.traceLeavingMethod("hwc.SUPStorage.key");
139 }
140 };
```

```
141
142  /**
143   * Helper method for parameter validation
144   * @private
145   * @param {string} input: input value .
146   * @returns {string} if input is null, return empty string
147   */
148   function checkNull(input) {
149       if (null === input) {
150           input = "";
151       }
152       return input;
153   }
154
155  /**
156   * Retrieves the value associated with a specified key.
157   *
158   * @desc Storage
159   * @memberOf hwc.SUPStorage
160   * @param {string} key String key corresponding to the
161   * requested value.
162   * @returns {string} A String value corresponding to the key,
163   * or null if either the key
164   * is not known, or if the key exists but its value was set
165   * to null.
166   * @example
167   * // Create the SUP Storage
168   * var store = new hwc.SUPStorage ("one");
169   * store.setItem ("foo", "bar"); // add an item.
170   * result = store.getItem ("foo"); // will returns "bar".
171   * result = store.getItem ("fool"); // fool does not exists;
172   * will return null.
173   */
174   hwc.SUPStorage.prototype.getItem = function(key) {
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
171         var value, isExist;
172         key = key ? key : "";
173
174         hwc.traceEnteringMethod("hwc.SUPStorage.getItem");
175         try {
176             if (hwc.isWindowsMobile() || hwc.isIOS()) {
177                 value =
178                 hwc.getDataFromContainer("workflowstorage",
179                 "&command=getItem&shared=" + this.bForSharedStorage +
180                 "&store=" + encodeURIComponent(this.store) +
181                 "&key=" + encodeURIComponent(key));
182
183
184                 if (value === null || typeof value === 'undefined'
185                 || value === "") {
186                     isExist =
187                     hwc.getDataFromContainer("workflowstorage",
188                     "&command=exist&shared=" + this.bForSharedStorage +
189                     "&store=" + encodeURIComponent(this.store) +
190                     "&key=" + encodeURIComponent(key));
191
192
193                     //WM returns empty string if an item does not
194                     exist or if the value is empty string
195                     //call exist to distinguish this
196                     if (isExist == "true") {
197                         value = "";
198                     }
199                     else {
200                         value = null;
201                     }
202                 }
203             }
204         }
205         else {
206             if (this.bForSharedStorage) {
207                 value = _SharedStorage.getItem(key,
208                 hwc.versionURLParam);
209             }
210         }
211     }
212 }
```



```
198         else {
199             value = SUPStorage.getItem(this.store,
200 key);
201         }
202     }
203     if (value === null || typeof value === 'undefined')
204     {
205         return null;
206     } else {
207         return value + "";
208     } finally {
209         hwc.traceLeavingMethod("hwc.SUPStorage.getItem");
210     }
211 };
212
213 /**
214  * A constant for the maximum length for a string being stored
215  on BB7
216  * BB7 cannot handle strings with length longer than 524000
217  * This restriction applies to real devices as well as
218  simulators.
219  */
220 hwc.SUPStorage.BB7_MAX_STRING_STORAGE_LENGTH = 524000;
221
222 /**
223  * Sets the value associated with a specified key. This
224  replaces the key's
225  * previous value, if any.
226  *
227  * @desc Storage
228  * @memberOf hwc.SUPStorage
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
226     * @param {string} key String key corresponding to the
value.
227     * @param {string} value String value to store.
228     * @example
229     * // Create the SUP Storage
230     * var store = new hwc.SUPStorage ("one");
231     * store.setItem ("foo", "bar"); // add an item.
232     */
233     hwc.SUPStorage.prototype.setItem = function(key, value) {
234         var result;
235         hwc.traceEnteringMethod("hwc.SUPStorage.setItem");
236         key = key ? key : "";
237         value = value ? value : "";
238         try {
239             if (hwc.isWindowsMobile() || hwc.isIOS()) {
240                 hwc.postDataToContainer("workflowstorage",
"command=setItem&store=" + encodeURIComponent(this.store) +
"&shared=" + this.bForSharedStorage + "&key=" +
241                     encodeURIComponent(key) + "&value=" +
encodeURIComponent(value));
242             }
243             else {
244                 if (hwc.isBlackBerry7() && value.length >
hwc.SUPStorage.BB7_MAX_STRING_STORAGE_LENGTH) {
245                     throw new
hwc.SUPStorageException(hwc.SUPStorageException.MAX_SIZE_REACHED,
"SUP storage maximum size reached - maximum length of string to store
on BB7 is 524000 but attempted to store string of length " +
value.length);
246                 }
247                 if (this.bForSharedStorage) {
248                     result = _SharedStorage.setItem(key, value,
hwc.versionURLParam);
249                 }
250             else {
251                 result = SUPStorage.setItem(this.store, key,
value);
```

```

252         }
253         if (result !== 0) {
254             throw new hwc.SUPStorageException(result, "SUP
storage maximum size reached");
255         }
256     }
257     } finally {
258         hwc.traceLeavingMethod("hwc.SUPStorage.setItem");
259     }
260 };
261
262 /**
263  * Removes the key and its associated value from this object.
If the
264  * key does not exist, has no effect.
265  *
266  * @desc Storage
267  * @memberOf hwc.SUPStorage
268  * @param {string} key String key to remove.
269  * @example
270  * // Create the SUP Storage
271  * var store = new hwc.SUPStorage ("one");
272  * store.setItem ("foo", "bar"); // add an item.
273  * store.removeItem ("foo");
274  * result = store.getItem ("food"); // will be null.
275  */
276     hwc.SUPStorage.prototype.removeItem = function(key) {
277         hwc.traceEnteringMethod("hwc.SUPStorage.removeItem");
278         try {
279             key = key ? key : "";
280             if (hwc.isWindowsMobile() || hwc.isIOS()) {
281                 hwc.getDataFromContainer("workflowstorage",
"&command=removeItem&shared=" + this.bForSharedStorage +

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
282         "&store=" + encodeURIComponent(this.store) +
"&key=" + encodeURIComponent(key));
283     }
284     else {
285         if (this.bForSharedStorage) {
286             _SharedStorage.removeItem(key,
hwc.versionURLParam);
287         }
288         else {
289             SUPStorage.removeItem(this.store, key);
290         }
291     }
292 } finally {
293     hwc.traceLeavingMethod("hwc.SUPStorage.removeItem");
294 }
295 };
296
297 /**
298  * Removes all key/value pairs from this object.
299  * @desc Storage
300  * @memberOf hwc.SUPStorage
301  */
302 hwc.SUPStorage.prototype.clear = function() {
303     hwc.traceEnteringMethod("hwc.SUPStorage.clear");
304     try {
305         if (hwc.isWindowsMobile() || hwc.isIOS()) {
306             hwc.getDataFromContainer("workflowstorage",
"&command=clear&shared=" + this.bForSharedStorage +
307                 "&store=" +
encodeURIComponent(this.store));
308         }
309         else {
310             if (this.bForSharedStorage) {
```

```
311         _SharedStorage.clear(hwc.versionURLParam);
312     }
313     else {
314         SUPStorage.clear(this.store);
315     }
316 }
317 } finally {
318     hwc.traceLeavingMethod("hwc.SUPStorage.clear");
319 }
320 };
321
322 /**
323  * Exception thrown when Storage space is exceeded.
324  * @desc Storage
325  * @constructor
326  * @memberOf hwc
327  * @param {Integer} code the error code
328  * @param {string} message the error message.
329  */
330 hwc.SUPStorageException = function(code, message) {
331     this.code = code;
332     this.message = message;
333 };
334
335 hwc.SUPStorageException.UNKNOWN_ERROR = 1;
336 hwc.SUPStorageException.MAX_SIZE_REACHED = 2;
337 hwc.SUPStorageException.SHARED_STORAGE_DISABLED = 3;
338
339 // shared storage key.
340 hwc.sharedStorageKey = undefined;
341
342 /**
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
343     * Method to return the shared storage key defined for the
hybrid app by designer. An empty string is returned if the shared
storage function is disabled.
344     * @desc Storage
345     * @memberOf hwc
346     * @returns {string} the shared storage key.
347     */
348     hwc.getSharedStorageKey = function() {
349         if (hwc.sharedStorageKey === undefined ) {
350             var key =
hwc.getQueryVariable("sharedStorageKey");
351             hwc.sharedStorageKey = (key === undefined) ?
"":key;
352         }
353         return hwc.sharedStorageKey;
354     };
355
356     /**
357     * Indicates whether the shared storage is enabled for the
hybrid app.
358     * @desc Storage
359     * @memberOf hwc
360     * @returns {boolean} true if the shared storage is enabled;
false otherwise.
361     */
362     hwc.isSharedStorageEnabled = function() {
363         var key = hwc.getSharedStorageKey();
364         if (key === undefined || key === "") {
365             return false;
366         }
367         else {
368             return true;
369         }
370     };
```

```

371
372     /**
373     * Constructs a new SUP shared storage. You can use the
374     * returned value to access the shared storage data with the existing
375     * SUPStorage interface,
376     * however, the operation only affects the items belonging to
377     * the specified shared storage key.
378     * @classdesc
379     * @memberOf hwc
380     * @desc Storage
381     */
382     hwc.SharedStorage = function() {
383         if (hwc.isSharedStorageEnabled() === false ) {
384             throw new
385             hwc.SUPStorageException(hwc.SUPStorageException.SHARED_STORAGE_DISA
386             BLED, "Shared storage is disabled");
387         }
388         this.bForSharedStorage = true;
389         this.store = "";
390     };
391
392     hwc.SharedStorage.prototype = new hwc.SUPStorage();
393     hwc.SharedStorage.constructor = hwc.SharedStorage;
394 } (hwc, window);

```

Timezone.js

```

1     /**
2     * Sybase Hybrid App version 2.3.4
3     *
4     * Timezone.js

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
5      * This file will not be regenerated, so it is possible to
modify it, but it
6      * is not recommended.
7      *
8      * Copyright (c) 2012 Sybase Inc. All rights reserved.
9      */
10
11     /**
12     * The namespace for the Hybrid Web Container javascript
13     * @namespace
14     */
15     hwc = (typeof hwc === "undefined" || !hwc) ? {} : hwc;    //
SUP 'namespace'
16
17     (function(hwc, window, undefined) {
18
19         /**
20         * Returns the current locale. The platform's locale string
should be available. However, if it is
21         * missing the function queries available JavaScript APIs for
a suitable value.
22         * @desc Timezone
23         * @memberOf hwc
24         * @public
25         * @returns {string} Returns a string containing the current
locale, or null if it is not available.
26         * @example
27         * var sLocale = hwc.getCurrentLocale();
28         *
29         */
30         hwc.getCurrentLocale = function() {
31             hwc.traceEnteringMethod("hwc.getCurrentLocale");
32
33             try {
```



```
34         if(hwc.lang) {
35             return hwc.lang;
36         }
37         else {
38             if ( navigator ) {
39                 if ( navigator.language ) {
40                     if (hwc.isAndroid()) {
41                         return navigator.userAgent.match(/
Android \d+(?:\.\d+){1,2}; [a-z]{2}-[a-z]{2}/).toString().match(/[a-
z]{2}-[a-z]{2}/).toString();
42                     }
43                     else {
44                         return navigator.language;
45                     }
46                 }
47                 else if ( navigator.browserLanguage ) {
48                     return navigator.browserLanguage;
49                 }
50                 else if ( navigator.systemLanguage ) {
51                     return navigator.systemLanguage;
52                 }
53                 else if ( navigator.userLanguage ) {
54                     return navigator.userLanguage;
55                 }
56             }
57         }
58     } finally {
59         hwc.traceLeavingMethod("hwc.getCurrentLocale");
60     }
61 };
62
63 /**
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
64      * Returns a localized representation of the given Date
object. Queries the platform OS for a locale-
65      * formatted date/time string.
66      * @desc Timezone
67      * @memberOf hwc
68      * @public
69      * @param {Date} date Date to be localized, initialized to
some valid time.
70      * @returns {string} Returns a localized date/time string, or
undefined if platform is unsupported.
71      * @example
72      * var sDT = hwc.getLocalizedDateTime( date );
73      *
74      */
75      hwc.getLocalizedDateTime = function( date ) {
76          var result, dMilliseconds, sTzId, response;
77          hwc.traceEnteringMethod("hwc.getLocalizedDateTime");
78          try {
79              if (hwc.isAndroid()) {
80                  dMilliseconds = Date.UTC(date.getFullYear(),
date.getMonth(), date.getDate(), date.getHours(), date.getMinutes(),
date.getSeconds() );
81                  sTzId = _HWC.getLocalizedDateTime( dMilliseconds ) +
'';
82                  result = sTzId;
83              }
84              else if (hwc.isWindowsMobile()) {
85                  // Feature was not needed on this platform
86                  result = undefined;
87              }
88              else if (hwc.isiOS()) {
89                  dMilliseconds = Date.UTC(date.getFullYear(),
date.getMonth(), date.getDate(), date.getHours(), date.getMinutes(),
date.getSeconds() );
90                  response = hwc.getDataFromContainer("tz",
"&command=tzdatetime&time=" + dMilliseconds);
```

```

91         result = (response);
92     }
93     else if (hwc.isBlackBerry()) {
94         dMilliseconds = Date.UTC(date.getFullYear(),
date.getMonth(), date.getDate(), date.getHours(), date.getMinutes(),
date.getSeconds() );
95         sTzId = TimeZone.tzdatetime( dMilliseconds );
96         result = sTzId;
97     }
98     else {
99         result = undefined;
100    }
101    return result;
102    } finally {
103        hwc.traceLeavingMethod("hwc.getLocalizedDateTime");
104    }
105    };
106
107    /**
108     * Returns a localized representation of the given Date
object. Queries the platform OS for a locale-
109     * formatted date string.
110     * @desc Timezone
111     * @memberOf hwc
112     * @public
113     * @param {Date} date Date to be localized, initialized to
some valid time.
114     * @returns {string} Returns a localized date string, or
undefined if platform is unsupported.
115     * @example
116     * var sD = hwc.getLocalizedDate( date );
117     *
118     */
119    hwc.getLocalizedDate = function( date ) {

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
120     var dMilliseconds, sTzId, response, result;
121     hwc.traceEnteringMethod("hwc.getLocalizedDate");
122     try {
123         if (hwc.isAndroid()) {
124             dMilliseconds = Date.UTC(date.getFullYear(),
125                                     date.getMonth(), date.getDate(), 12, 0, 0 );
126             sTzId = _HWC.getLocalizedDate( dMilliseconds ) +
127                 '';
128             result = sTzId;
129         }
130     } else if (hwc.isWindowsMobile()) {
131         // Feature was not needed on this platform
132         result = undefined;
133     } else if (hwc.isIOS()) {
134         dMilliseconds = Date.UTC(date.getFullYear(),
135                                 date.getMonth(), date.getDate(), 12, 0, 0 );
136         response = hwc.getDataFromContainer("tz",
137                                             "&command=tzdate&time=" + dMilliseconds);
138         result = (response);
139     } else if (hwc.isBlackBerry()){
140         dMilliseconds = Date.UTC(date.getFullYear(),
141                                 date.getMonth(), date.getDate(), 12, 0, 0 );
142         sTzId = TimeZone.tzdate( dMilliseconds );
143         result = sTzId;
144     } else {
145         result = undefined;
146     }
147     return result;
148 } finally {
149     hwc.traceLeavingMethod("hwc.getLocalizedDate");
150 }
```

```

149     };
150
151     /**
152      * Returns a localized representation of the given Date
153      * object. Queries the platform OS for a locale-
154      * formatted time string.
155      * @desc Timezone
156      * @memberOf hwc
157      * @public
158      * @param {Date} date Date to be localized, initialized to
159      * some valid time.
160      * @returns {string} Returns a localized time string, or
161      * undefined if platform is unsupported.
162      * @example
163      * var sT = hwc.getLocalizedTime( date );
164
165      */
166     hwc.getLocalizedTime = function( date ) {
167         var dMilliseconds, sTzId, response, result;
168         hwc.traceEnteringMethod("hwc.getLocalizedTime");
169         try {
170             if (hwc.isAndroid()) {
171                 dMilliseconds = Date.UTC(date.getFullYear(),
172                 date.getMonth(), date.getDate(), date.getHours(), date.getMinutes(),
173                 date.getSeconds() );
174                 sTzId = _HWC.getLocalizedTime( dMilliseconds ) +
175                 '';
176                 result = sTzId;
177             }
178             else if (hwc.isWindowsMobile()) {
179                 // Feature was not needed on this platform
180                 result = undefined;
181             }
182             else if (hwc.isIOS()) {

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
177         dMilliseconds = Date.UTC(date.getFullYear(),
date.getMonth(), date.getDate(), date.getHours(), date.getMinutes(),
date.getSeconds() );

178         response = hwc.getDataFromContainer("tz",
"&command=tztime&time=" + dMilliseconds);

179         result = (response);

180     }

181     else if (hwc.isBlackBerry()){

182         dMilliseconds = Date.UTC(date.getFullYear(),
date.getMonth(), date.getDate(), date.getHours(), date.getMinutes(),
date.getSeconds() );

183         sTzId = TimeZone.tztime( dMilliseconds );

184         result = sTzId;

185     }

186     else if (hwc.isWindows()){

187         // For debugging on a browser of windows
platform

188         result = date.toString();

189     }

190     else {

191         result = undefined;

192     }

193     return result;

194 } finally {

195     hwc.traceLeavingMethod("hwc.getLocalizedTime");

196 }

197 };

198

199 /**

200  * Converts the given Date object to the device's local time,
and returns the new Date.

201  * @desc Timezone

202  * @memberOf hwc

203  * @public
```

```
204     * @param {Date} date Date to be converted, initialized to
some valid UTC time.
205     * @returns {Date} Returns the converted Date object.
206     * @example
207     * var localDate = hwc.convertUtcToLocalTime( date );
208     *
209     */
210     hwc.convertUtcToLocalTime = function( date )
211     {
212         hwc.traceEnteringMethod("hwc.convertUtcToLocalTime");
213         try {
214             var iMilliseconds, totalOffsetInMinutes, time,
localDate;
215             iMilliseconds = date.valueOf();
216             totalOffsetInMinutes =
hwc.getOffsetFromUTC( date );
217             totalOffsetInMinutes = totalOffsetInMinutes *
60000;
218             time = iMilliseconds + totalOffsetInMinutes;
219             localDate = new Date();
220             localDate.setTime( time );
221             return localDate;
222         } finally {
223             hwc.traceLeavingMethod("hwc.convertUtcToLocalTime");
224         }
225     };
226
227     /**
228     * Converts the given Date object to UTC time, and returns the
new Date.
229     * @desc Timezone
230     * @memberOf hwc
231     * @public
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
232      * @param {Date} date Date to be converted, initialized to
some valid local time.
233      * @returns {Date} Returns the converted Date object.
234      * @example
235      * var utcDate = hwc.convertLocalTimeToUtc( date );
236      *
237      */
238      hwc.convertLocalTimeToUtc = function( date )
239      {
240          hwc.traceEnteringMethod("hwc.convertLocalTimeToUtc");
241          try {
242              var iMilliseconds, totalOffsetInMinutes, time,
utcDate;
243              iMilliseconds = date.valueOf();
244              totalOffsetInMinutes =
hwc.getOffsetFromUTC( date );
245              totalOffsetInMinutes = totalOffsetInMinutes *
60000;
246              time = iMilliseconds - totalOffsetInMinutes;
247              utcDate = new Date();
248              utcDate.setTime( time );
249              return utcDate;
250          } finally {
251              hwc.traceLeavingMethod("hwc.convertLocalTimeToUtc");
252          }
253      };
254
255      /**
256      * Returns the total offset (difference) between the given
"local" time and UTC including any daylight
257      * savings offsets if applicable. Example: if the device was
in London timezone (Gmt +1) and it is
258      * currently practicing DST, the function would return "120":
60 minutes normal offset plus 60 minutes
```



```

259     * for its daylight savings offset.
260     * @desc Timezone
261     * @memberOf hwc
262     * @public
263     * @param {Date} date Date at which time to determine offset,
    initialized to some valid time.
264     * @returns {int} Returns the GMT offset in minutes.
265     * @example
266     * var totalOffset = hwc.getOffsetFromUTC(date);
267     *
268     */
269     hwc.getOffsetFromUTC = function( date )
270     {
271         var lMilliseconds, iMilliseconds, iMinutesOffset,
    response, dt,
272             year, month, day, hour, minute, second, request, d,
273             dMilliseconds, result;
274
275         hwc.traceEnteringMethod("hwc.getOffsetFromUTC");
276         try {
277             if (hwc.isAndroid()) {
278                 lMilliseconds = date.getTime();
279                 iMinutesOffset =
    _HWC.getOffsetFromUTC(lMilliseconds);
280                 result = iMinutesOffset;
281             }
282             else if (hwc.isWindows()) {
283                 dt = new Date();
284                 iMinutesOffset = dt.getTimezoneOffset() * (-1);
285                 result = iMinutesOffset;
286             }
287             else if (hwc.isWindowsMobile())
288                 {

```

Develop Hybrid Apps Using Third-party Web Frameworks

```
289 // JavaScript's Date and WM's DateTime objects
differs in their base starting time

290 // and definition. It was necessary to pass a
"time" to the OS - see below comment

291 lMilliseconds = date.getTime();

292 // Rather than pass a date string (which might be in
a different locale format)

293 // the raw parameters of the particular "date" are
sent

294 // this also avoids a date string parse on the OS
side.

295 year = date.getFullYear();
296 month = date.getMonth() + 1;
297 day = date.getDate();
298 hour = date.getHours();
299 minute = date.getMinutes();
300 second = date.getSeconds();
301 request = "utcoffset=utcoffset&";
302 request += "year=";
303 request += year.toString();
304 request += "&";
305 request += "month=";
306 request += month.toString();
307 request += "&";
308 request += "day=";
309 request += day.toString();
310 request += "&";
311 request += "hour=";
312 request += hour.toString();
313 request += "&";
314 request += "minute=";
315 request += minute.toString();
316 request += "&";
317 request += "second=";
```

```
318         request += second.toString();
319
320         response = hwc.postDataToContainer("tz",
request);
321         d = response * 1;
322         iMinutesOffset = d;
323
324         result = iMinutesOffset;
325     }
326     else if (hwc.isBlackBerry()){
327         dMilliseconds = date.getTime();
328         iMinutesOffset =
TimeZone.totaloffset(dMilliseconds);
329         result = iMinutesOffset;
330     }
331     else if (hwc.isIOS()) {
332         lMilliseconds = date.getTime();
333         result = hwc.getDataFromContainer("tz",
"&command=utcoffset&time=" + lMilliseconds);
334     }
335     else {
336         result = undefined;
337     }
338     return result;
339 } finally {
340     hwc.traceLeavingMethod("hwc.getOffsetFromUTC");
341 }
342 };
343
344 /**
345  * Returns whether daylight savings rules are in effect for
the current timezone at the given time.
346  * @desc Timezone
347  * @memberOf hwc
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
348     * @public
349     * @param {Date} date Date at which to determine whether
daylight savings is in effect.
350     * @returns {boolean} Returns true iff daylight savings rules
are in effect at the given time in the
351     * current timezone.
352     * @example
353     * var isAwareAtTime = hwc.isDstActiveAtGivenTime(date);
354     *
355     */
356     hwc.isDstActiveAtGivenTime = function( date )
357     {
358         var lMilliseconds, iMilliseconds, iMinutesOffset,
response, dt,
359         year, month, day, hour, minute, second, request, d,
360         dMilliseconds, result;
361         hwc.traceEnteringMethod("hwc.isDstActiveAtGivenTime");
362         try {
363             if (hwc.isAndroid()) {
364                 iMilliseconds = date.getTime();
365                 result =
_HWC.isDstActiveAtGivenTime(iMilliseconds);
366             }
367             else if (hwc.isWindowsMobile())
368             {
369                 // JavaScript's Date and WM's DateTime objects
differs in their base starting time
370                 // and definition. It was necessary to pass a
"time" to the OS - see below comment
371                 lMilliseconds = date.getTime();
372                 // Rather than pass a date string (which might be in
a different locale format)
373                 // the raw parameters of the particular "date" are
sent
374                 // this also avoids a date string parse on the OS
side.
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
375         request = "indst=indst&";
376         response = undefined;
377         year = date.getFullYear();
378         month = date.getMonth() + 1;
379         day = date.getDate();
380         hour = date.getHours();
381         minute = date.getMinutes();
382         second = date.getSeconds();
383
384         request += "year=";
385         request += year.toString();
386         request += "&";
387         request += "month=";
388         request += month.toString();
389         request += "&";
390         request += "day=";
391         request += day.toString();
392         request += "&";
393         request += "hour=";
394         request += hour.toString();
395         request += "&";
396         request += "minute=";
397         request += minute.toString();
398         request += "&";
399         request += "second=";
400         request += second.toString();
401
402         response = hwc.postDataToContainer("tz",
request);
403
404         result = (response === 'true');
405     }
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
406         else if (hwc.isBlackBerry()){
407             dMilliseconds = date.getTime();
408             result = TimeZone.indst(dMilliseconds);
409         }
410         else if (hwc.isIOS()) {
411             lMilliseconds = date.getTime();
412             response = hwc.getDataFromContainer("tz",
413 "&command=indst&time=" + lMilliseconds);
414             result = (hwc.parseBoolean(response));
415         }
416         else {
417             result = false;
418         }
419         return result;
420     } finally {
421         hwc.traceLeavingMethod("hwc.isDstActiveAtGivenTime");
422     };
423
424     /**
425      * Returns the daylight savings offset in minutes for the
426      * current timezone at the given time.
427      * Example: for Mountain Standard Time, at March 31st
428      * (currently is practicing DST), the returned offset is 60.
429      * Example: for Mountain Standard Time, at November 31st
430      * (currently is not practicing DST), the returned offset is 0.
431      * @desc Timezone
432      * @memberOf hwc
433      * @public
434      * @param {Date} date Date at which to determine daylight
435      * savings offset.
436      * @returns {int} Returns the number of minutes offset for
437      * daylight savings for the current
```

```
433     * timezone and at the given Date, or 0 if the current
434     * @example
435     * var iDstOffsetAtTime =
436     * hwc.getDstOffsetAtGivenTimeInMinutes(date);
437     */
438     hwc.getDstOffsetAtGivenTimeInMinutes = function ( date )
439     {
440         var lMilliseconds, iMilliseconds, iMinutesOffset,
441         response, dt,
442         year, month, day, hour, minute, second, request, d,
443         dMilliseconds, result;
444
445         hwc.traceEnteringMethod("hwc.getDstOffsetAtGivenTimeInMinutes");
446         try {
447             if (hwc.isAndroid()) {
448                 iMilliseconds = date.getTime();
449                 iMinutesOffset =
450                 _HWC.getDstOffsetAtGivenTimeInMinutes(iMilliseconds);
451                 result = iMinutesOffset;
452             }
453             else if (hwc.isWindowsMobile())
454             {
455                 // JavaScript's Date and WM's DateTime objects
456                 // differs in their base starting time
457                 // and definition. It was necessary to pass a
458                 // "time" to the OS - see below comment
459                 lMilliseconds = date.getTime();
460                 // Rather than pass a date string (which might be in
461                 // a different locale format)
462                 // the raw parameters of the particular "date" are
463                 // sent
464                 // this also avoids a date string parse on the OS
465                 // side.
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
459         request = "dstoffset=dstoffset&";
460         year = date.getFullYear();
461         month = date.getMonth() + 1;
462         day = date.getDate();
463         hour = date.getHours();
464         minute = date.getMinutes();
465         second = date.getSeconds();
466
467         request += "year=";
468         request += year.toString();
469         request += "&";
470         request += "month=";
471         request += month.toString();
472         request += "&";
473         request += "day=";
474         request += day.toString();
475         request += "&";
476         request += "hour=";
477         request += hour.toString();
478         request += "&";
479         request += "minute=";
480         request += minute.toString();
481         request += "&";
482         request += "second=";
483         request += second.toString();
484
485         response = hwc.postDataToContainer("tz",
request);
486         d = response * 1;
487         iMinutesOffset = d;
488
489         result = iMinutesOffset;
```



```
490     }
491     else if (hwc.isBlackBerry()){
492         dMilliseconds = date.getTime();
493         iMinutesOffset =
494         TimeZone.dstoffset(dMilliseconds);
495         result = iMinutesOffset;
496     }
497     else if (hwc.isIOS()) {
498         lMilliseconds = date.getTime();
499         response = hwc.getDataFromContainer("tz",
500         "&command=dstoffset&time=" + lMilliseconds);
501         result = parseInt(response, 10);
502     }
503     else {
504         result = undefined;
505     }
506     return result;
507 } finally {
508     hwc.traceLeavingMethod("hwc.getDstOffsetAtGivenTimeInMinutes");
509 }
510 /**
511  * Returns a string containing the current Timezone's standard
512  * name. The name will not change based
513  * on daylight savings periods. The native OS returns the
514  * string in the current locale where applicable.
515  * Currently this string is derived from using available
516  * platform OS APIs. The values for the same
517  * timezone will be different among platforms.
518  * @desc Timezone
519  * @memberOf hwc
520  * @public
```

Develop Hybrid Apps Using Third-party Web Frameworks

```
518     * @returns {string} Returns a string containing the current
519     * @example
520     * var sTzId = hwc.getTimezoneId();
521     *
522     */
523     hwc.getTimezoneId = function () {
524         var sTzId, request, response, result;
525
526         hwc.traceEnteringMethod("hwc.getTimezoneId");
527         try {
528             if (hwc.isAndroid()) {
529                 sTzId = _HWC.getTimezoneId() + '';
530                 result = sTzId;
531             }
532             else if (hwc.isWindowsMobile())
533             {
534                 request = "tzid=tzid";
535                 response = hwc.postDataToContainer("tz",
536 request);
537                 result = response;
538             }
539             else if (hwc.isIOS()) {
540                 response = hwc.getDataFromContainer("tz",
541 "&command=tzid");
542                 result = (response);
543             }
544             else if (hwc.isBlackBerry()){
545                 sTzId = TimeZone.tzid();
546                 result = sTzId;
547             }
548             else {
549                 result = undefined;
```

```
548         }
549         return result;
550     } finally {
551         hwc.traceLeavingMethod("hwc.getTimezoneId");
552     }
553 };
554
555 /**
556  * Returns whether the device's current timezone practices
557  * daylight savings. If a device's current
558  * timezone never practices daylight savings, this function
559  * returns "false". If a device's current
560  * timezone practices DST, but DST rules are not currently in
561  * effect, function returns "true".
562  * @desc Timezone
563  * @memberOf hwc
564  * @public
565  * @returns {boolean} Returns true iff the device's current
566  * timezone practices daylight savings,
567  * irrespective of whether daylight savings is currently in
568  * effect.
569  * @example
570  * var isDstAware = hwc.getUsesDST();
571  */
572 hwc.getUsesDST = function () {
573     var date, lMilliseconds, request, response, result;
574
575     hwc.traceEnteringMethod("hwc.getUsesDST");
576     try {
577         if (hwc.isAndroid()) {
578             result = _HWC.useDaylightTimeCurrently();
579         }
580         else if (hwc.isWindowsMobile())
```

```
577         {
578             date = new Date();
579             lMilliseconds = date.getTime();
580             request = "dstaware=";
581             response = undefined;
582
583             request += lMilliseconds.toString(); // left for
potential future use
584
585             response = hwc.postDataToContainer("tz",
request);
586             result = (response === 'true');
587         }
588         else if (hwc.isIOS()) {
589             response = hwc.getDataFromContainer("tz",
"&command=dstaware");
590             result = hwc.parseBoolean(response);
591         }
592         else if (hwc.isBlackBerry()){
593             result = TimeZone.dstaware();
594         }
595         return result;
596     } finally {
597         hwc.traceLeavingMethod("hwc.getUsesDST");
598     }
599 };
600
601 }) (hwc, window);
602
```

MBO Access JavaScript API Samples

This section shows some sample JavaScript APIs that access MBOs.

Calling a Create Function

1. Create a JavaScript object, in this case, Department.

```
var dep1 = new Department();
```

2. Set the values for all the fields. The fields names map to the Department MBO create operation's parameter name.

```
dep1.dept_id = "800";
dep1.dept_name="Dept";
dep1.dept_head_id="888";
```

3. Call the create online request function.

```
department_create_onlineRequest(dep1,
    "supusername=supAdmin&suppassword=s3pAdmin",
    function() { alert("error occurred");});
```

4. For an online request, you should implement the hwc.processDataMessage function, for example:

```
hwc.processDataMessage = function
processDataMessage(incomingWorkflowMessage, noUI, loading,
fromActivationFlow, dataType) {
    if
    ( (incomingWorkflowMessage.indexOf("<XmlWidgetMessage>") === 0)
    ||
    (incomingWorkflowMessage.indexOf("<XmlWorkflowMessage>") === 0)
    || (incomingWorkflowMessage.indexOf("<M>")
    === 0)) {
        var workflowMessage = new
WorkflowMessage(incomingWorkflowMessage);

if ( workflowMessage.getRequestAction() ==
Department.createAction ){
    alert("Department id=" +
workflowMessage.getValues().getData('Department_create_dept_id_pa
ramKey').getValue() + " has been created!");
    }else if ( workflowMessage.getRequestAction()
    ==Sales_order.findAllAction){
        alert("Return Item count =" +
workflowMessage.getValues().getData('Sales_order').value.length )
;
//By default database it should return 54 items.
    }
    }else{
        alert("TODO: Please fix me,
incomingWorkflowMessage="+ incomingWorkflowMessage);
    }
}
```

Calling an Update Function With Old Arguments

1. Set old arguments values:

```
var oldDep = new Department();
oldDep.dept_id = "800";
```

```
oldDep.dept_name="Dept";  
oldDep.dept_head_id="888";
```

2. Set the new values:

```
var newDep = new Department();  
newDep.dept_id = "800";  
newDep.dept_name="DeptUpdated";  
newDep.dept_head_id="777";
```

3. Call the update submit function:

```
department_update_submit(newDep, oldDep, "", true );
```

Passing a Personalization Key Value

1. Create Sales_order object:

```
var sales_order = new Sales_order();
```

2. Set the onload personalization key value:

```
sales_order.pks.put(Sales_rep_PK_pkKey, "667");
```

3. Call the findAll online request:

```
sales_order_findAll( sales_order , "", function() {});  
}
```

4. In the process workflowMessage function, to process incoming message, add:

```
hwc.processDataMessage=function processDataMessage  
(incomingWorkflowMessage, noUI, loading, fromActivationFlow,  
dataType) {  
  
    if  
( (incomingWorkflowMessage.indexOf("<XmlWidgetMessage>") === 0)  
    ||  
(incomingWorkflowMessage.indexOf("<XmlWorkflowMessage>") === 0)  
    || (incomingWorkflowMessage.indexOf("<M>")  
    === 0)) {  
        var workflowMessage = new  
WorkflowMessage(incomingWorkflowMessage);  
if ( workflowMessage.getRequestAction() ==  
Sales_order.findAllAction){  
            alert("Return Item count =" +  
workflowMessage.getValues().getData('Sales_order').value.length )  
; //By default database it should return 54 items.  
        }  
    }else{  
  
        alert("TODO: Please fix me,  
incomingWorkflowMessage="+ incomingWorkflowMessage);  
    }  
  
}
```

Calling a Create Function on MBOs With a One to Many Relationship

1. Create a new Department:

```
var dep = new Department();  
dep.dept_id="2";
```

```
dep.dept_name="My Dep";
dep.dept_head_id="1";
```

2. Create a new employee:

```
var emp1 = new Employee();
emp1.emp_id = "1";
emp1.manager_id = "2";
emp1.emp_fname = "Yan";
emp1.emp_lname= "Gong";
emp1.street ="King Street";
emp1.city="Waterloo";
emp1.state ="ON";
emp1.zip_code ="n2v314";
emp1.phone="518-8836863";
emp1.status="A";
emp1.ss_number="024601768"
emp1.salary ="324234";
emp1.start_date="1996-12-30";
emp1.termination_date ="1999-12-20";
emp1.birth_date ="1956-12-20";
emp1.bene_health_ins ="Y";
emp1.bene_life_ins ="Y";
emp1.bene_day_care="Y";
emp1.sex="F";
```

3. Create a second employee:

```
var emp2 = new Employee();
emp2.emp_id = "2";
emp2.manager_id = "2";
emp2.emp_fname ="Yan2";
emp2.emp_lname= "Gong2";
emp2.street ="King Street";
emp2.city="Waterloo";
emp2.state ="ON";
emp2.zip_code ="n2v314";
emp2.phone="518-8836863";
emp2.status="A";
emp2.ss_number="024601768"
emp2.salary ="324234";
emp2.start_date="1996-12-30";
emp2.termination_date ="1999-12-20";
emp2.birth_date ="1956-12-20";
emp2.bene_health_ins ="Y";
emp2.bene_life_ins ="Y";
emp2.bene_day_care="Y";
emp2.sex="F";
```

4. Add the two employees to Department:

```
dep.Employee.push( emp1 );
dep.Employee.push( emp2 );
```

5. Call department create online request, it would create a new department and two new employees entries in the database:

```
department_create_onlineRequest(dep,
    "", function() {});
```

Calling a Delete Function on MBOs With a One to Many Relationship

To delete an MBO and its children, you need to find the MBO instance online request and, from the `processDataMessage` function, after the online request, you need to find each child's surrogate key value from the incoming message, create a child JavaScript instance, then add the child JavaScript instance to the parent JavaScript instance. Subsequently, when the `delete` function is called on the parent instance, the children are also deleted. The details of this are shown in this example in **bold** font.

If the delete operation has old value arguments, you also need to set old values for parent and child MBOs. This example assumes the delete operation has old value arguments, and the data (1 department and 2 employee) has been inserted into back end:

1. Call the `department_findByPrimaryKey` online request to find the department instance:

```
function deleteDepartment() {

    var dep = new Department();
    dep.dept_id="2";

    alert("before delete Deperatment and its children Empeoyee, we need
to call findByPrimary key first.")
    department_findByPrimaryKey( dep, "" , function(error)
    {alert(error)});

}
```

2. In the `processDataMessage` function, find the surrogatekey value for each Employee and create Employee instance and add it to department instance:

```
if ( workflowMessage.getRequestAction() ===
Department.findByPrimaryKeyAction) {

    var employees =
    workflowMessage.getValues().getData('Department_employees').value
    ;

    if
    ( workflowMessage.getValues().getData('Department_dept_id_attribK
ey').getValue() == '2') {
    var dep = new Department();
    dep.dept_id=workflowMessage.getValues().getData('Department_dept_
id_attribKey').getValue();
    dep.dept_head_id=workflowMessage.getValues().getData('Department_
dept_head_id_attribKey').getValue();
    dep.dept_name=workflowMessage.getValues().getData('Department_dep
t_name_attribKey').getValue();

    var oldDep = new Department();
    oldDep.dept_id=workflowMessage.getValues().getData('Department_de
pt_id_attribKey').getValue();
    oldDep.dept_name=workflowMessage.getValues().getData('Department_
dept_name_attribKey').getValue();
    oldDep.dept_head_id=workflowMessage.getValues().getData('Departme
nt_dept_head_id_attribKey').getValue();
```



```

for( var i = 0; i < employees.length ; i++ ) {
    var emp = new Employee();
    emp.emp_id =
employees[i].getData('Employee_emp_id_attribKey').getValue();
    emp.manager_id =
employees[i].getData('Employee_manager_id_attribKey').getValue();
    emp.emp_fname =
employees[i].getData('Employee_emp_fname_attribKey').getValue();
    emp.emp_lname =
employees[i].getData('Employee_emp_lname_attribKey').getValue();
    emp.dept_id =
employees[i].getData('Employee_dept_id_attribKey').getValue();
    emp.street =
employees[i].getData('Employee_street_attribKey').getValue();
    emp.city =
employees[i].getData('Employee_city_attribKey').getValue();
    emp.state =
employees[i].getData('Employee_state_attribKey').getValue();
    emp.zip_code =
employees[i].getData('Employee_zip_code_attribKey').getValue();
    emp.phone =
employees[i].getData('Employee_phone_attribKey').getValue();
    emp.status =
employees[i].getData('Employee_status_attribKey').getValue();
    emp.ss_number =
employees[i].getData('Employee_ss_number_attribKey').getValue();
    emp.salary =
employees[i].getData('Employee_salary_attribKey').getValue();
    emp.start_date =
employees[i].getData('Employee_start_date_attribKey').getValue().
substr(0, 10);
    emp.termination_date =
employees[i].getData('Employee_termination_date_attribKey').getVal
ue().substr(0, 10);
    emp.birth_date =
employees[i].getData('Employee_birth_date_attribKey').getValue().
substr(0, 10);
    emp.bene_health_ins =
employees[i].getData('Employee_bene_health_ins_attribKey').getVal
ue();
    emp.bene_life_ins =
employees[i].getData('Employee_bene_life_ins_attribKey').getValue
();
    emp.bene_day_care =
employees[i].getData('Employee_bene_day_care_attribKey').getValue
();
    emp.sex =
employees[i].getData('Employee_sex_attribKey').getValue();
    //set surrogateKey for employ
    emp._surrogateKey
=employees[i].getData('_surrogateKey').getValue();
    dep.Employee.push(emp );
    dep.OldValue_Employee.push( emp );
}

```

3. Call `department_delete_onlineRequest` to delete the department and all of its children:

```
department_delete_onlineRequest( dep, oldDep, function( error)
{ alert(error)});

}
.....
}
```

MediaCache Examples

```
var resourceUrl = "http://someserver/someimage.jpg";
document.write("<img src=\"\" + MediaCache.getUrl(resourceUrl,
hwc.MediaCache.Policy.CACHE_FIRST) + \"\" />");
```

```
var oImg=document.createElement("img");
oImg.setAttribute('src', MediaCache.getUrl('http://someserver/
someimage.jpg'));
document.body.appendChild(oImg);
```

Null Value Support

Null data values are represented in `MessageValue` objects, belonging to a `MessageValue` collection that is created from the data message sent by the server.

Note: Null data values are not supported on the Windows Mobile platform.

This document refers to example HTML. You can see the example HTML by downloading the *hybridapp_null_value.zip* file and extracting the *hybridapp_null_value.html* file.

In the example, `MessageValueCollection` is referenced by `var values = myDataMessage.getValues();` and `myDataMessage` is created in the `onHybridAppLoad` method.

A specific `MessageValue` is referenced by `values.getData(string key)`. If you use the Hybrid App designer, the IDE manages keys for you but with JavaScript API, you have to implement key management in the code yourself.

This example follows the IDE style of giving each control an ID and using that as the key for data that will be used in that control:

```
<input class="right" type="number"
id="Nullvaluetest_int_value2_attribKey"
smp_allows_null="true" smp_valuechanged="false"
onchange="inputChanged(this)"/>
```

So if you want the `MessageValue` object corresponding to that control:

```
var value = values.getData("Nullvaluetest_int_value2_attribKey");
```

Once you have the `MessageValue` object you can see if it is null with:

```
var isNull = value.getNullAttribute();
```

Null Values and HTML

HTML usually puts an empty string into a control if it is assigned a null value. If the control is not changed but you get the data from it for its `MessageValue` object, the `MessageValue` object will have an empty string as its value instead of `NULL`. This means the `NullAttribute` is not set properly unless you set it yourself.

When using null values, keep in mind that the contents of the control do not tell you whether it should be null. This can cause bad data on the server. Putting an empty string into a number type `MessageValue` can throw a formatting exception on the server, so when using JavaScript API, you are responsible for maintaining null values.

The Sample HTML

This section references the `hybridapp_null_value.html` file to show examples of how to implement null values.

- **Recognizing NULL values** – The example uses the same techniques as an Hybrid App generated with the designer to keep track of data values, keys, controls and null-ness.

Controls that allow null have a special attribute that identifies it is okay to be `NULL`:

```
<input class="right" type="number"
id="Nullvaluetest_int_value2_attribKey"
smp_allows_null="true" smp_valuechanged="false"
onchange="inputChanged(this)"/>
```

This example processes the incoming data message and checks control attributes for null friendliness and issues an alert message for a null value in the wrong place.

```
hwc.processDataMessage = function(incomingDataMessageValue, noUI,
loading, fromActivationFlow, dataType)
```

- **Handling input to NullValue controls** – This example uses an event handler to recognize user input:

```
<input class="right" type="number"
id="Nullvaluetest_int_value2_attribKey"
smp_allows_null="true" smp_valuechanged="false"
onchange="inputChanged(this)"/>
```

`inputChanged` uses another special attribute to indicate that the user has put something in the control and it is no longer null.

- **Setting a value to NULL** – In the HTML example, `setKeyValueNull` and `setControlValueNull` show how to set a value to null while managing the control attributes and the `MessageValue` null attribute.
- **Sending data to the server** – In the example HTML, `doUpdate` uses the `getUpdatedValue` method to set the right value in the `newNVT` object. `getUpdatedValue` checks the control attributes and the `MessageValue` null attribute to decide what to send to the server.

- **Creating data with null values** – In the example HTML, `doCreate` and `doCreate2` show two ways of creating a record with null values.

Calling the Hybrid Web Container

It is easiest to learn how to call the Hybrid Web Container by examining the `API.js` and `Utils.js` files, which are located in `<SMP_HOME>\MobileSDK<version>\HybridApp\API\AppFramework`.

Making calls to the Hybrid Web Container is platform-dependent, as shown in this example:

```
if (isWindowsMobile()) {
    var xmlhttp = getXMLHttpRequest();
    xmlhttp.open("POST", "/sup.amp?
querytype=setscreentitle&version=2.0", false);
    xmlhttp.send("title=" + encodeURIComponent(screenTitle));
}
else if (isIOS()) {
    var xmlhttpReq = getXMLHttpRequest();
    xmlhttpReq.open("GET", "http://localhost/sup.amp?
querytype=setscreentitle&version=2.0&title=" +
encodeURIComponent(screenTitle), true);
    xmlhttpReq.send("");
}
else if (isAndroid()) {
    var request = "http://localhost/sup.amp?
querytype=setscreentitle&version=2.0&title=" +
encodeURIComponent(screenTitle);
    _WorkflowContainer.getData(request);
}
else { //must be BlackBerry
    var xmlhttp = getXMLHttpRequest();
    xmlhttp.open("POST", "http://localhost/sup.amp?
querytype=setscreentitle&version=2.0", false);
    xmlhttp.send("title=" + encodeURIComponent(screenTitle));
}
```

From a high-level perspective, these are the query types used for calling the Hybrid Web Container.

setscreentitle

Sets the native screen title on the Hybrid Web Container.

close

Closes the native Hybrid Web Container (Windows Mobile only).

addMenuItem

Adds a single menu item to the Hybrid Web Container.

removeallmenuitems

Removes all the menu items from the Hybrid Web Container.

clearrequestcache

Clears the entire Online Request cache for the current Hybrid App.

clearrequestcacheitem

Clears a single Online Request cache entry for the current Hybrid App.

logtoworkflow

Logs a message to the `AMPHostLog.txt` (`mocalog.txt` for iOS) on the device. You can retrieve this log file remotely from SAP Control Center.

showcertpicker

Shows a native platform certificate picker on the device for selecting certificate credentials.

showInBrowser

On iOS, this function shows the URL in the Hybrid Web Container in a separate browser instance. On all other platforms, this launches the native Web browser in another window with the given URL.

showattachment

Using third party file viewers, this function displays an attachment that has previously been downloaded using the `downloadattachment` querytype in a separate window.

Note: On iOS, the attachment is shown within the Hybrid Web Container.

showlocalattachment

Using third party file viewers, this function displays an attachment that was included as part of the Hybrid App .zip package, in a separate window.

Note: On iOS, the attachment is shown within the Hybrid Web Container.

rmi

This function executes an online request to the SAP Mobile Server synchronously, in other words, a network connection must be available. This can indicate results should be cached for future access (in which case a network connection does not need to be available).

downloadattachment

Requests an attachment to be downloaded from the SAP Mobile Server through an object query. A network connection is required for this operation. This operation occurs asynchronously, and the calling JavaScript is notified when it is complete.

submit

Submits the current `MessageValueCollection` to the SAP Mobile Server for processing by the server plug-in. This operation occurs asynchronously. If a network

Develop Hybrid Apps Using Third-party Web Frameworks

connection is not available when this operation is performed, the request is queued up and executed the next time a network connection is available.

alert

Shows a message box in native code (iOS and Android platforms only).

loadtransformdata

Requests the Hybrid Web Container for the transform data (the contents of the e-mail message) for the current message.

addallmenuitems

Instructs the Hybrid Web Container to add the supplied list of menu items.

formredirect

Notifies the Hybrid Web Container that a screen navigation is occurring, and to update credentials in the credentials cache, if required.

AttachmentViewer and Image Limitations

There are some limitations on the size of the attachments and images that you can include as part of the Hybrid App message.

These limitations vary by platform.

Platform	Size Limit
iOS	Large attachments can produce longer processing times.
Android	Large attachments can produce longer processing times. There is a 1MB limit for attachments on Android devices.
Windows Mobile	The maximum size of a JavaScript variable for Windows Mobile is 2MB, which allows for more memory. Warning messages are shown if the script continues for a long time, which can cause the memory to run out.
BlackBerry 5.0 and BlackBerry 6.0	On BlackBerry 5.0, the maximum size of a JavaScript variable is 500KB and on BlackBerry 6.0 and later, the maximum size of a JavaScript variable is 2MB. The maximum size must be larger than the attachment and the rest of the Hybrid App message. If the attachment is Base64-encoded, also allow for an increase in the attachment size.

Note: When accessing very large binary (image) data in the mobile business object associated with the Hybrid App, ensure that the attribute set in the mobile business object is a **BigBinary** datatype, rather than **Binary**.

Package Hybrid Apps

Package the files for the Hybrid App so that you can deploy them to the server.

Packaging Hybrid Apps Using the Packaging Tool

Use the packaging tool to package existing files into a Hybrid App package.

1. Navigate to `<SMP_HOME>\MobileSDK23\HybridApp\PackagingTool` and double-click the `packagingtool.bat` file if you are using a 32-bit JDK, or `packagingtool64.bat` if you are using a 64-bit JDK.
2. Click **Browse** to enter the filepath for the output directory where your projects are located, and click **OK**.
All of the projects stored in the output directory you set appear in the Project Explorer list box.
3. (Optional) Select a project to see the details of the project in the right pane. You can make changes to any of the General Information properties and click **Save**.
4. (Optional) To create a new project:
 - a) Click **New** at the bottom of the Project Explorer list box.
 - b) Enter a project name.
 - c) Click **Browse** to select a folder for the Web application folder from the local hard disk.

The Web root folder is the location of your HTML files, typically, with any dependent HTML, JavaScript, CSS, images, and so on, files being in the same folder or subfolders. The `WorkflowClient.xml` file should also be in the Web application folder.

Note: The Web application folder cannot be a subfolder of the workspace, and the workspace folder cannot be a subfolder of the Web application folder.

- d) Click **OK**.

The new project name is added to the Project Explorer, and a project file is created in the workspace folder with the `.pkgproj` extension. The project will have a separate folder under the workspace to store all temporary files for deployment.

5. (Optional) To remove a project from Project Explorer, select the project to remove and click **Delete** at the bottom of the Project Explorer list box.
6. Set the configuration information for the project in the General Information tab.

- Module name – the name of the Hybrid App on the server. The default value is the project name. This is required.
 - Module version – this can be any number. The default value is 1. It is required.
 - Module description – (optional) enter description text.
 - Display name – (optional) the display name.
 - Client icon – the default value is 48. It is required.
 - MBO package name – if the Hybrid App uses MBOs, input the MBO package name.
 - MBO package version – enter the version for the MBO package.
 - Invokable on client – a boolean value to determine whether the Hybrid App can be invoked from the client. The default value is true.
 - Processed Messages
 - Mark as read – the default value is false.
 - Delete – the default value is true.
 - Cache key – (optional) the key to represent the cache.
 - Activation key – (optional) define the key to use.
 - Shared storage key – (optional) enter the shared storage key.
 - SAP Mobile Platform server information – the manifest.xml file may require hard-coded credentials for logging in to SAP Mobile Server.
 - User name – enter the user name for logging into SAP Mobile Server.
 - Simple password – enter the password for logging into SAP Mobile Server.
 - Certificate – enter the certificate information for logging into SAP Mobile Server.
7. Click the applicable platform tab to choose files for packaging.

Five platforms are available: Android, BlackBerry 5, BlackBerry 6, iOS, and Windows Mobile 6. For each platform, you can choose whether to include the specific platform in the package, the files needed for the platform, the HTML files for the the platform, and the start screen to show for this platform.

The start screen is the screen to show by default for the selected platform. The html (or htm) file in the HTML File for the Start Page textbox is parsed and all screens are then listed. If the file is not an html file or there is no screen defined in the file, the start screen textbox is empty.

8. (Optional) Click the **Matching Rules** tab to add matching rules.

Matching rules describe the collection of rules that are used to determine if a given server notification will be sent to the application for processing. Each matching rule describes the field to test (for example, Subject), and the regular expression to test against for matches.

9. (Optional) Click **Custom Icon** to add a custom icon for the Hybrid App package.

When you add a custom icon, the manifest.xml file is updated when you generate the package.

10. (Optional) Click **Client Variables** to add client variables for data that is associated with a particular client and application and that must be saved between user sessions.

11. Click **Generate**.

Configuration files are created and packaged in a ZIP file and placed in the output directory you specified.

Refreshing the Packaging Tool Treeview

Refresh the treeview to reflect the latest changes to the package.

There are several ways to refresh the treeview.

- Exit the packaging tool and restart it. All the new files appear in the treeview automatically.
- Switch to another project, then switch back.
- Click the **Support <xxx> platform** checkbox, uncheck it and then check it. When you set the **Support <xxx> platform** checkbox to true, the treeview is refreshed to get the latest files from the Web app folder.

Packaging Hybrid Apps Manually

While using the packaging tool is the easiest way to package Hybrid Apps, it is also possible to create a Hybrid App package without the tool.

Hybrid AppPackage Files

To build a Hybrid App package manually, you should first familiarize yourself with its contents.

This section describes the contents of the Hybrid App package—which files are required, and what the contents of those files should be. Particular attention is paid to the contents of the `manifest.xml` and `WorkflowClient` XML files, along with the Web application files (HTML, JavaScript, CSS), most specifically the public API functions available to you.

The Web Application Files

A Hybrid App package contains Web application files.

When developing a Hybrid App package manually:

- Include HTML files that follow the same general pattern as the files generated when using the Hybrid App Designer to generate the Hybrid App package.
- Use the `API.js`, `Callbacks.js`, `Camera.js`, `Certificate.js`, `ExternalResource.js`, `SUPStorage.js`, and `Timezone.js` files to communicate with the Hybrid Web Container. These files are in the `<SMP_HOME>\MobileSDK23\HybridApp\API\Container` and `<SMP_HOME>\MobileSDK23\HybridApp\API\AppFramework` directories.
- Use `WorkflowMessage.js` to view and manipulate the Hybrid App messages. This file is located in `<SMP_HOME>\MobileSDK23\HybridApp\API\AppFramework`

HTML Format

This is a commonly used HTML format.

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
    <meta name="HandheldFriendly" content="True" />
    <meta http-equiv="PRAGMA" content="NO-CACHE" />
    <link rel="stylesheet" href="css/MyStylesheet.css"
type="text/css" />
    [...]
    <script src="js/API.js"></script>
    <script src="js/Utils.js"></script>
    <script src="js/WorkflowMessage.js"></script>
    <script src="js/MyJavaScript.js"></script>
    [...]
    <script>
[...]
```

```
    </script>
  </head>
  <body onload="onHybridAppLoad();">
    <div id="Screen1KeyScreenDiv" smp_screen_title="Screen1Title"
style="display: none"
smp_menuitems="NativeMenu1Key,NativeMenu1DisplayName,NativeMenu2Key
,NativeMenu2DisplayName" smp_okaction="myOKAction()">
[...]
```

```
      <form style="margin: 0px;" name="Screen1KeyForm"
id="Screen1KeyForm" onSubmit="return false;" autocomplete="on">
[...]
```

```
        </form>
[...]
```

```
      </div>
    </body>
    <script>
[...]
```

```
$(document).ready( function() {
    [...]
});
[...]
```

```
</script>
</html>
```

Manifest.xml File

The manifest.xml file describes how the contents of the Hybrid App package .zip file are organized.

This file must reside at the root of the Hybrid App ZIP package. This shows the outline of what the manifest.xml file contains.

Manifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Manifest xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```

xsi:noNamespaceSchemaLocation="AMPManifest.xsd">
  <ModuleName>...</ModuleName>
  <ModuleVersion>...</ModuleVersion>
  <ModuleDesc>...</ModuleDesc>
  <ModuleDisplayName>...</ModuleDisplayName>
  <ClientIconIndex>...</ClientIconIndex>
  <InvokeOnClient>...</InvokeOnClient>
  <PersistAppDomain>...</PersistAppDomain>
  <MarkProcessedMessages>...</MarkProcessedMessages>
  <DeleteProcessedMessages>...</DeleteProcessedMessages>
  <ProcessUpdates>...</ProcessUpdates>
  <CredentialsCache>...</CredentialsCache>
  <RequiresActivation>...</RequiresActivation>
  < SharedStorage key> ... </ SharedStorage >

<TransformPlugin>
<File shared="true">WorkflowClient.dll</File>
<Class>Sybase.UnwiredPlatform.WorkflowClient.Transformer</Class>
</TransformPlugin>
- <ResponsePlugin>
<File shared="true">WorkflowClient.dll</File>
<Class>Sybase.UnwiredPlatform.WorkflowClient.Responder</Class>
</ResponsePlugin>

<ClientWorkflows>
  <WindowsMobileProfessional>
    <HTMLWorkflow>
      <File>...</File>
      <HtmlFiles>
        <HtmlFile>...</HtmlFile>
        <HtmlFile>...</HtmlFile>
      </HtmlFiles>
    </HTMLWorkflow>
  </WindowsMobileProfessional>
  <BlackBerry>
    <HTMLWorkflow>
      <File>...</File>
      <HtmlFiles>
        <HtmlFile>...</HtmlFile>
        <HtmlFile>...</HtmlFile>
      </HtmlFiles>
    </HTMLWorkflow>
  </BlackBerry>
  <BlackBerry6>
    <HTMLWorkflow>
      <File>...</File>
      <HtmlFiles>
        <HtmlFile>...</HtmlFile>
        <HtmlFile>...</HtmlFile>
      </HtmlFiles>
    </HTMLWorkflow>
  </BlackBerry6>
  <Android>
    <HTMLWorkflow>
      <File>...</File>

```

```
<HtmlFiles>
  <HtmlFile>...</HtmlFile>
  <HtmlFile>...</HtmlFile>
</HtmlFiles>
</HTMLWorkflow>
</Android>
<iPhone>
  <HTMLWorkflow>
    <File>...</File>
    <HtmlFiles>
      <HtmlFile>...</HtmlFile>
      <HtmlFile>...</HtmlFile>
    </HtmlFiles>
  </HTMLWorkflow>
</iPhone>
</ClientWorkflows>

<ContextVariables>
  <ContextVariable>
    <Name>...</Name>
    <Value>...</Value>
    <Certificate>...</Certificate>
    <Password>...</Password>
  </ContextVariable>
</ContextVariables>

<Metadata version="1">
  <Item>
    <Name>Key1</Name>
    <Value>Value1</Value>
  </Item>
  <Item>
    <Name>Key2</Name>
    <Value>Value2</Value>
  </Item>
</Metadata>
<MatchRules>
  <SubjectRegExp>...</SubjectRegExp>
  <ToRegExp>...</ToRegExp>
  <FromRegExp>...</FromRegExp>
  <CCRegExp>...</CCRegExp>
  <BodyRegExp>...</BodyRegExp>
</MatchRules>
</Manifest>
```

ModuleName

```
<ModuleName>SampleActivitiesModule</ModuleName>
```

The `ModuleName` defines the name of the Hybrid App package.

ModuleVersion

```
<ModuleVersion>2</ModuleVersion>
```

The `ModuleVersion` defines the version of the Hybrid App package.

ModuleDesc

```
<ModuleDesc>AMP Sample - Activities Collection</ModuleDesc>
```

The `ModuleDesc` provides a short description of the Hybrid App package.

ModuleDisplayName

```
<ModuleDisplayName>Activities Sample</ModuleDisplayName>
```

The name of the Hybrid App package that is displayed to the user in the Hybrid App list on the device for Hybrid Apps that are client-invoked. When the Hybrid App package is deployed, you can override the `DisplayName` specified here with one of your own choosing.

ClientIconIndex

```
<ClientIconIndex>35</ClientIconIndex>
```

The index of the icon associated with the Hybrid App package. This icon is shown beside the e-mail message in the device's Inbox listing instead of the regular e-mail icon. When the Hybrid App package is deployed, you can override the icon that is specified here with one of your own choosing.

InvokeOnClient

```
<InvokeOnClient>1</InvokeOnClient>
```

Specifies whether this Hybrid App can be used without an associated e-mail. 1 = true, 0 = false. If 1 is specified, the Hybrid App is shown in the Hybrid App list on the device and can be used without the context of an e-mail message.

PersistAppDomain

```
<PersistAppDomain>1</PersistAppDomain>
```

States whether this Hybrid App uses a persistent application domain when the .NET assembly plugin is loaded. 1 = true, 0 = false. By default, it is set to false, meaning an application domain is created and removed every time the plugin is loaded.

MarkProcessedMessages

```
<MarkProcessedMessages>1</MarkProcessedMessages>
```

Indicates whether a Hybrid App message shows a visual indication in the Inbox after it has been processed (1 = true, 0 = false).

Note: When a Hybrid App message shows a visual indication that it has been processed, the visual indication disappears if the device is re-registered, or if the device user performs a `Refresh All Data` action.

DeleteProcessedMessages

```
<DeleteProcessedMessages>1</DeleteProcessedMessages>
```

Indicates whether a Hybrid App message is deleted from the mobile device's Inbox after it has been processed (1 = true, 0 = false).

Note: You cannot set both `DeleteProcessedMessages` and `MarkProcessedMessages` to true (1). To set `MarkProcessedMessages` to true, you must set `DeleteProcessedMessages` to false (0) as shown:

```
<MarkProcessedMessages>1</MarkProcessedMessages>
  <DeleteProcessedMessages>0</DeleteProcessedMessages>
```

ProcessUpdates

```
<ProcessUpdates>1</ProcessUpdates>
```

Indicates whether Hybrid App messages associated with this Hybrid App package that are already delivered to the device can be updated from the server with modified content. (1 = true, 0 = false). By default, this is set to false (0).

CredentialsCache

```
<CredentialsCache key="activity_credentials">1</
CredentialsCache>
```

Specifies whether a Hybrid App requires credentials (1 = true, 0 = false). Different Hybrid Apps can specify different credentials keys. Hybrid Apps with the same credentials key share that set of credentials. In the case of shared credentials, they are requested only once by the Hybrid App that is launched first.

RequiresActivation

```
<RequiresActivation key="shared_activation_key">1</
RequiresActivation>
```

Specifies whether a Hybrid App requires activation (1 = true, 0 = false). If set to true, the screen defined in the `ActivationScreen` tag is displayed the very first time the Hybrid App is launched, before the default screen is displayed.

If the Activation Screen contains credentials controls (and the Hybrid App requires credentials), the values are updated to the Credentials Cache automatically, without further prompting, with the specified Credentials Screen.

Different Hybrid Apps can specify different activation keys. Hybrid Apps with the same activation key share their activation status. For example, if Hybrid App A and Hybrid App B both specify an activation key of AB (using the key attribute on the `RequiresActivation` tag), when Hybrid App A gets activated, it also activates Hybrid App B so that when Hybrid App B is invoked for the very first time, its activation screen is not seen; it goes directly to the default screen.

TransformPlugin

```
<TransformPlugin> <File/> <Class/> </TransformPlugin>
```

(Optional) If this is defined, the `ResponsePlugin` tag must also be defined. Describes the server module implemented as a .NET assembly that implements the `IMailProcessor` interface. This module is responsible for processing the intercepted e-mail message before it gets delivered to the device.

Inner tags

`<File shared="true">WorkflowClient.dll</File>` The path, including the filename of the assembly that implements the `IMailProcessor` interface. The path is relative to the Hybrid App ZIP package. If the `shared` property is present and set to `true`, the DLL is located in the `<UnwiredPlatform_InstallDir>\Servers\MessagingServer\bin` folder (installed by an external process) and all Hybrid Apps using that DLL will use the same version of the DLL. If the `shared` property is not present, or is present and is set to `false`, each Hybrid App will use its own version of that DLL in the Hybrid App's own folder.

`<Class>Sybase.UnwiredPlatform.WorkflowClient.Transformer</Class>` The .NET Type in the assembly that implements the `IMailProcessor` interface.

ResponsePlugin

`<ResponsePlugin> <File/> <Class/> </ResponsePlugin>`

(Optional) If this is defined, the `TransformPlugin` tag must also be defined. Describes the server module implemented as a .NET assembly that implements the `IResponseProcessor` interface. This module is responsible for processing the response from the device.

Inner tags

`<File shared="true">WorkflowClient.dll</File>` The path, including the filename, of the assembly that implements the `IResponseProcessor` interface. The path is relative to the Hybrid App .zip package. If the `shared` property is present and set to `true`, the DLL is expected to be located in the `<UnwiredPlatform_InstallDir>\Servers\MessagingServer\bin` folder (installed by an external process), and all Hybrid Apps using that DLL will use the same version of the DLL. If the `shared` property is not present, or is present and set to `false`, each Hybrid App will use its own version of that DLL in the Hybrid App's own folder.

`<Class>Sybase.UnwiredPlatform.WorkflowClient.Responder</Class>` The .NET Type in the assembly that implements the `IResponseProcessor` interface.

ClientWorkflows

```
<ClientWorkflows>
  <WindowsMobileProfessional>
    <HTMLWorkflow>
      <File>...</File>
      <HtmlFiles>
        <HtmlFile>...</HtmlFile>
        <HtmlFile>...</HtmlFile>
      </HtmlFiles>
    </HTMLWorkflow>
  </WindowsMobileProfessional>
</ClientWorkflows>
```

```
</HTMLWorkflow>
</WindowsMobileProfessional>
<BlackBerry>
  <HTMLWorkflow>
    <File>...</File>
    <HtmlFiles>
      <HtmlFile>...</HtmlFile>
      <HtmlFile>...</HtmlFile>
    </HtmlFiles>
  </HTMLWorkflow>
</BlackBerry>
<BlackBerry6>
  <HTMLWorkflow>
    <File>...</File>
    <HtmlFiles>
      <HtmlFile>...</HtmlFile>
      <HtmlFile>...</HtmlFile>
    </HtmlFiles>
  </HTMLWorkflow>
</BlackBerry6>
<iPhone>
  <HTMLWorkflow>
    <File>...</File>
    <HtmlFiles>
      <HtmlFile>...</HtmlFile>
      <HtmlFile>...</HtmlFile>
    </HtmlFiles>
  </HTMLWorkflow>
</iPhone>
<Android>
  <HTMLWorkflow>
    <File>...</File>
    <HtmlFiles>
      <HtmlFile>...</HtmlFile>
      <HtmlFile>...</HtmlFile>
    </HtmlFiles>
  </HTMLWorkflow>
</Android>
</ClientWorkflows>
```

This section of the `manifest.xml` file describes the supported device platforms for the Hybrid App and the corresponding client module to use for each platform.

Inner tags

- `<WindowsMobileProfessional>...</WindowsMobileProfessional>` – Windows Mobile Professional device support
- `<iPhone>...</iPhone>` – iOS device support
- `<BlackBerry>...</BlackBerry>` – BlackBerry 5.0 device support
- `<BlackBerry6>...</BlackBerry6>` – BlackBerry 6.0 device support
- `<Android>...</Android>` – Android device support

```
<File>...</File>
```


Contains a reference to an XML file. That XML file should have contents similar to this:

```
<?xml version="1.0" encoding="utf-8"?>
<widget>
  <screens src="html/myAndroidHybridApp.html"
  default="Start_Screen">
    <screen key="html/myAndroidHybridApp.html">
      </screen>
    </screens>
  </widget>
```

The referenced HTML file must be present in the list of `HtmlFiles` tags that follow and must also be present in the Hybrid App .zip package.

```
<HtmlFile>...</HtmlFile>
```

Indicates that the named file (`html/js/API.js`, `html/myAndroidHybridApp.html`) will be used on the specified platform. The referenced file must be present in the Hybrid App .zip package.

ContextVariables

```
<ContextVariables>...</ContextVariables>
```

Describes the collection of context variables that will be made available to the methods in the `IMailProcessor` and `IResponseProcessor` interfaces. When the Hybrid App package is deployed by the administrator, the Display Name that is specified here can be overridden with one of their own choosing.

```
<ContextVariables >
<ContextVariable>
<Name/>
<Value/>
<Certificate/>
<Password/>
</ContextVariable>
```

Describes a context variable that will be made available to the methods in the `IMailProcessor` and `IResponseProcessor` interfaces. When administrators deploy a Hybrid App package, they have the ability to override the value of the context variable that is specified here.

It is good practice for developers of Hybrid Apps to provide sufficient documentation so that administrators can knowledgeably edit a context variable's value as necessary. Context variables are a good place to store configuration information that will likely change between development and production environments.

Inner tags

`<Name>OutputFolder</Name>` The name of the context variable. This is the key used to retrieve the value of the context variable in the methods defined in the `IMailProcessor` and `IResponseProcessor` interface.

Note: The value of the `<Name>` tag supports single-byte characters only.

`<Value>C:\ActivitiesSampleOutput</Value>` The value of the context variable. When administrators deploy a Hybrid App, they have the ability to override the value of the context variable that is specified here.

Note: The value of the `<Value>` tag supports single-byte, double-byte, or both, characters.

`<Certificate>>false</Certificate>` Indicates whether this context variable is a Base64 string representation of an X.509 certificate. If this value is set to true, SAP Control Center displays a dialog specific to selecting an X.509 certificate.

`<Password>>false</Password>` Indicates whether this context variable is a password. If set to true, the value is displayed as asterisks in the SAP Control Center console.

Client Variables

You can define client variables on the server side and retrieve it on the client side by using either native API or JavaScript API. In the JavaScript API, you can call the `hwc.getClientVariables(moduleid, version)` method to retrieve the client variables.

An optional metadata element in `manifest.xml` is used to specify clientvariables information. It has a `version` attribute of integer type to identify and keep track of metadata changes. You can set any positive integer value as the initial version. After the Hybrid App is deployed, each time the metadata gets updated, the version number is increased by one.

```
<Metadata version="1">
  <Item>
    <Name>Key1</Name>
    <Value>Value1</Value>
  </Item>
  <Item>
    <Name>Key2</Name>
    <Value>Value2</Value>
  </Item>
</Metadata>
```

You can update the client variables for a Hybrid App in SAP Control Center, and the change will be pushed to the already deployed clients. The client variables received on the client side are treated as read-only. The client cannot update the client variables.

Similar to server side Hybrid App context variables, client variables are stored as name/value pairs. Both name and value are string type, and the name is case sensitive. The maximum length of the client variable key name is 256 in ANSI code (not Unicode). Although the name is case sensitive, it cannot have the same item names that differ only by case. The metadata item key name cannot be an empty string. The object of a complex type needs to be serialized to string values to set the value.

Note: Due to a limitation on Windows Mobile platforms, the total length of all the client variables (keys and values) cannot exceed 2000 characters.

If the client side variables are updated, the change is applied the next time the Hybrid App is opened.

Similar to context variables, when the Hybrid App package is deployed in SAP Control Center with the option of "Replace," the updated client variables for the old Hybrid App package are not automatically passed to the new Hybrid App package.

MatchRules

`<MatchRules>...</MatchRules>`

Describes the collection of match rules that are used to determine if a message is sent to a TransformPlugin server module for processing. When administrators deploy a Hybrid App, they have the ability to Add, Delete, and override the Match Rules that are specified here.

`<MatchRule>... </MatchRule>` Describes a single match rule.

Note: The value of the `<MatchRule>` tag supports double-byte characters.

Inner tags

`<SubjectRegExp>...</SubjectRegExp>` The value to test for against the "Subject" line of a message.

`<ToRegExp>...</ToRegExp>` The value to test for against the "To" line of a message.

`<FromRegExp>...</FromRegExp>` The value to test for against the "From" line of a message.

`<CCRegExp>...</CCRegExp>` The value to test for against the "CC" line of a message.

`<BodyRegExp>...</BodyRegExp>` The value to test for against the `<Body>` text of a message.

WorkflowClient.xml File

The `WorkflowClient.xml` file contains metadata that specifies how to map the data in the Hybrid App message to and from calls to Mobile Business Object (MBO) operations and object queries.

WorkflowClient.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Workflow xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="WorkflowClient.xsd" >
  <Globals>
    <DefaultScreens activation="..." credentials="..." />
  </Globals>
  <Triggers>
    <Actions>
      <Action name="..." sourcescreen="..." targetscreen="..."
errorscreen="...">
        <Methods>
          <Method type="replay" mbo="..." package="..." >
```

```

        <InputBinding optype="..." opname="..."
generateOld="...">
        <Value sourceType="..." workflowKey="..." paramName="..."
mboType="..."/>
        <Value sourceType="..." workflowKey="..."
relationshipName="..." mboType="list">
        <InputBinding optype="delete" opname="..." generateOld="...">
        <Value sourceType="..." workflowKey="..." paramName="..."
attribName="..." mboType="..."/>
        </InputBinding>
        <InputBinding optype="update" opname="..." generateOld="...">
        <Value sourceType="..." workflowKey="..." paramName="..."
attribName="..." mboType="..."/>
        </InputBinding>
        <InputBinding optype="create" opname="..." generateOld="...">
        <Value sourceType="..." workflowKey="..." paramName="..."
attribName="..." mboType="..."/>
        </InputBinding>
        </Value>
        </InputBinding>
        <OutputBinding generateOld="...">
        <Mapping workflowKey="..." workflowType="..." attribName="..."
mboType="..."/>
        <Mapping workflowKey="..." workflowType="list"
mboType="list">
        <Mapping workflowKey="..." workflowType="..." attribName="..."
mboType="..."/>
        </Mapping>
        </OutputBinding>
        </Method>
    </Methods>
</Action>
</Actions>
<Notifications>
    <Notification type="onEmailTriggered"
targetscreen="...">
        <Transformation>
        <Rule type="regex-extract" source="..." workflowKey="..."
workflowType="..." beforeMatch="..." afterMatch="..." format="..."/>
        </Transformation>
        <Methods>
        <Method name="..." type="..." mbo="..." package="...">
        <InputBinding opname="..." optype="...">
        <Value sourceType="..." workflowKey="..." paramName="..."
attribName="..." mboType="..."/>
        </InputBinding>
        <OutputBinding generateOld="...">
        <Mapping workflowKey="..." workflowType="..." attribName="..."
mboType="..."/>
        <Mapping workflowKey="..." workflowType="list"
mboType="list">
        <Mapping workflowKey="..." workflowType="..."
attribName="..." mboType="..."/>
        </Mapping>
        </OutputBinding>
        </Method>
    </Methods>
    </Notification>
</Notifications>

```

```

    </Methods>
  </Notification>
</Notifications>
</Triggers>
</Workflow>

```

Globals

```

<Globals> <DefaultScreens activation="Introduction"
credentials="Authentication"/> </Globals>

```

Describes the global information for the Hybrid App metadata.

Inner tags

`<DefaultScreens activation="..." credentials="..." />` contains two optional attributes—activation and credentials—that allow you to specify the screens to use for activation and credential requests.

Triggers

```

<Triggers> <Actions> ... </Actions> <Notifications> ... </
Notifications> </Triggers>

```

Describes the conditions under which MBO operations and/or object queries run and, where appropriate, what to return to the device.

Inner tags

`<Actions> ... </Actions>` Contains the description for one or more MBO operations and/or object queries to execute when an online request or submit action is received from the client.

`<Notifications> ... </Notifications>` Contains the description of, at most, one way to extract values from an incoming server notification, execute an MBO object query, and send that notification on to the device.

Action

```

<Action name="Online_Request" sourcescreen="Reports_Create"
targetscreens="OnReportsCreateSuccess"
errorscreens="OnReportsCreateFailure"> ... </Action>

```

Describes the conditions under which MBO operations and/or object queries run and, where appropriate, what to return to the device.

Table 1. Attributes

Attribute	Description
name	The name of the action, which typically corresponds to the key of the menuitem that invoked the action.

Attribute	Description
sourcescreen	The screen from where the action was invoked.
targetscreen	This attribute is optional. The screen to which the client will return, by default, if the MBO operation/object query succeeds. If left unspecified, the client application remains on the current screen. This attribute is applicable only to online request actions.
errorscreen	This attribute is optional. The screen to which the client will return, by default, if the MBO operation/object query fails. If left unspecified, the client application remains on the current screen. This attribute is applicable only to online request actions.
<ul style="list-style-type: none"> errorlogskey errorlogmessagekey errorlogmessageaslistkey 	The keys used to fill any error log messages.

Inner tags

`<Methods> ... </Methods>` Contains the description for one or more MBO operations and/or object queries to be executed when this online request or submit action is received from the client.

Method

```
<Method type="replay" mbo="Reports" package="testReports:1.0"> ... </Method>
```

Describes the conditions under which MBO operations and/or object queries run and, where appropriate, what to return to the device.

Table 2. Attributes

Attribute	Description
type	The type of method to invoke. For object queries, this must be search . For operations, it must be replay .
mbo	The name of the mobile business object (MBO).

Attribute	Description
package	The Hybrid App package name and version of the MBO, separated by a colon, for example, <package_name>:<mbo_version>.

Inner tags

<InputBinding> ... </InputBinding> Contains the description of how to map the key values to the parameters of one or more of the MBO operations and/or object queries to be executed when this online request or submit action is received from the client.

<OutputBinding> ... </OutputBinding> Contains the description of how to map the response from the object query to key values.

InputBinding

```
<InputBinding optype="create" opname="create"
generateOld="false"> ... </InputBinding>
```

Contains the MBO operation to invoke and how to map the key values to the parameters of that operation.

Table 3. Attributes

Attribute	Description
optype	The type of MBO operation to invoke. Must be one of these types: <ul style="list-style-type: none"> • none • create • update • delete • other
opname	The name of the MBO operation to invoke.
generatedOld	A boolean that indicates whether or not to generate old value keys.

Inner tags

<Value> ... </Value> Contains the description of where to obtain the parameter values of the MBO operations to be executed when this online request or submit action is received from the client.

Value

```
<Value sourceType="Key"
workflowKey="Reports_type_id_attribKey" attribName="id"
mboType="int"/>
```

Describes how to obtain the parameter value or attribute value from the Hybrid App message.

Table 4. Attributes

Attribute	Description
sourceType	The source of the data. Must be one of these types: <ul style="list-style-type: none"> • Key • BackEndPassword • BackEndUser • DeviceId • DeviceName • DeviceType • UserName • MessageId • ModuleName • ModuleVersion • QueueId • ContextVariable
workflowKey	If the sourceType is Key , the name of the key in the Hybrid App message from which to obtain the value.
contextVariable	If the sourceType is ContextVariable , the name of the context variable from which to obtain the value.
paramName	If present, the name of the parameter the value is supplying.
pkName	If present, the name of the personalization key the value is supplying.
attribName	If present, the name of the attribute name the value is supplying. This value may, or may not, be present together with paramName.
parentMBO	The name of the parent MBO, if any.

Attribute	Description
relationshipName	The name of the relationship, if any.
mboType	<p>The type of the value in MBO terms. Must be one of these types:</p> <ul style="list-style-type: none"> • string • char • date • datetime • time • int • byte • short • long • decimal • boolean • binary • float • double • list • integer • structure
array	A boolean that indicates whether or not the value is an array. The default is false.
length	The length of the parameter/attribute/personalization key.
precision	The precision of the parameter/attribute/personalization key.
scale	The scale of the parameter/attribute/personalization key.
convertToLocalTime	A boolean that indicates whether or not to convert the value to a local time before passing it to the MBO. The default is false.

Inner tags

`<InputBinding> ... </InputBinding>` If the `mboType` is “list,” it will be necessary to specify child input bindings to indicate which MBO operations to invoke when a child is updated, deleted, or created.

OutputBinding

```
<OutputBinding generateOld="true"> ... </OutputBinding>
```

Contains a series of mappings that indicate how to map the results of the object query to the Hybrid App message.

Table 5. Attributes

Attribute	Description
generatedOld	A boolean that indicates whether or not to generate old value keys.

Inner tags

<Mapping> ... </Mapping> Contains the description of how to map the results of the object query to a key in the Hybrid App message.

Mapping

```
<Mapping workflowKey="Department_dept_id_attriKey"
workflowType="number" attriName="dept_id" mboType="int"/>
```

Describes how to fill a key’s value in the Hybrid App message from the results of the object query.

Table 6. Attributes

Attribute	Description
workflowKey	The name of the key in the Hybrid App message to fill with the results of the object query.
workflowType	The type of the data in the Hybrid App message. Must be one of these types: <ul style="list-style-type: none"> • text • number • boolean • datetime • date • time • list • choice
attriName	If present, the name of the attribute name to which the key is mapped.

Attribute	Description
hardCodedValue	If the workflowType is not choice, and attrib-Name is not present, the hard-coded value to which the key is mapped.
keyWorkflowKey	If the workflowType is choice, the key to which to map the dynamic display names of the choice.
valueWorkflowKey	If the workflowType is choice, the key to which to map the dynamic values of the choice.
assumeLocalTime	A boolean to indicate whether or not to assume that the values coming back from the object query are in local time or not. The default is false.
array	A boolean that indicates whether or not the value is an array. The default is false.
mboType	<p>The type of the value in MBO terms. Must be one of these types:</p> <ul style="list-style-type: none"> • string • char • date • datetime • time • int • byte • short • long • decimal • boolean • binary • float • double • list • integer • structure
relationshipName	The name of the relationship, if any.

Inner tags

`<Mapping> ... </Mapping>` If the mboType is list, you must specify child mappings to indicate how to map the attributes of child MBO instances to keys in the Hybrid App message.

Notification

```
<Notification type="onEmailTriggered" targetscreen="dept"> ...
</Notification>
```

Describes how to formulate the Hybrid App message for the given notification type and which screen to open on the device when that Hybrid App message is opened.

Table 7. Attributes

Attribute	Description
type	The type of the notification. Must be onEmail-Triggered.
targetscreen	The screen to which the client will be opened if the object query succeeds.
errorscreen	The screen to which the client will be opened, by default, if the object query fails.
<ul style="list-style-type: none"> • errorlogskey • errorlogmessagekey • errorlogmessageaslistkey 	The keys to use to fill any error log messages.

Inner tags

<Transformation> ... </Transformation> Contains the description for one or more rules that dictate how to extract values from the server notification and map it to a key in the Hybrid App message.

<Methods> ... </Methods> Contains the description for one or more object queries to be executed when this online request or submit action is received from the client.

Rule

```
<Rule type="regex-extract" source="subject" workflowKey="ID"
workflowType="number" beforeMatch="Purchase order request \(("
afterMatch="\) is ready for review" format=""/>
```

Describes how to extract a value from the server notification and map it to a key in the Hybrid App message.

Table 8. Attributes

Attribute	Description
type	The type of the rule. Must be regex-extract .

Attribute	Description
source	<p>The source of the data to be extracted. Must be one of these sources:</p> <ul style="list-style-type: none"> • body • subject • from • to • cc • receivedDate • custom1, custom2, custom3, custom4, custom5, custom6, custom7, custom8, custom9, or custom10
workflowKey	<p>The name of the key in the Hybrid App message to fill with the value extracted from the server notification.</p>
workflowType	<p>The type of the data in the Hybrid App message. Must be one of these data types:</p> <ul style="list-style-type: none"> • text • number • boolean • datetime • date • time • list • choice
assumeLocalTime	<p>A boolean to indicate whether or not to assume that the values coming back from the object query are in local time or not. The default is false.</p>
beforeMatch	<p>A regular expression used to indicate where the value starts.</p>
afterMatch	<p>A regular expression used to indicate where the value ends.</p>
format	<p>If the workflowType is datetime or time, the C# formatting string to be passed to DateTime.ParseExact when converting the value to a datetime.</p>

The Look and Feel XML Files

Each device platform (WindowsMobile Professional, BlackBerry, BlackBerry6, iOS, and Android) provides a `<File>...</File>` tag, which refers to an XML file in the Hybrid App ZIP package.

The contents are similar to this:

```
<?xml version="1.0" encoding="utf-8"?>
<widget>
  <screens src="html/myAndroidhybridapp.html"
default="Start_Screen">
    <screen key="html/myAndroidhybridapp.html">
    </screen>
  </screens>
</widget>
```

Different platforms can share the same look and feel XML file, or they can use different XML files, depending on the application design. Different XML files can refer to the same HTML file, or to different HTML files, depending, again, on the application design.

When a Hybrid App package is generated using the Hybrid App Designer, with the **Optimized for appearance** option selected in Preferences, three look and feel XML files are generated: `hybridapp.xml`, `hybridapp_Custom.xml`, and `hybridapp_JQM.xml`.

For iOS Hybrid Apps that do not fully display buttons on the bottom of the screen, use the `<AdjustWebviewFrame>1</AdjustWebviewFrame>` element after the `</screens>` entry. For example:

```
<?xml version="1.0" encoding="utf-8"?>
<widget>
  <screens src="html/NonDefaultMetadataTest.html" default="" >
    <screen key="html/NonDefaultMetadataTest.html">
    </screen>
  </screens>
  <AdjustWebviewFrame>1</AdjustWebviewFrame>
  <zoom>0</zoom>
</widget>
```

Using Third-party Files

To load external JavaScript and CSS files dynamically when creating a Hybrid App package manually:

Add the path of the third-party JavaScript or CSS files to the `manifest.xml` file, in the device platform section. For example:

```
<BlackBerry>
<HTMLWorkflow>
<File>TokenSI_CustomLookAndFeel.xml</File>
<HtmlFiles>
<HtmlFile>html/css/bb/some-3rd-part.css</HtmlFile>
<HtmlFile>html/css/bb/checkbox.css</HtmlFile>
```

```

<HtmlFile>html/css/bb/datepicker.css</HtmlFile>
<HtmlFile>html/css/bb/editBox.css</HtmlFile>
<HtmlFile>html/css/bb/img/btn_check_off.png</HtmlFile>
<HtmlFile>html/css/bb/img/btn_check_on.png</HtmlFile>
<HtmlFile>html/css/bb/img/btn_radio_off.png</HtmlFile>

```

Deploying a Hybrid App Package with the Deploy Wizard

Use the Deploy wizard to make Hybrid App packages available on SAP Mobile Server.

If you are deploying to a target domain, replicate the value in the context variable. The domain deployment target must match the context variable defined. If the developer has used an incorrect context variable (for example, one used for testing environments), you can change the value assigned to one that is appropriate for production deployments.

1. In the left navigation pane of SAP Control Center, click **Hybrid Apps**.
2. From the **General** tab, click **Deploy**.
3. Click **Browse** to locate the Hybrid App package.
4. Select the file to upload and click **Open**.
5. Select the deployment mode:
 - **New** – deploys an SAP Mobile Server package and its files for the first time.
 - If the uploaded file does not contain an SAP Mobile Server, or an SAP Mobile Server with the same name and version is already deployed to SAP Mobile Server, you see an error message.
 - **Update** – installs a new SAP Mobile Server package with the original package name and assigns a new, higher version number than the existing installed SAP Mobile Server package. SAP recommends that you use this deployment mode for major new changes to the SAP Mobile Server package.
 - During the update operation, SAP Mobile Server:
 - Acquires a list of assigned application connections from the original package.
 - Installs and assigns the package a new version number.
 - Prompts the administrator to specify application connection assignments from the acquired list of assigned application connections.
 - Preserves existing notifications.
 - Preserves the previous SAP Mobile Server package version.
 - **Replace** – replaces an existing SAP Mobile Server package with a new package, but maintains the same name and version. SAP recommends that you use the replace deployment mode for minor changes and updates to the SAP Mobile Server package, or during initial development.
 - During the replace operation, SAP Mobile Server:
 - Acquires a list of assigned application connections for each user of the original package.

Develop Hybrid Apps Using Third-party Web Frameworks

- Uninstalls the original package.
- Installs the new package with the same name and version.
- Assigns the original application connections list to the new package, thus preserving any application connection assignments associated with the original package.

The package is added to the list of deployed packages, which are sorted by Display Name.

Next

Configure the deployed package if you want it to have a different set of properties in a production environment.

Develop a Hybrid App Using the Hybrid App Designer

Hybrid Apps support the occasionally connected user and addresses the replication and synchronization issues those users present for the back-end system.

A Hybrid App application requires an integration module on the server side, which is implemented by a static set of logic that processes Hybrid App-specific metadata to map keys to and from mobile business object attributes, personalization keys, and parameters. This integration module processes the notifications identified by matching rules configured for the server-initiated starting point and also processes the responses sent to the server from the device.

You can develop Hybrid Apps that work on these platforms:

- Android
- Apple iOS
- BlackBerry
- Windows Mobile Professional

The Hybrid App Designer provides UI controls that make development of Hybrid Apps fast and easy. For information about using the Hybrid App Designer to design and develop Hybrid Apps, see online help, *SAP Mobile WorkSpace - Hybrid App Package Development*.

See *Supported Hardware and Software* for supported version levels.


Deploy the Hybrid App Package to SAP Mobile Server

Use the Hybrid App generation wizard to generate the Hybrid App package and deploy it to SAP Mobile Server to make it available for device clients.

Generating Hybrid App Files and Deploying a Package

Use the Hybrid App Package Generation wizard to generate a Hybrid App package, or to generate Hybrid App files that you can deploy to specific devices.

You cannot run Hybrid Apps created in a newer version of the Hybrid App Designer on an older version of the Hybrid Web Container. The newly generated code could call native functionality that previous Hybrid Web Containers are unaware of.

1. In the Hybrid App Designer, click the  code generation icon on the toolbar.
Alternatively, right-click in the Flow Design or Screen Design page and select **Generate Hybrid App Package**.

Develop a Hybrid App Using the Hybrid App Designer

2. Specify the Mobile Server profile.
3. Choose the option to either generate a package or generate files for one or more specific platforms. Specify the required parameters, and click **Finish** to generate the files and close the wizard.

Note: The files to be generated are listed in the File Order tab of the Flow Design properties view for the application. You can optionally add or remove files or change the order in which they are loaded in the running application. See *Flow Design Application Properties* for more information.

The generated files are created in your project, visible in Workspace Navigator under Generated Hybrid App.

4. Deploy the Hybrid App to an appropriate device or simulator.

See the *Developer Guide: Hybrid Apps* for information about how to configure devices or simulators for the Hybrid App Package.

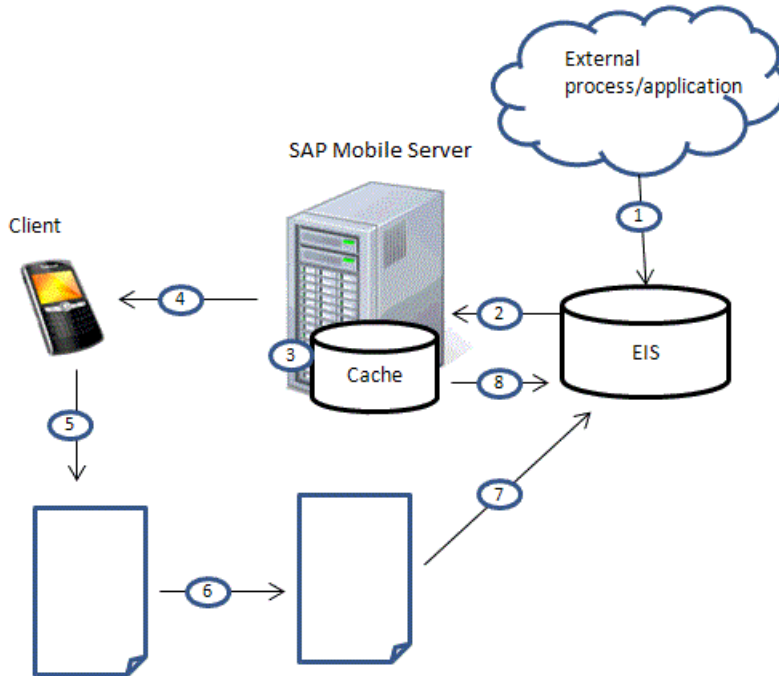
See *SAP Control Center for SAP Mobile Platform* documentation for information about managing devices, Hybrid App assignments, and users.

Hybrid App Patterns

The Hybrid Web Container allows you to create lightweight applications that implement various business solutions. These are some of the primary Hybrid App and the SAP Mobile Platform patterns (models):

- Online lookup – the client retrieves data directly from the EIS. This pattern typically uses a client-initiated starting point.
- Server notification – the enterprise information system (EIS) notifies SAP Mobile Platform of data changes and SAP Mobile Platform sends notifications to subscribed devices based on the rules.
- Cached data – the client retrieves data from the SAP Mobile Server cache. This pattern typically uses a client-initiated starting point.

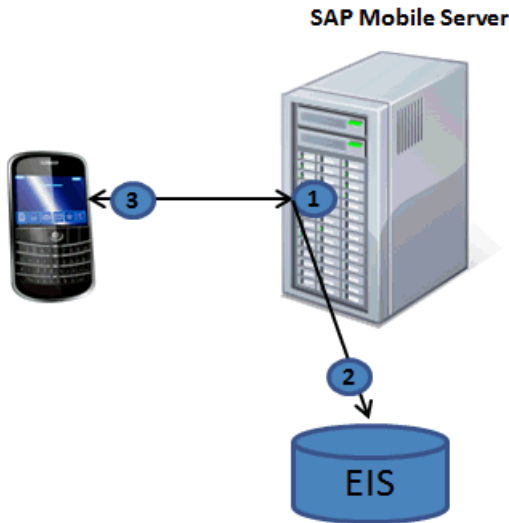
These patterns are not mutually exclusive. You can create applications that combine patterns in various ways to meet business needs. For example:



1. An external process or application updates EIS data.
2. The changed data triggers a data change notification (DCN), which is sent to SAP Mobile Server, or a message from another client updates mobile business object (MBO) data contained on SAP Mobile Server.
3. The DCN could be programmed to update MBO data.
4. SAP Mobile Server notifies the client that some action needs to be taken.
5. The client views the message.
6. The client opens a screen to perform the required action. The form may, for example, call an object query to return cached data or online data, call an MBO operation, or perform some other action.
7. The client sends an update to SAP Mobile Server.
8. SAP Mobile Server updates the EIS.

Online Lookup

This pattern provides direct interaction between the data requester (client) and the enterprise information system (EIS), supplying real-time EIS data or cached data.



While the server notification and cached data patterns are flexible regarding MBO definition and cache group policy, the online lookup pattern must have at least one `findByParameter` and use the Online cache group policy:

1. The client requests data using the `findByParameter` object query.
2. Since the MBO associated with the object query is in a cache group that uses an Online policy, SAP Mobile Server retrieves the requested data directly from the EIS and not the cache.
3. Online data is returned to the client.

In this example, online data retrieval by the client is triggered when the user selects the menu item that calls the `findByParameter` object query.

Implementing Online Lookup for Hybrid Apps

Define an MBO with at least one load argument that maps to a propagate-to attribute, add the MBO to a cache group that uses an Online policy, then define the Hybrid App that calls the `findByParameter` object query to return real-time results from the EIS.

Defining Load Arguments from Mapped Propagate to Attributes

Create an MBO with at least one load argument, map as propagate to attributes, then assign the MBO to a cache group that uses an Online policy.

1. From SAP Mobile WorkSpace, create an MBO that has at least one load argument. For example, you could define an Employee MBO as:

```
SELECT emp_id, emp_fname, emp_lname, dept_id
FROM sampledb.dba.employee WHERE dept_id = :deptIdLP
```

2. In the MBO Properties view, select the **Attributes > Load Arguments** tab, map each load argument to be used as an operation load argument for the Hybrid App package to a Propagate to Attribute. This example requires you to map the deptIdLP load argument to the empDeptId attribute. You must also verify that data types are INT and the default value is a valid INT.
3. Set the Online cache group policy for the MBO.
 - a) Add the MBO to a cache group that uses the Online cache group policy. For example, create a new cache group named CacheGroupOnline and set the policy to **Online**.
 - b) Drag and drop the MBO to CacheGroupOnline.

The findByParameter object query is automatically generated based on all load arguments that have propagate-to attributes:
4. Deploy the project that contains the MBO to SAP Mobile Server.

Binding the findByParameter Object Query to a Menu Action

For synchronous, online data access, define an Online Request menu action and bind it to the findByParameter object query.

Prerequisites

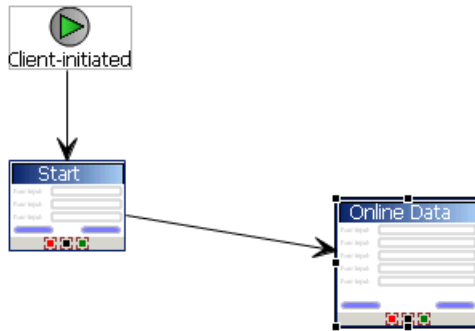
You must have propagate-to attributes mapped to MBO load parameters, and the deployed MBO must use an Online cache group policy. SAP Mobile Platform services must be running.

Task

1. From SAP Mobile WorkSpace, launch the Hybrid App Designer.
2. From the Flow Design screen, double-click the screen for which you are defining a mapping to open it in the Screen Design tab.

For example, you can have a client-initiated starting point with a Start screen that connects to the Online Data screen.

Develop a Hybrid App Using the Hybrid App Designer



3. Highlight the menu item you want to map, or create a new menu item.
4. Define a Submit action that invokes the `findByParameter` object query:
 - a) From the General tab, select **Online Request** as the Type.
 - b) In the Details section, select **Search** to locate the MBO that contains the `findByParameter` object query.
 - c) Click the **General** tab, select **Invoke object query** and select **`findByParameter`**.

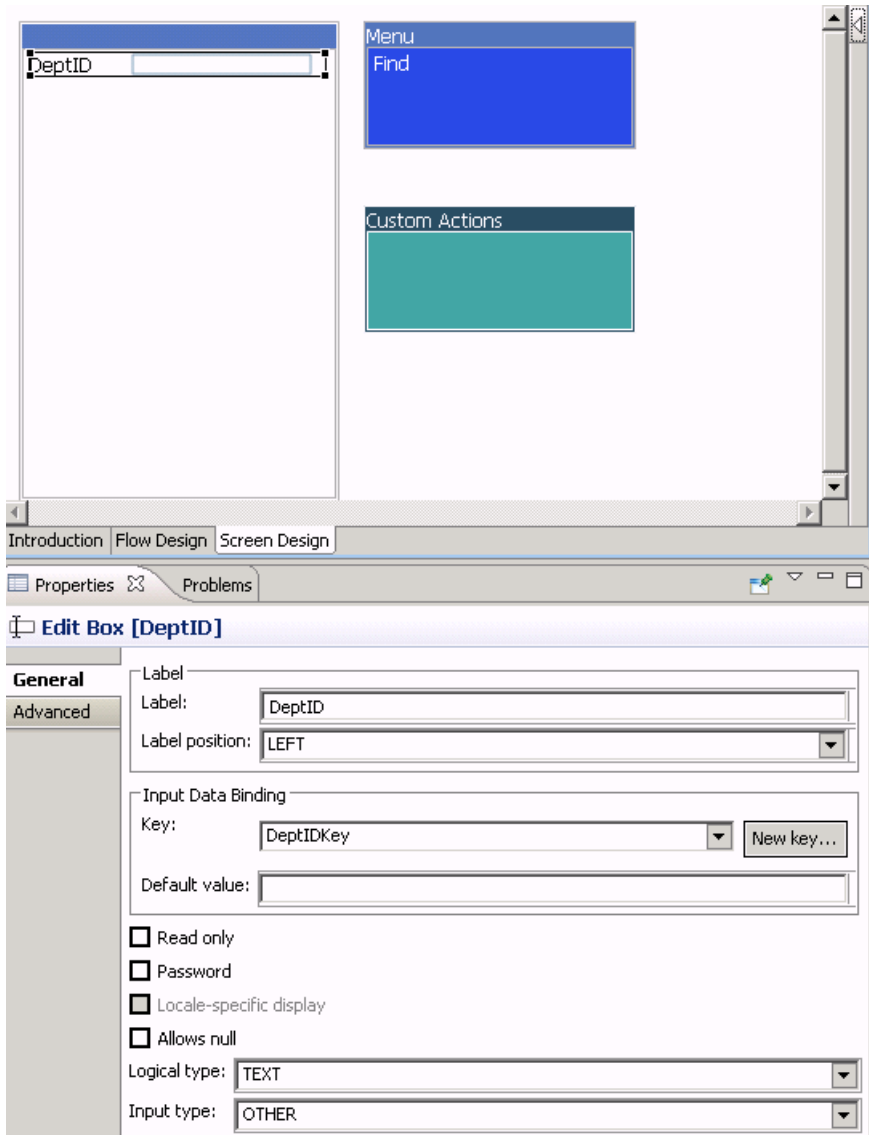
If you select the Parameter Mappings tab, you see all the load parameters defined for the MBO and used to generate the `findByParameter` object query. In addition to Key, you can map parameters to `BackEndPassword`, `BackEndUser`, `DeviceId`, `DeviceName`, `DeviceType`, `UserName`, `MessageId`, `ModuleName`, `ModuleVersion`, and `QueueId`.

Unmapped parameters can get their value from the default value, if specified, or from the personalization key value they are mapped to, if that is specified. If the key is unmapped, and the parameter has no default value and is not mapped to a personalization key value, the parameter value is empty (NULL for string, 0 for numeric, and so on).

Defining the Control that Contains the `findByParameter` Object Query Parameter

Add a control to pass the load argument to SAP Mobile Server. Define a screen that displays the results returned from the EIS.

1. Define a control that passes the load argument to SAP Mobile Server from the screen (named Online Data) that contains the menu item (named Find) that invokes the `findByParameter` object query:
 - a) Select an **EditBox** control and click in the control area.
 - b) Name the EditBox `DeptId`.
 - c) From the Properties view, select **New key** and name it `DeptIdKey`. Click **OK**.

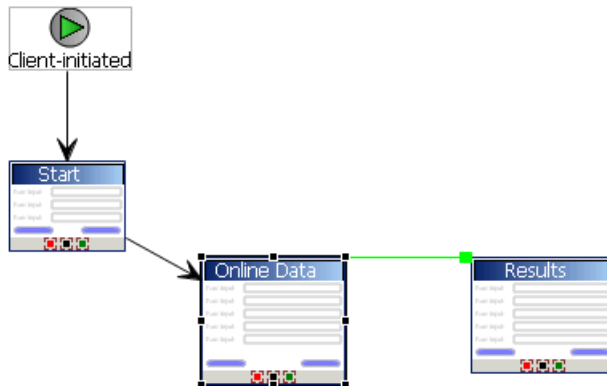


2. Select the **Find** menu item, and from the Parameter Mappings tab, map parameters to input keys defined for the controls. For example, map the deptIDLp parameter to the DeptIdKey key.
3. Define a screen that displays the results of the findByParameter object query:
 - a) From the Flow Design window, add a new Screen and name it Results. Select the Screen Design tab.
 - b) Drag and drop a **Listview** control onto the control area.

Develop a Hybrid App Using the Hybrid App Designer

- c) Select the Flow Design tab and double-click the **Online Data** screen to open it.
- d) Select the **Find** menu item, and in the Properties view, specify **Results** as the success screen.

The Online Data screen now sends successful results returned by the EIS to the Results screen. The Flow Design window indicates the connection between the screens.



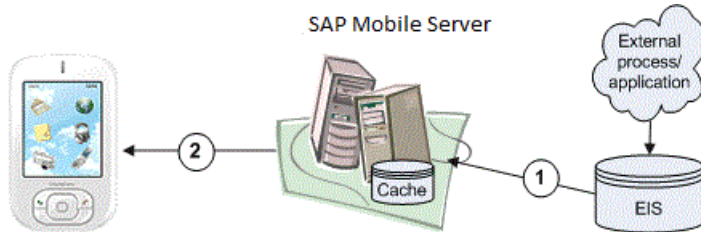
4. Configure the Results screen to display the results. In this example, the Emp MBO, contains three attributes: Id, empName, and empDeptId. Create a Listview with a cell for each attribute to display the results returned from the EIS as a list:
 - a) From the Flow Design window, double-click the **Results** screen to display it in the Screen Design window.
 - b) Select the control area, select the General tab in the Properties view, and for the Input Data Binding Key select <MBOName> (where MBOName is the name of the MBO).
 - c) Select the **Cell** tab, then click **Add** to add cell line 0.
 - d) Select **Add** in the "Fields for cell line 0" section, then select the **Emp_id_attribKey** key. Click **OK**.

This maps cell line 0 with the id attribute for the Emp MBO results returned by the object query.
 - e) Repeat steps 3 and 4 again for the remaining two attributes.
5. Select the **Problems** view, and verify there are no errors.

You now have a deployable Hybrid App package that passes the DeptID value to the findByParameter object query which returns matching EIS results and displays them in the Results screen.

Server Notification

Configure matching rules for MBO-related data on SAP Mobile Server. Any data changes matching these rules trigger a notification from SAP Mobile Server to the client.



1. MBO data is updated from the EIS, by an external process or application that updates EIS data and triggers a data change notification (DCN), or a scheduled data refresh.
2. If matching rules that correspond to the notification message fields are configured for the MBO and Hybrid App package, SAP Mobile Server sends a notification to the client.

Implementing Server Notification for Hybrid Apps

Set up SAP Mobile Server to send notifications to Hybrid Apps when matching rules are encountered.

Defining the Mobile Business Object for Server Notification

The server notification pattern supports any number of MBO definitions. For this example, create an MBO with one load argument, assign the load argument a propagate-to attribute value, then assign the MBO to a cache group that uses an Online policy.

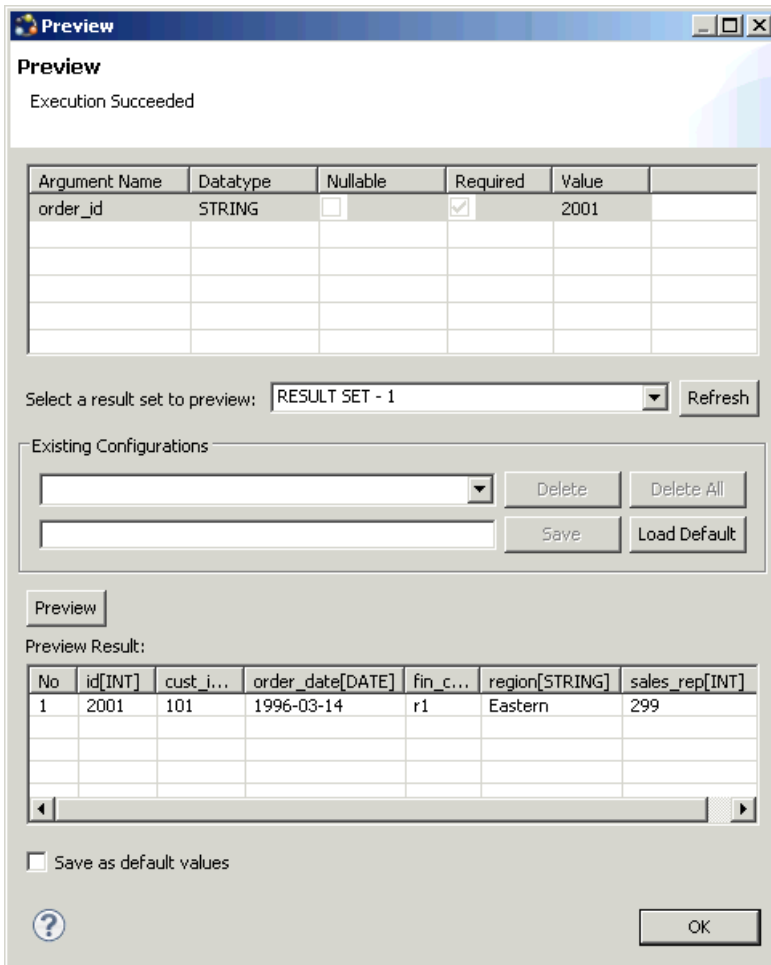
The MBO definition described here allows retrieval of online results by the Hybrid App to which the MBO belongs.

1. In SAP Mobile WorkSpace, create an MBO from the sampledb database that has at least one load argument. For example, you could define a Sales_order MBO as:

```
SELECT  id,
        cust_id,
        order_date,
        fin_code_id,
        region FROM sampledb.dba.sales_order
WHERE id = :order_id
```

2. Preview the MBO by selecting **Preview** from the Definition tab. Enter 2001 as the value. The preview returns one row from the sales_order table based on the id attribute (2001).

Develop a Hybrid App Using the Hybrid App Designer



3. In the MBO Properties view, click the **Load Arguments** tab, select the **id** attribute as the Propagate to attribute that maps to the order_id load argument. Change the datatype to INT, and include an integer value for the data source default value.
4. Set the Online cache group policy for the MBO.
 - a) Add the MBO to a cache group that uses the Online cache group policy. For example, create a new cache group named CacheGroupOnline and set the policy to **Online**.
 - b) Drag and drop the MBO to CacheGroupOnline.

The findByParameter object query is automatically generated based on the order_id load argument:

```
SELECT x.* FROM Sales_order x WHERE x.id = :order_id
```

5. Deploy the project that contains the MBO to SAP Mobile Server.

Creating the Server-Driven Notification Starting Point

Create a new Hybrid App with a server-initiated starting point.

1. From SAP Mobile WorkSpace, select **File > New > Hybrid App Designer**.
2. Select the folder that contains the Sales_order MBO as the parent folder, name the file Sales_order.xbw, and click **Next**.
3. In the Starting Points screen, select **Responds to server-driven notifications**, and click **Next**.
4. Configure the starting point:
 - a) In the Select a Mobile Business Object and Object Query screen, select **Search**.
 - b) Select the project that contains the Sales_order MBO and select **Search**. Select the **Sales_order** MBO and select **OK**.
 - c) Select the **findByParameter** object query.
The order_id parameter appears in the Parameters field. Click **Next**.
 - d) Specify a sample notification. Enter Order (2001) created in the Subject line. Click **Next**.
 - e) Click and drag to select "Order (", while this phrase is highlighted, right-click and select **Select as Matching Rule**.
 - f) Click **Next**. Select **order_id**. In the Extraction Rule Properties:
 1. Select **Subject** as the field.
 2. Select "Order (" as the Start tag.
 3. Select ") created" as the End tag.

When the notification is sent to the client, the sample value (2001 in this example), is replaced with the order_id key, which identifies the id attribute of the object query. The Hybrid App the client receives is populated with values returned by the findByParameter object query.

Develop a Hybrid App Using the Hybrid App Designer

Identify Parameter Values
For each parameter of the MBO operation, specify where to find the data in the incoming message.

Extraction Rules

Key	Field	Start Tag	End Tag	Format	Sample Value	Type
order_id	subject	Order {()} created		2001	int

Extraction Rule Properties

Field: Subject

Start tag: Order {(

End tag:)} created

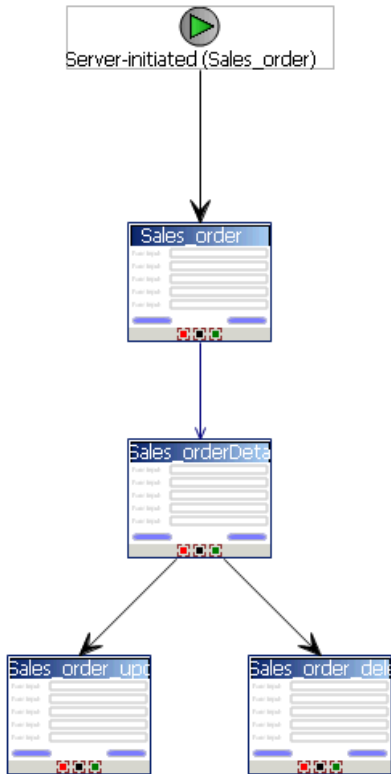
Format:

Sample notification field contents:
Order (2001) created

Value extracted from sample notification field contents:
2001

< Back Next > Finish Cancel

5. Click **Finish** to create default screens and starting points.
Screens are populated with menu items and controls based on the MBO definition.



6. Deploy the Hybrid App package to SAP Mobile Server.

Sending an Order Notification to the Device

Use the "Send a notification" option to send a message to the registered user, which tests the server notification process.

Prerequisites

Before sending notification to the client, you must:

1. Register the Hybrid App connection in SAP Control Center.
2. Download and configure the Hybrid Web Container on the device or emulator.

Task

Use this method only for testing purposes, during development. In a production system, notifications would come in as DCN, or e-mail-based notifications.

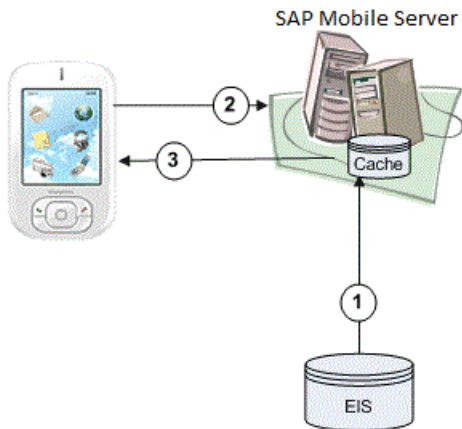
Develop a Hybrid App Using the Hybrid App Designer

1. In the Flow Design of the Hybrid App Designer, right-click and select **Send a notification**.
2. Select **Get Device Users**, and set the "To" field to **User1**, or whatever device user is registered in SAP Control Center and assigned to the Hybrid App package.
3. In the Subject field, enter a sales order that meets the matching rules criteria defined for the Sales_order Hybrid App. For example:
Order (2001) created
4. Click **Send**.

The message is sent to the device. The number 2001 in the notification identifies and returns row 2001 (the findByParameter object query parameter).

Cached Data

This pattern is efficient when access to cached data is sufficient to meet business needs. For example, it may be sufficient to refresh the cache once a day for noncritical MBO data that changes infrequently.



1. EIS data is cached based on the MBO cache policy (Scheduled or On demand). Either policy lets you define the length of time for which cached data is valid.
2. The Hybrid App requests data through an object query.
3. Cached data is returned to the client if it is within the cache policy's specified cache interval.

Implementing the Cached Data Pattern

Define an MBO that uses either a Scheduled or On demand cache group policy to allow the Hybrid App to which it belongs to retrieve cached data.

Defining the Mobile Business Object

Create an MBO with the required attributes, assign the MBO to a cache group that uses a scheduled policy, and define an object query that returns the results from the SAP Mobile Server cache (also called the CDB) to the client.

This example defines an MBO that retrieves employee benefit information for all employees of a given department based on the dept_id attribute using the findByDeptId object query.

1. From SAP Mobile WorkSpace, create an MBO. For example, you could define the employee MBO as:

```
SELECT emp_id,  
       emp_fname,  
       emp_lname,  
       dept_id,  
       bene_health_ins,  
       bene_life_ins,  
       bene_day_care  
FROM sampledb.dba.employee
```

2. Set the cache group policy for the MBO:
 - a) Create a new cache group named CacheGroupScheduled and set the policy to **Scheduled**. Set the **Cache interval** to 24 hours, so the cache is refreshed once a day.
 - b) Drag and drop the MBO to CacheGroupScheduled.
3. Define an object query for the MBO that retrieves employee information based on the dept_id attribute. For example, define the findByDeptId object query as:

```
SELECT x.* FROM Employee x  
WHERE x.dept_id = :deptIDLp
```

Develop a Hybrid App Using the Hybrid App Designer

Object Query 'findByDeptId'

Edit Object Query
Edit the object query

Name:

Comment:

Parameters:

Name	Datatype	Nullable	Mapped to
deptIDL	INT	<input type="checkbox"/>	dept_id

Query Definition

```
SELECT x.* FROM Employee x
WHERE x.dept_id = :deptIDL
```

Create an index

Return Type:

Return a single object Return multiple objects Return a result set

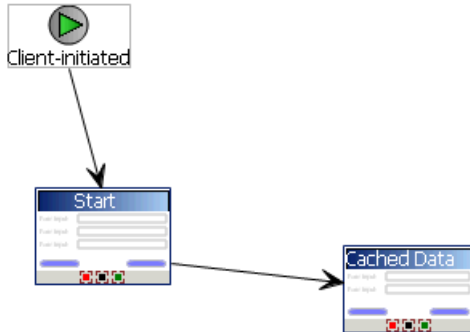
4. Deploy the project that contains the MBO to SAP Mobile Server.

Binding the findByDeptId Object Query to a Menu Action

For access to cached data, define a menu action and bind it to the findByDeptId object query.

1. From SAP Mobile WorkSpace, launch the Hybrid App Designer.
2. From the Flow Design screen, double-click the screen for which you are defining a mapping to open it in the Screen Design tab.

For example, you can have a client-initiated starting point with a Start screen that connects to the Cached Data screen.



3. Highlight the menu item you want to map, or create a new menu item.
4. Define a Submit action named FindBenefitsInfo that invokes the findByDeptId object query:
 - a) In the Properties view, in the General properties for the selected menu item, select **Online Request** as the Type.
 - b) In the Details section, select **Search** to locate the MBO that contains the findByDeptId object query.
 - c) Click the **General** tab, select **Invoke object query** and select **findByDeptId**.

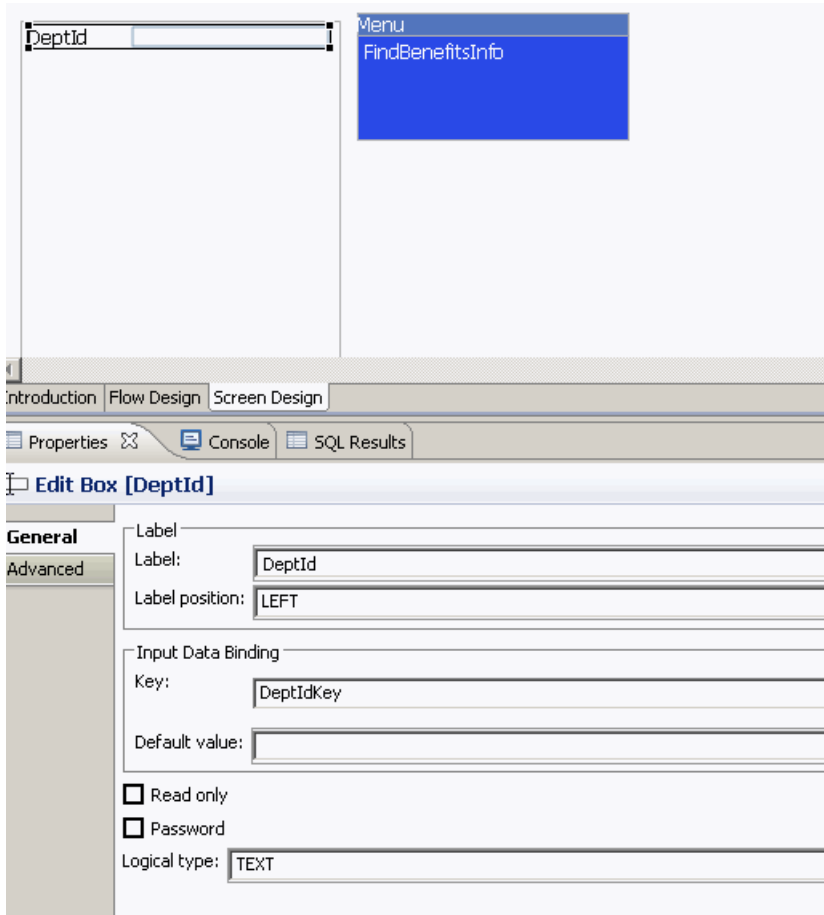
If you select the Parameter Mappings tab, you see the parameters associated with the object query (findByDeptId). Map this parameter to a key.

Defining the Control that Contains the findByDeptId Object Query Parameter

Add a control to pass the object query parameter to SAP Mobile Server. Define a screen that displays the results returned from the SAP Mobile Server cache.

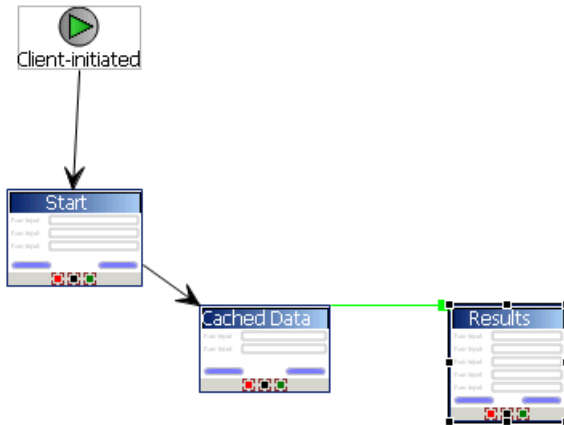
1. Define a control that passes the object query parameter to SAP Mobile Server from the screen (named Cached Data) that contains the menu item (named FindBenefitsInfo) that invokes the findByDeptId object query:
 - a) Select an **EditBox** control and click in the control area.
 - b) Name the EditBox DeptId.
 - c) From the Properties view, select **New key** and name it DeptIdKey. Click **OK**.

Develop a Hybrid App Using the Hybrid App Designer



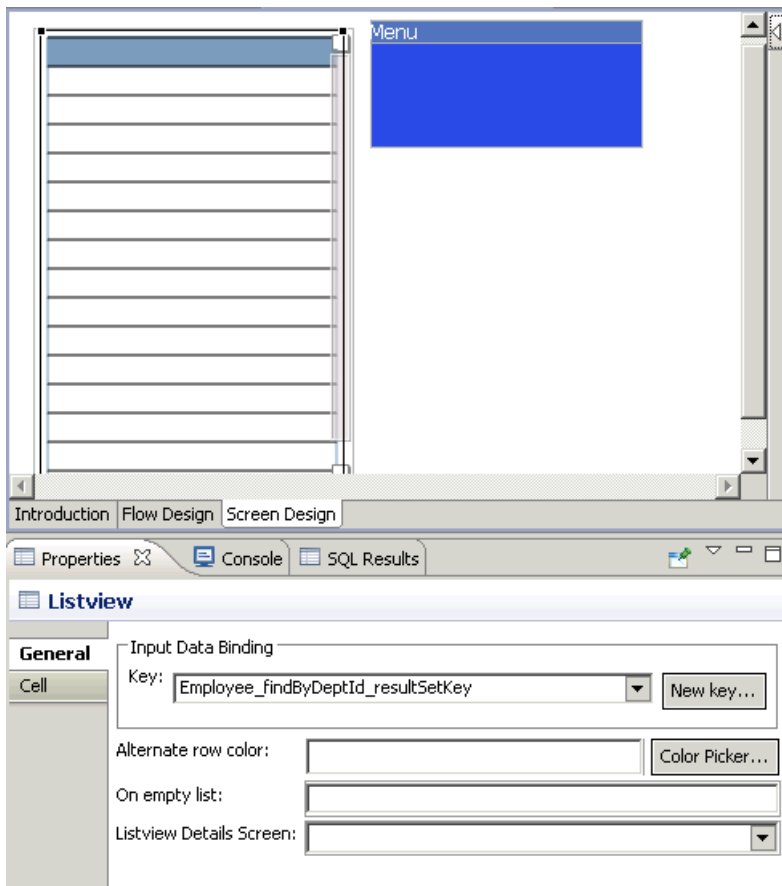
2. Select the FindBenefitsInfo menu item, and from the Parameter Mappings tab, map parameters to input keys defined for the controls. For example, map the deptIDL parameter to the DeptIdKey key.
3. Define a screen that displays the results of the findByDeptId object query:
 - a) From the Flow Design window, add a new Screen and name it Results. Select the Screen Design tab.
 - b) Drag and drop a **Listview** control onto the control area.
 - c) Select the Flow Design tab and double-click the **Cached Data** screen to open it.
 - d) Select the **FindBenefitsInfo** menu item, and in the Properties view, in General properties, select **Online Request** as the Type and in the Details section, select **Results** as the Success screen.

The Cached Data screen now sends successful results returned by the SAP Mobile Server cache to the Results screen. The Flow Design window indicates the connection between the screens.

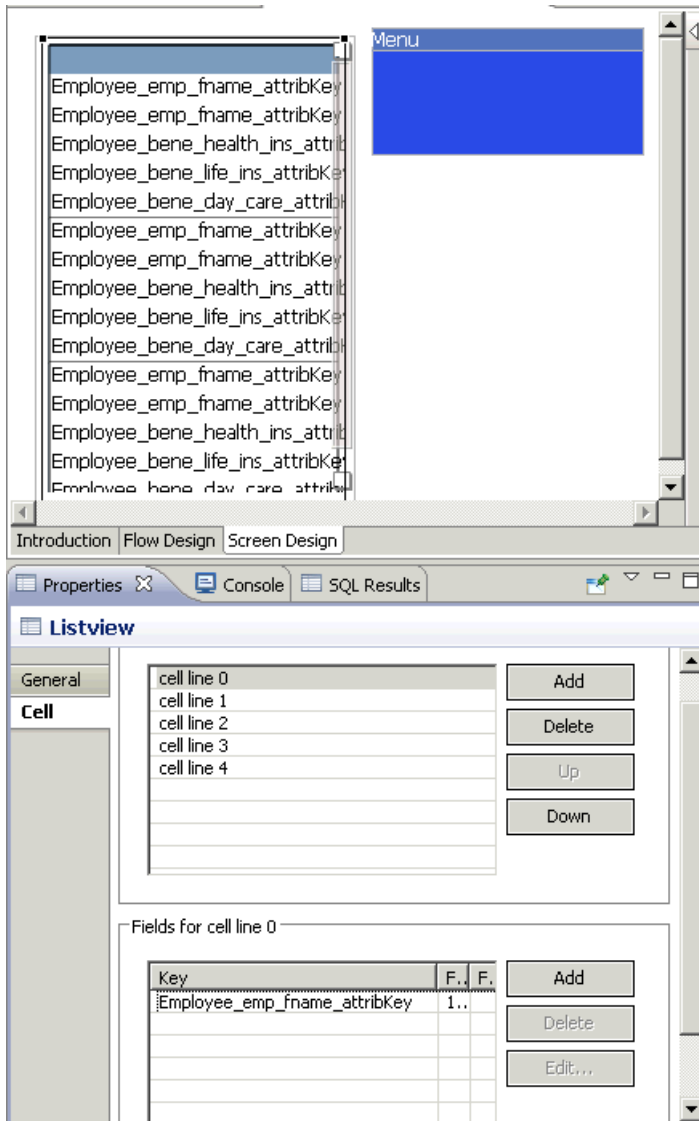


4. Configure the Results screen to display the results. In this example, the Employee MBO, contains seven attributes that identify the employee and their benefits. Create a Listview with a cell for each attribute to display the results returned from the cache as a list:
 - a) From the Flow Design window, double-click the **Results** screen to display it in the Screen Design window.
 - b) Select the control area, select the **General** tab in the Properties view, and for the Input Data Binding Key select **MBOName_findByDeptId_resultSetkey** (where MBOName is the name of the MBO).

Develop a Hybrid App Using the Hybrid App Designer



- c) Select the **Cell** tab, then click **Add** to add cell line 0.
- d) Select **Add** in the "Fields for cell line 0" section, then select the **Employee_emp_fname_attribKey** key. Click **OK**.
This maps cell line 0 with the id attribute for the Emp MBO results returned by the object query.
- e) Repeat steps 3 and 4 again for the remaining employee's last name and benefits related attributes.



5. Select the **Problems** view, and verify there are no errors.

You now have a deployable Hybrid Apppackage that passes the DeptID value to the findByDeptId object query which returns matching cached results and displays them in the Results screen.

Binding Transient Personalization Keys to Hybrid App Keys

Use transient personalization key values to determine the data to be cached.

Prerequisites

You must have transient personalization keys mapped to Mobile Business Object load arguments.

Task

1. Launch the Hybrid App Designer from SAP Mobile WorkSpace and create a new Hybrid App:
 - a) Select **File > New > Hybrid App Designer**.
 - b) Select the parent folder that contains the MBO with a load argument mapped to a transient personalization key. Name the file and click **Next**.
 - c) Select **Responds to server-driven email notifications** from the Starting Points screen and click **Next**.
 - d) Select the MBO that contains the load argument to transient key mapping in the Search for MBO screen and click **OK**, then click **Next**.
 - e) Specify sample e-mail contents and click **Next**.
 - f) Specify the matching rules used to trigger a screen flow by highlighting the text, right-clicking it, and selecting **Select as matching rule**.
 - g) Click **Finish**.
2. In the Hybrid App Designer, map the personalization keys to the Hybrid App keys for the menu item:
 - a) From the Flow Design screen select the operation for which you are defining a mapping.
 - b) Select the Screen Design tab, and highlight the menu item you want to map.
 - c) Select **Personalization Key Mappings**, click **Add**, and select a personalization key from the drop-down list and the key to which it maps.

You can also fill the personalization key values from values extracted from the e-mail, depending on from where you are invoking the object query.

When the application runs, the values are sent from the client which are used to fill the load argument values, and determine what data is cached in the SAP Mobile Server cache (CDB) and returned to the client.

Hybrid App Package Customization

The designer-based user interface is customizable using HTML, JavaScript and CSS Web technologies.

Customizing Generated Code

Modify generated JavaScript code to customize the Hybrid App.

1. Use the Hybrid App Package Generation wizard to generate the Hybrid App package and its files.

When the Hybrid App package is generated, the `Custom.js` file is generated if not already present in the project. The `Custom.js` file is located in `Generated Hybrid App\<hybridapp_project_name>\html\js`.

2. Right-click the `Custom.js` file and select the editor in which to open the file.
3. Modify the JavaScript code in the file or add your own code.
4. Save and close `Custom.js`.

Since `Custom.js` is generated only if it is not already present in the Hybrid App project, it is not created again if you subsequently generate the Hybrid App package. In this way, your customizations are preserved.

5. Deploy the Hybrid App package to SAP Mobile Server.

Any time you customize the code, you must redeploy the Hybrid App package to SAP Mobile Server.

You can also add your own separate JavaScript files to `Generated Hybrid Apps \hybridapp_project_name\html\js`, then add custom code to the `Custom.js` file that calls the functions in the JavaScript files you added. Modularizing your custom code can prevent the `Custom.js` file from becoming too long, and make it easier for multiple developers to collaborate on the same Hybrid App.

Adding Local Resources to a Hybrid App Project

When loading resources using custom JavaScript, be aware of the folder structure.

Depending on localization, the structure and path to the local resource may be different.

Possible folder paths include:

- `.../html/default/hybridapp.html`
- `.../html/{locale}/hybridapp.html`
- `.../html/hybridapp.html`

Referencing custom resources in HTML elements requires the use of relative URLs. The parent directory may be the HTML directory, the root, or something else. There is no guarantee that the URL structure is always `http://hostname/html/hybridapp.html`. It is possible to copy the resources into each localization directory or reference the resources from one directory (paying attention to localization paths).

An example of a useful helper function to get the relative path to the HTML directory is:

```
/**
 * Returns relative URL to the html directory
 */
```

Develop a Hybrid App Using the Hybrid App Designer

```
function getRelativeRoot()
{
    return ((resources != null) ? "../" : "")
}

// Helper function usage
var imageElement = document.getElementById("ImageElement");
imageElement.src = getRelativeRoot() + "images/myImage.gif";
```

Generated Hybrid App Files

When you use the Hybrid App Generation wizard to create a Hybrid App package, all the package files are generated the first time. Subsequent generations overwrite only a small subset of the files.

Generated package files are created in a top-level folder with the name of the Hybrid App. If you choose the option to generate into the current project, this file is visible in WorkSpace Navigator under the project `Generated Hybrid App` folder.

These files are always generated:

- *hybridapp-name.zip* – a single archive containing all of the Hybrid App files, including the Web application files, look and feel files, and JavaScript files.
- *manifest.xml* – describes the contents of the *hybridapp-name.zip* file.
- *datajs-version.js* – a JavaScript library of functions for ODATA and native device services that are not included in Hybrid Apps by default. By referencing these functions in your customization (in *Custom.js*, you can incorporate functionality from third-party JavaScript SDKs into your Hybrid Apps.

These files are regenerated only if you select the **Generate platform specific files** option in the Hybrid App Package Generation wizard:

- *hybridapp.html* – contains all the screens in the Hybrid App, each in its own div element. This is used with the **Optimize for performance** look and feel. On Windows Mobile, it is used for all looks-and-feels.
- *hybridapp_Custom.html* – contains all the screens in the Hybrid App.
- *hybridapp_jQM.html* – contains all the screens in the Hybrid App. This is used with the **Optimize for appearance** look and feel on iOS, BlackBerry, and Android.
- *WorkflowClient.xml* – contains metadata that specifies how to map the data in the Hybrid App message to and from calls to Mobile Business Object (MBO) operations and object queries.
- *hybridapp_name.xml* – look and feel file that uses the basic *hybridapp_name.html* file.
- *js* and *css* – subfolders containing the Javascript and CSS style sheet files for the application, including these files:
 - *Resources.js* – allows you to access localized string resources.

- `HybridApp.js` – contains functions for common menu, screen, and database operations.
- PhoneGap JavaScript file. Typically named `js\platform\cordova-x.x.x.javascript`, for any Hybrid App package that is built for an Android, iOS, or BlackBerry device using the PhoneGap library. The file is copied from `<SMP_HOME>\MobileSDK<version>\HybridApp\API\Container`.

These files are generated only if you select the **Generate** option and the files do not exist:

- `API.js` and `Utils.js` – provide Hybrid App functions used to communicate with the Hybrid Web Container.
- `Custom.js` – enables you to add JavaScript code to customize the Hybrid App. Your file is preserved each time you regenerate the package.

You can edit this file to customize your Hybrid App. It is generated the first time, but is not overwritten subsequently. In this way, your changes are preserved each time you regenerate the Hybrid App package. Examples of ways you can customize the Hybrid App include:

- Manipulating HTML elements.
- Writing code that is called before or after generated behavior is invoked for menu items.
- Implementing custom validation logic.
- `WorkflowMessage.js` – provides functions to access Hybrid App message resources.
- All `*.css` files – defines formatting rules to render the screens in HTML.

These files are overwritten when you regenerate a package:

- All the files in the top-level `Generated Hybrid App\hybridapp-name` folder, including the XML and ZIP files.
- The files in the `html` subfolder.

Generated HTML Files

The Hybrid App Designer generates these HTML files.

- `hybridapp.html` – contains all the screens in the Hybrid App, each in its own `div` element. This is used with the **Optimize for performance** look and feel. On Windows Mobile, it is used for all looks-and-feels.
- `hybridapp_Custom.html` – contains all the screens in the Hybrid App.
- `hybridapp_jQM.html` – contains all the screens in the Hybrid App. This is used with the **Optimize for appearance** look and feel on iOS, BlackBerry, and Android.

Note: In Preferences, **Optimize for appearance** is the default look and feel.

Look and Feel Files

By default, on BlackBerry 6.0, Android, and iOS platforms, the jQuery Mobile look and feel is used. On BlackBerry 5.0, a custom look and feel is used as the default.

Note: In Preferences, **Optimize for appearance** is the default look and feel.

CSS files include:

- `jquery.mobile-1.1.0.css` – located in `Generated Hybrid App\Hybrid App name\html\css\jquery` folder and used on BlackBerry 6.0, Android, and iOS platforms. By default, pages are generated using the B data theme. Modify the `ui-body-a` class selector in this file to modify the look and feel, for example, the background image or color.
- `master.css` – located in `Generated Hybrid App\Hybrid App name\html\css\bb` and used on the BlackBerry 5.0 platform. This is used on the BlackBerry 5.0 platform when the Optimize for appearance preference is selected. Modify the `body` selector to change the look and feel, for example, the background color.
- `stylesheet.css` – located in `Generated Hybrid App\Hybrid App name\html\css`. This look and feel is considerably simpler, using no JavaScript code to manipulate the controls, and only a single CSS file. This style sheet is used on all platforms for the Optimize for performance preference is selected. To modify the background color for this look and feel, modify the `body` selector.

Default Look and Feel

The default look and feel is provided by the jQuery Mobile framework.

In Preferences, **Optimize for appearance** is the default look and feel.

For the standard look and feel, the layout of the HTML at a high level is:

- Each screen has a block, contained in a div element, with attributes `data-role="page"` and `data-theme="a"`. Each div element has a div child element with a `data-role="header"` attribute and a child element for the menu. Use the contents of the header div to manipulate the menu.

```
<div data-role="page" data-theme='a'  
id="Department_createScreenDiv">  
  <div data-role="header" data-position="inline">  
    <a data-icon="arrow-l"  
id="Department_createScreenDivCancel" name="Cancel"  
onclick="menuItemCallbackDepartment_createCancel();" > Cancel</a>  
    <h1>Department_create</h1>  
    <a id="Department_createScreenDivCreate" name="Create"  
onclick="menuItemCallbackDepartment_createSubmit_Workflow();" >  
Create</a>  
  </div>  
</div>
```

- The menu has one anchor (a) element for each menu item:

```
<a id="Department_createScreenDivCreate" name="Create"
onclick="menuItemCallbackDepartment_createSubmit_Workflow();">
Create</a>
```

- In addition to a menu, each screen div has a child div element with a `data-role="content"` attribute, where the controls are hosted. The content div element has a child div with a `data-role="scroller"` attribute. This div in turn has a form with a number of div elements. The content div is where you can do customizations, such as branding.

```
<div data-role="content" class="wrapper" >
  <div data-role="scroller">
    <form name="Department_createForm"
id="Department_createForm">
      <div class="customTopOfFormStyle" ><span
id="Department_createForm_help" class="help"></span></div>
      <div class="customTopOfFormStyle"
id="topOfDepartment_createForm"></div>
      <div class="editbox">
        <label class="left"
for="Department_create_dept_name_paramKey">Dept name:</label>
        <input class="right" type="text"
id="Department_create_dept_name_paramKey"/><span
id="Department_create_Department_create_dept_name_paramKey_help"
class="help"></span>
      </div>
```

The first div element is a block used to display help in a span element.

The next div is a built-in element that can be used to find the top of the form. The last div is another built-in element that can be used to find the bottom of the form.

In the `Custom.js` file, it is recommended that you add customizations such as branding to the div element, "TopOf" ScreenKey "Form" and "bottomOf" screenKey "Form." For example:

```
/*
var screenKey = getCurrentScreen();
var form = document.forms[screenKey + "Form"];
if (form) {
var topOfFormElem = document.getElementById("topOf" screenKey +
"Form");
! topOfFormElem.innerHTML = "Use this screen to ...";
var bottomOfFormElem = document.getElementById("bottomOf"
screenKey + "Form");
bottomOfFormElem.innerHTML = "<a href=\"help.html\">Click here to
open help</a>";
}
*/
```

All the other divs in the form correspond to the controls put on that screen during design time in the Hybrid App Designer. You might see, for example, a div that holds a label and a textbox (input element). When the page is opened, the controls are enhanced by jQuery Mobile to supply additional functionality for controls like buttons, sliders, text inputs, and combo boxes.

Develop a Hybrid App Using the Hybrid App Designer

A typical Hybrid App with this look and feel, without extraneous attributes, might look like this:

```
<html>
  <body onload="hwc.onHybridAppLoad();">
    <div data-role="page" data-theme='a'
id="Department_createScreenDiv">
      <div data-role="header" data-position="inline">
        <a data-icon="arrow-l" id="Department_createScreenDivCancel"
name="Cancel" onclick="menuItemCallbackDepartment_createCancel();">
Cancel</a>
        <h1>Department_create</h1>
        <a id="Department_createScreenDivCreate" name="Create"
onclick="menuItemCallbackDepartment_createSubmit_Workflow();">
Create</a>
      </div>
      <div data-role="content" class="wrapper" >
        <div data-role="scroller">
          <form name="Department_createForm"
id="Department_createForm">
            <div class="customTopOfFormStyle" ><span
id="Department_createForm_help" class="help"></span></div>
            <div class="customTopOfFormStyle"
id="topOfDepartment_createForm"></div>
            <div class="editbox">
              <label class="left"
for="Department_create_dept_name_paramKey">Dept name:</label>
              <input class="right" type="text"
id="Department_create_dept_name_paramKey"/><span
id="Department_create_Department_create_dept_name_paramKey_help"
class="help"></span></div>
              <div class="customBottomOfFormStyle"
id="bottomOfDepartment_createForm"></div>
            </form>
          </div>
        </div>
      </div>
    </body>
  </html>
```

Default Look and Feel CSS Files

CSS look and feel files include:

- `jquery.mobile-1.1.0.css` – located in Generated Hybrid App `\hybridapp-name\html\css\jquery` folder. By default, pages are generated using the B data theme. Modify the `ui-body-a` class selector in this file to modify the look and feel, for example, the background image or color.
- `master.css` – located in Generated Hybrid App `\hybridapp-name\html\css\bb`. Modify the `body` selector to change the look and feel, for example, the background color.
- `stylesheet.css` – located in Generated Hybrid App `\hybridapp-name\html\css`. This look and feel is simple: it uses no JavaScript code to manipulate the

controls, and only a single CSS file. This style sheet is used on all platforms for which the Optimize for performance preference is selected. To modify the background color for this look and feel, modify the `body` selector.

BlackBerry Custom Look and Feel File

`hybridapp_Custom.html` defines the HTML structure for the BlackBerry custom look and feel.

Each screen has a `div` element block with a form element, and each form has a number of `div` child elements. The first `div` in the form has a `span` used to display help. The next `div` is a built-in element that can be used to find the top of the form. The last `div` is another built-in element that can be used to find the bottom of the form. All the `div`s in the form correspond to the controls put on that screen in the Hybrid App Designer. You might get, for example, a `div` that holds a label and a textbox (input element).

This example shows a Hybrid App with this look and feel, without extraneous attributes:

```
<html>
  <body onload="hwc.onHybridAppLoad();" >
    <div id="Department_createScreenDiv">
      <form name="Department_createForm"
id="Department_createForm">
        <div class="customTopOfFormStyle" ><span
id="Department_createForm_help" class="help"></span></div>
        <div class="customTopOfFormStyle"
id="topOfDepartment_createForm"></div>
        <div class="editbox">
          <label class="left"
for="Department_create_dept_name_paramKey">Dept id:</label>
          <input class="right" type="text"
id="Department_create_dept_name_paramKey"/><span
id="Department_create_Department_create_dept_id_paramKey_help"
class="help"></span>
        </div>
      </form>
    </div>
  </body>
</html>
```

Optimize for Performance Look and Feel

This is a simple look and feel option that you can use on all platforms.

Note: Windows Mobile 6.x Professional platforms always use the Optimize for performance look and feel, as this platform is not supported by jQuery Mobile.

Choose the **Optimize for performance** option when you configure Hybrid App Designer preferences. For this look and feel, the layout of the HTML at a high level is:

- Each screen has a block, a `<div>` element. Each of those `<div>` elements has an unordered list element, ``, a child element for the menu. The menu has one list item, ``, for each menu item.

Develop a Hybrid App Using the Hybrid App Designer

- In addition to a menu, each <div> has a form element, <form>, where the controls are hosted.
- Each form has a single table, <table>, with a number of table rows, <tr>. The first table row has a block to display help, a element. The next table row is a built-in element, a table data or <td>, that can be used to find the top of the form.
- The last table row is another built-in element, a <td>, that can be used to find the bottom of the form.
- All the other rows in the form correspond to the controls put on that screen in the Hybrid App Designer. You might get, for example, a row with two table datas, the first holding a <label> and the second holding a textbox (<input>).
- A column can have only one width, so if you have more than one line, one column may contain different widths, which means the last width prevails. The contents of a field are wrapped only where there is a space. If there is no space, the contents are not wrapped. As a result, depending on the length of the data, Listviews may not respect the field widths specified in the Hybrid App Designer with this look-and-feel.

A typical Hybrid App with this look and feel, without extraneous attributes, looks similar to this:

```
<html>
  <body onload="onHybridAppLoad();" >
    <div id="Department_createScreenDiv">
      <ul id="Department_createScreenDivMenu" class="menu">
        <li><a class="nav" name="Create"
onclick="menuItemCallbackDepartment_createSubmit_Workflow();">Creat
e</a></li>
        <li><a class="nav" name="Cancel"
onclick="menuItemCallbackDepartment_createCancel();">Cancel</a></
li>
      </ul>
      <form name="Department_createForm"
id="Department_createForm">
        <table class="screen">
          <tr>
            <td colspan="2"><span id="Department_createForm_help"
class="help"></span></td>
          </tr>
          <tr>
            <td colspan="2" id="topOfDepartment_createForm"></td>
          </tr>
          <tr>
            <td class="left"><label
for="Department_create_dept_name_paramKey">Dept name:</label></td>
            <td class="right"><input class="right" type="text"
id="Department_create_dept_name_paramKey"/><span
id="Department_create_Department_create_dept_name_paramKey_help"
class="help"></span></td>
          </tr>
          <tr><td colspan="2" id="bottomOfDepartment_createForm"></
td></tr></table>
        </form>
      </div>
```

```
</body>
</html>
```

Reference

This section describes the generated files and the Hybrid App client API.

Hybrid App Client API

SAP Mobile Platform Hybrid Apps include a JavaScript API that open Hybrid Apps to customization, from including client-side business logic to changing the presentation layer.

Use the client API to build custom applications to support SAP Mobile Platform Hybrid App features and functionality.

Public JavaScript Functions

The JavaScript files contain the functions that you can access for use with Hybrid App package customization.

The files where the Hybrid Web Container JavaScript APIs are defined are located in `<SMP_HOME>\UnwiredPlatform\MobileSDK<version>\HybridApp\API\Container`.

Note: The detail of the individual APIs is not available if you are viewing this document from DocCommentXchange (<http://dcx.sybase.com>) or in PDF format. You can access this information by going to Product Documentation: access <http://sybooks.sybase.com/sybooks/sybooks.xhtml?id=1289&c=firsttab&a=0&p=categories>, then navigate to the current version of this topic.

These JavaScript files are also included:

- `Utils.js` – does not contain public functions to call
- `HybridApp.js` – does not contain public functions to call
- `json2.js` – third-party library. For information about the functions in this library, see the JSON documentation at <http://json.org>
- `cordova-2.0.0.javascript` – contains PhoneGap APIs. For information about PhoneGap APIs, see the documentation at www.phonegap.com.

API.js

The `API.js` file contains several different types of functions.

They include:

- General and Hybrid App utility functions
- Validation functions
- Credential functions

Hybrid App UI Functions

Functions that allow you to access the Hybrid App user interface (UI).

`updateUIFromMessageValueCollection`

To completely override the behavior provided by `updateUIFromMessageValueCollection` for a given screen, provide a `UIUpdateHandler` object for that screen. That `UIUpdateHandler` object has a `screenName` property, which indicates which screen's behavior it is overriding, and a callback function that indicates the function to call for that screen. That function is passed in the relevant `MessageValueCollection` object and it is its responsibility to update the controls' values based on its contents. An example of this is:

```
function MyListViewUpdateHandler() {
    this.screenName = "Prev_Expenses";
    this.values;
}

MyListViewUpdateHandler.prototype.callback = function(valuesIn)
{
    // Rows returned from RMI Call
    this.values = valuesIn;

    // construct our table
    try {
        var mvc =
this.values.getData("PurchaseTrackingJC_findOtherRequests_resultSet
Key");
        var txt = "";
        var htmlOut = "<p>";

        // Do we have any rows to display?
        if (mvc.value.length > 0) {
            // Start the table and header
            htmlOut += "<table id='MyPrevExpensesTable'
class='altrowstable'>";
            htmlOut += "<tr><th>Item Name</th><th>Cost</th></tr>";

            // Draw the rows+H15
            for (var rows = 0; rows < mvc.value.length; rows++) {
                var mvName =
mvc.value[rows].getData("PurchaseTrackingJC_itemName_attribKey");
                var mvCost =
mvc.value[rows].getData("PurchaseTrackingJC_itemCost_attribKey");

                if (mvName && mvCost) {
                    // Alternate the row colors
                    htmlOut += "<tr
onclick='navigateForward(\"Prev_Expenses_Detail\", \" +
mvc.value[rows].getKey() + \");'";
                    if (rows % 2 == 0) {
                        htmlOut += " class='evenrowcolor'>";
                    }
                }
            }
        }
    }
}
```



```

        else {
            htmlOut += " class='oddrowcolor'>";
        }

        htmlOut += "<td>" + mvName.getValue() + "</
td><td>" + mvCost.getValue(); +"</td></tr>";
    }
}

// Finish the table
htmlOut += "</table>";
}
else {
    htmlOut += "No rows returned.";
}
htmlOut += "</p>";

//Now add the table to the document
var form = document.forms[curScreenKey + "Form"];
if (form) {
    //var topOfFormElem = document.getElementById("topOf" +
curScreenKey + "Form");

    var topOfFormElem =
document.getElementById("PurchaseTrackingJC_findOtherRequests_resul
tSetKey");
    topOfFormElem.innerHTML = htmlOut;
}

}
catch (e) {
    alert(e.message);
}
} // function callback

function customAfterWorkflowLoad() {
    //Setup UIHandler to draw our Listview Screen
    UIUpdateHandlers[0] = new MyListViewUpdateHandler();
}

```

Hybrid App Native Device Functions

Access the native features of the device using the native device functions.

showUrlInBrowser(url)

To have a hyperlink in the default value for the HtmlView control, or for doing customization in Javascript, follow the **showUrlInBrowser** method without using standard HTML. To add HTML in the default value for the HtmlView control, you can use something similar to:

```

<html>
<body>
<b>Welcome</b><br>
<br>Your activation was successful, the newly created Hybrid App
requests will automatically be pushed to you.<br>
<br>For more information contact your administrator or visit us

```

Develop a Hybrid App Using the Hybrid App Designer

```
at:<br>
<br>
<a href="javascript:showUrlInBrowser('http://www.sap.com/
unwiredenterprise')">SAP Mobile Platform</a>
</body>
</html>
```

View an attachment such as an image, a Word document, a PDF file, and so on as part of the Hybrid App package. This example uses an image file.

1. Generate the Hybrid App package and its files.
2. In WorkSpace Navigator, go to the location where the generated Hybrid App files are located and add an images folder under the html folder, for example, Generated Hybrid App\`<hybridapp_name>`\html\images.
3. Copy an image to the images folder.
4. In the Hybrid App Designer, add a menu item to the Hybrid App.
5. Open the Custom.js file with a text editor and edit the method `customBeforeMenuItemClick`:

```
if (screen === "ScreenKeyName" && menuItem === "ShowAttachment") {
    showLocalAttachment("html/images/ipod.jpg");
    return false;
}
```

6. Save and close the Custom.js file.
7. Deploy the Hybrid App package to SAP Mobile Server.

Hybrid App Message Data Functions

Access the Hybrid App message data functions.

A Hybrid App has an in-memory data structure where it stores data. This data is used to update the controls on the screen through `updateUIFromMessageValueCollection()`. Values are extracted from those controls and used to update the data through `updateMessageValueCollectionFromUI()`.

You can program the data content and use it to make decisions on the client. To get the active instance of this data structure, you start by calling `getDataMessage()`. This returns a `WorkflowMessage` object. This object has a function, `getValues()`, that is used to return the top-level `MessageValueCollection` object. This object has a list of key-value pairs, represented by `MessageValue` objects and is retrieved by calling `getData(key)`. `getData()` returns either a single `MessageValue` object, or an array of `MessageValueCollection` objects.

A typical Hybrid App message might look similar to this.

```
WorkflowMessage
.getHeader()           <undefined>
.getWorkflowScreen()  "salesorderList_newSOCreate"
.getRequestAction()   "Submit_Workflow"
.getValues()           MessageValueCollection
.getData("salesorderList_newSOCreate_WITHOUT_COMMIT_paramKey")
```

```

        .getKey()
"salesorderList_newSOCreate_WITHOUT_COMMIT_paramKey"
        .getType()      "TEXT"
        .getValue()     "1"
        .getData("BAPI_SALESORDER_CREATEFROMDAT1_ORDER_HEADER_IN
_DOC_TYPE_attribKey")
        .getKey()
"BAPI_SALESORDER_CREATEFROMDAT1_ORDER_HEADER_IN_DOC_TYPE_attribKey"
        .getType()      "TEXT"
        .getValue()     "1"
        .getData("BAPI_SALESORDER_CREATEFROMDAT1_ORDER_HEADER_IN
_SALES_ORG_attribKey")
        .getKey()
"BAPI_SALESORDER_CREATEFROMDAT1_ORDER_HEADER_IN_SALES_ORG_attribKey
"
        .getType()      "TEXT"
        .getValue()     "1"
        .getData("BAPI_SALESORDER_CREATEFROMDAT1_ORDER_HEADER_IN
_DISTR_CHAN_attribKey")
        .getKey()
"BAPI_SALESORDER_CREATEFROMDAT1_ORDER_HEADER_IN_DISTR_CHAN_attribKe
y"
        .getType()      "TEXT"
        .getValue()     "1"
        .getData("BAPI_SALESORDER_CREATEFROMDAT1_ORDER_HEADER_IN
_DIVISION_attribKey")
        .getKey()
"BAPI_SALESORDER_CREATEFROMDAT1_ORDER_HEADER_IN_DIVISION_attribKey"
        .getType()      "TEXT"
        .getValue()     "1"
        .getData("salesorderList_newSOCreate_ORDER_PARTNERS_para
mKey")
        MessageValue
        .getKey()
"salesorderList_newSOCreate_ORDER_PARTNERS_paramKey"
        .getType()      "LIST"
        .getValue()     MessageValueCollection[]
        [0].getKey()    "6476c1a4-94e9-e5a4-b903-
caf2ca613c4a"
        [0].getState()  "add"
        [0].getData("PARTN_ROLE")
        MessageValue
        .getKey()      "PARTN_ROLE"
        .getType()     "TEXT"
        .getValue()    "1"
        [0].getData("PARTN_NUMB")
        MessageValue
        .getKey()      "PARTN_NUMB"
        .getType()     "TEXT"
        .getValue()    "1"

```

getCurrentMessageValueCollection

Handling individual items

Develop a Hybrid App Using the Hybrid App Designer

```
var message = getCurrentMessageValueCollection();

var cityObj = message.getData("Customer_city_attribKey");
var city = cityObj.getValue();

var stateObj = message.getData("Customer_state_attribKey");
var state = stateObj.getValue();

var zipObj = message.getData("Customer_zip_attribKey");
var zip = zipObj.getValue();
```

List

```
var message = getCurrentMessageValueCollection();
var itemList = message.getData("CustDocs");

var items = itemList.getValue();
var noOfItems = items.length;
var i = 0;

while (i < noOfItems) {
    var theItems = items[i];
    var
fileNameObj=theItems.getData("CustDocs_fileName_attribKey");
    var fileName = fileNameObj.getValue();
    i = i + 1;
}
```

Callbacks.js File

This file contains callback functions.

Callback functions are typically used for event handlers that are asynchronous.

Camera.js

These functions allow you to take a picture from the camera, or pick one from the photo library and use the picture in the Hybrid App.

getPicture Function

The `getPicture` function provides access to the device's default camera application or device's photo library for retrieving a picture asynchronously.

If the `SourceType` is `CAMERA` or `BOTH`, the `getPicture` function opens the device's default camera application (if the device has a camera) so the user can take a picture. Once the picture is taken, the device's camera application closes and the Hybrid App is restored. If the device does not have a camera application, the function reports that it is not supported.

Using the getPicture Function for Larger Image Sizes

For larger images, use the `IMAGE_URI` destination type.

For larger images, use the `IMAGE_URI` destination type. The `MIME` type for the image URI is determined using the extension of the file name parameter in the `onGetPictureSuccess`

callback. You must add this extension information to the Hybrid App message as a separate MessageValue to use it on the server. For the HTML image tags, the browser should be able to determine the type through the HTTP connection opened on the URI.

You must create a new option object similar to this:

```
var options = { destinationType:
PictureOption.DestinationType.IMAGE_URI,
                sourceType: PictureOption.SourceType.CAMERA
            };

getPicture(onPictureError, onPictureSuccess, options);
```

The destinationType can be PictureOption.DestinationType.IMAGE_DATA (Base64 string behavior), or the new PictureOption.DestinationType.IMAGE_URI type. Depending on the destination type specified, the picture success callback's second parameter may be a Base64 string or a URI. The source type can be PictureOption.SourceType.CAMERA, PictureOption.SourceType.PHOTOLIBRARY., or PictureOption.SourceType.BOTH.

The image URI passed back is expected to be valid and resolvable to the image by the browser. You can create an HTML image tag with a URI to display the image, for example, . This can also be used to create thumbnails.

Uploading the Image to the Server for a URI

To upload the image to the server for a URI, you must create a MessageValue in the JavaScript with a "FILE" type. When the JavaScript Hybrid App message is serialized it will identify if the message contains files. During a submit or online request, the query sent to the container will contain a new query parameter that identifies that this message must be parsed again. The query looks similar to: ?querytype=submit&parse=true.

Note: When you upload a large image to the server using an online request, rather than a submit Hybrid App, the image contents come back from the online request, which can result in too large of a Hybrid App message for the container to handle. It is recommended that you use the submit action instead of online request action when it is likely that the message size will be very large, such as when it includes large images.

The custom code must call the function

```
getDataMessage().setHasFileMessageValue(true);
```

for the parse query to be sent to the container.

When uploading the image to the server for a URI, the JavaScript looks similar to the following example. Note that this example is specific to top level screens. See Example 3 for a more general code example:

```
var options = { destinationType:
PictureOption.DestinationType.IMAGE_URI, sourceType:
PictureOption.SourceType.PHOTOLIBRARY };

getPicture( onGetPictureError, onGetPictureSuccess, options );
```

Develop a Hybrid App Using the Hybrid App Designer

```
function onGetPictureSuccess(fileName, imageUri){
    // Set file for upload
    var fileDataKey = "Picture_create_fileData_paramKey";

    //Code for calling from top level screen
    var messageValue =
    getDataMessage().getValues().getData(fileDataKey);

    if (messageValue)
    {
        // Update file for upload
        messageValue.setValue(imageUri);
    }
    else
    {
        // Add file for upload
        messageValue = new MessageValue();
        messageValue.setKey(fileDataKey);
        messageValue.setValue(imageUri);
        messageValue.setType(MessageValueType.FILE);
        //Code for calling from top level screen
        getDataMessage().getValues().add(fileDataKey, messageValue);
    }

    getDataMessage().setHasFileMessageValue(true);
}
```

Handling a larger image size example:

```
function reportError(errCode)
{
    if (errCode != PictureError.USER_REJECT) {
        // error occurred
    }
}

function reportImage(fileName, imageUri)
{
    // Image captured
    alert("Photo taken");

    // Optional - Display preview in image tag
    var imageTagId = "Thumbnail"; // The id of your image tag
    var imageElement = document.getElementById(imageTagId);
    imageElement.src = imageUri;

    // Optional - Create message value to upload image
    var fileKey = "Picture_create_fileData_paramKey"; // Key that
maps to submit or online request parameter
    var messageValue = new MessageValue();
    messageValue.setKey(fileKey);
    messageValue.setValue(imageUri);
    messageValue.setType(MessageValueType.FILE);

    // Add message value to Workflow message - NOTE: Code may differ
```

```
dependent on the context for adding image (Eg. ListView).
    getDataMessage().getValues().add(fileKey, messageValue);

    getDataMessage().setHasFileMessageValue(true); // Explicitly
tell Workflow about image
}
var options = { destinationType:
PictureOption.DestinationType.IMAGE_URI, sourceType:
PictureOption.SourceType.CAMERA};
getPicture( onGetPictureError, onGetPictureSuccess, options );
```

When uploading the image to the server for a URI, the JavaScript looks similar to the following example. This example is more general compared to Example 1, since it is invoked from a lower level screen:

```
// invoke from a lower level screen
var messageValue =
getCurrentMessageValueCollection().getData(contentDataKey);

if (messageValue)
{
// Update file for upload
messageValue.setValue(base64String);
}
else
{
// Add file for upload
messageValue = new MessageValue();
messageValue.setKey(contentDataKey);
messageValue.setValue(base64String);
messageValue.setType(MessageValueType.TEXT);
//invoke from a lower level screen
getCurrentMessageValueCollection().add(contentDataKey,
messageValue);
}
```

Limitations

The server has a limit of 75MB per parameter, which is what the Hybrid Web Container uses as the `XmlWorkflowMessage`. Therefore, the server imposes a maximum size limit of 50 MB (assuming one picture per `XmlWorkflowMessage`, and no other keys are present). Keep in mind that clients may impose a lower limit than 50MB.

Note: When accessing very large binary (image) data in the mobile business object associated with the Hybrid App, ensure that the attribute set in the mobile business object is a **BigBinary** datatype, rather than `Binary`.

Certificate.js

Provides functions for X.509 credential handling.

Use these functions to create a user interface in HTML and JavaScript, that uses X.509 certificates as the Hybrid App credentials.

Develop a Hybrid App Using the Hybrid App Designer

This file contains the functions that allow parsing a certificate date, creating a certificate from a JSON string value, retrieving a certificate from a file (Android), retrieving a certificate from the server (iOS), and so on.

You can choose to set the results of a `getSignedCertificate` function as the password.

```
certificateLabels(filterSubject, filterIssuer)

// The following script gets all the labels for certificates
// with the provided subject and issuer
var certStore = CertificateStore.getDefault();
var labels = certStore.certificateLabels("MyUser", "mydomain.com");

- getPublicCertificate(label)

// The following script gets the certificate data for the first
// certificate to match the provided subject and issuer
var certStore = CertificateStore.getDefault();
var labels = certStore.certificateLabels("MyUser", "mydomain.com");
var cert = certStore.getPublicCertificate(labels[0]);

- getSignedCertificate(label)

// The following script gets the signed certificate data for the
// first
// certificate to match the provided subject and issuer
var certStore = CertificateStore.getDefault();
var labels = certStore.certificateLabels("MyUser", "mydomain.com");
var cert = certStore.getSignedCertificate(labels[0]);

var username = cert.subjectCN;
var password = cert.signedCertificate;

- listAvailableCertificatesFromFileSystem(sFolder, sFileExtension)

// The following script gets an array of file paths for files on
// the sdcard with the extension p12
var certStore = CertificateStore.getDefault();
var certPaths = certStore.listAvailableCertificatesFromFileSystem("/sdcard/", "p12");

- getSignedCertificateFromFile(filePath, password)

// The following script gets the signed certificate data for the
// first
// p12 file found on the sdcard
var certStore = CertificateStore.getDefault();
var certPaths = certStore.listAvailableCertificatesFromFileSystem("/sdcard/", "p12");
var cert = certStore.getSignedCertificateFromFile(certPaths[0], "password");

- getSignedCertificateFromServer(username, serverPassword, certPassword)
```



```
// The following script gets the signed certificate data for the
// user MYDOMAIN\MYUSERNAME from the server
var certStore = CertificateStore.getDefault();
cert = certStore.getSignedCertificateFromServer("MYDOMAIN\
\MYUSERNAME", "myserverpassword", "mycertpassword");
```

Custom.js File

The first time you generate the Hybrid App package files, the `Custom.js` file is generated.

In subsequent file generations for the same Hybrid App package, this file will not be overwritten, so any customizations you make are preserved.

These touch points are available for customization: `WorkflowLoad`, `Submit`, `NavigateForward`, `NavigateBackward`, `ShowScreen`, `MenuItemClick`, and `Save`. At each touch point, a **customBefore** method is invoked and a **customAfter** method is invoked. The `customBefore` method returns a boolean. If it returns true, it continues to execute the default behavior, for example, navigating to a new screen or performing an online request. If it returns false, it does not execute the default behavior, so you can override the default behavior by customizing these methods.

The `Custom.js` file contains these methods:

Note: You can delegate the implementation of these functions to different functions supplied in other custom JavaScript files. It is not necessary to include all of your customization logic in the single `Custom.js` file.

```
//Use this method to add custom html to the top or bottom of a form
function customBeforeWorkflowLoad() {

    var form = document.forms[curScreenKey + "Form"];
    if (form) {
        // header
        var topOfFormElem = document.getElementById("topOf" +
curScreenKey + "Form");

        if (topOfFormElem) {
            topOfFormElem.innerHTML = "<img id='ImgSylogo' src='./
images/syLogo.gif' /><br/>";

            // footer
            var bottomOfFormElem = document.getElementById("bottomOf"
+ curScreenKey + "Form");
            bottomOfFormElem.innerHTML = "<p>Copyright 2010, Sybase
Inc.</p>";
        }
    }
    return true;
}
```

When using the `customBeforeNavigateForward(screenKey, destScreenKey) {}` function, if you want to create your own JQuery Mobile style listview, remember that JQueryMobile does

Develop a Hybrid App Using the Hybrid App Designer

not allow duplicate ID attributes. So if there is an existing listview with the same ID attribute, you must:

1. Delete the existing listview with the same ID attribute.
2. Re-create the listview.
3. Call **refresh** for your listview.

For example:

```
//Use this method to add custom code to a forward screen transition.
If you return false, the screen
//transition will not occur.
function customBeforeNavigateForward(screenKey, destScreenKey) {

..
try {
    if (destScreenKey == 'Personal_Work_Queue') {

        //grab the results from our object query
        var message = getCurrentMessageValueCollection();
        var itemList = message.getData("PersonalWorkQueue");
        var items = itemList.getValue();
        var numOfItems = items.length;
        var i = 0;

        //iterate through the results and build our list
        var htmlOutput = '<div id="CAMSCustomViewList"><ul data-
role="listview" data-filter="true">';
        var firstOrder = '';

        while ( i < numOfItems ){
            var currItem= items[i];
            var opFlags =
currItem.getData("PersonalWorkQueue_operationFlags_attribKey").getV
alue();
            var orderId =
currItem.getData("PersonalWorkQueue_orderId_attribKey").getValue();
            var operationNumber =
currItem.getData("PersonalWorkQueue_operationNumber_attribKey").get
Value();
            var description =
currItem.getData("PersonalWorkQueue_description_attribKey").getValu
e();
            try {
                var promDate =
currItem.getData("PersonalWorkQueue_datePromised_attribKey").getVal
ue();
            } catch (err) {
                var promDate = "";
            }

            try {
                var planDate =
currItem.getData("PersonalWorkQueue_dateStartPlan_attribKey").getVa
lue();
```

```

        } catch (err) {
            var planDate = "";
        }

        var onHold =
currItem.getData("PersonalWorkQueue_onHold_attribKey").getValue();

        htmlOutput += '<li><a id="' + currItem.getFullKey() +
'" class="listClick">';
        htmlOutput += '<p><b>Flags: </b>' + opFlags + '</p>';
        htmlOutput += '<p><b>Order Id: </b>' + orderId + '</
p>';
        htmlOutput += '<p><b>Operation No: </b>' +
operationNumber + '</p>';
        htmlOutput += '<p><b>Title: </b>' + description + '</
p>';
        htmlOutput += '</a></li>';

        i++;
    }

    htmlOutput += '</ul></div>';

//append the html to the appropriate form depending on the
key
    if (destScreenKey == 'Personal_Work_Queue') {

        var listview = $('div[id="CAMSCustomViewList"]');
        //Try to remove it first if already added
        if (listview.length > 0) {
            var ul = $(listview[0]).find('ul[data-
role="listview"]');
            if (ul.length > 0) {
                htmlOutput = htmlOutput.replace('<div
id="CAMSCustomViewList"><ul data-role="listview" data-
filter="true">', '');
                ul.html(htmlOutput);
                ul.listview('refresh');
            }
            } else {
                $
                $('#Personal_Work_QueueForm').children().eq(2).hide();
                $
                $('#Personal_Work_QueueForm').children().eq(1).after(htmlOutput);
            }
        }
//add the listener based on the class added in the code
above
        $(".listClick").click(function() {
            currListDivID = $(this).parent().parent();
            $(this).parent().parent().addClass("ui-btn-active");

            //special case for bb
            navigateForward("Shop_Display", this.id );

```

```
        if (isBlackBerry()) {
            return;
        }
    });
}
```

Overriding the showErrorFromNative Function

The generated JavaScript allows you to override the behavior of the `showErrorFromNative` function using the `customBeforeReportErrorFromNative(errorString)` and `customAfterReportErrorFromNative(errorString)` methods.

This shows an example of how to override or customize the error message based on the returned numeric error codes through `customBeforeReportErrorFromNative`.

```
function customBeforeReportErrorFromNative(errorString) {
    var errorCode = getURLParamFromNativeError("errCode",
errorString);
    // 500 and above are network errors
    if ( errorCode >= 500 )
    {
        // Could check lang global variable if so desired
        //if ( lang == ... )
        {
            // Show your own custom error message based on errorCode
            showAlertDialog("Do you have a network connection?", "My
custom error");
            // return false to by pass default behavior
            return false;
        }
    }
    return true;
}
```

Identified error scenarios include:

- Any network related errors during an online (synchronous) request contain an error code of 500 or greater (check for `>= 500`)
- `public static final int UNKNOWN_ERROR = 1; // "unknown error"`
- `public static final int ATTACHMENT_NOT_DOWNLOADED = 100; // "Attachment has not been downloaded"`
- `public static final int UNKNOWN_MIME_TYPE = 101; // "Unknown MIME type"`
- `public static final int FILENAME_NO_EXTENSION = 102; // "File name without extension"`
- `public static final int REQUIRED_PARAMETER_NOT_AVAILABLE = 103; // "Required parameter is not available"`
- `public static final int UNSUPPORTED_ATTACHMENT_TYPE = 105; // "attachment type is not supported"`

- `public static final int SSOCERT_EXCEPTION = 106; //SSO Certificate manager exception`
- `public static final int FAIL_TO_SAVE_CREDENTIAL = 107; // Fail to save credential`
- `public static final int FAIL_TO_SAVE_CERTIFICATE = 108; // Fail to save certificate`
- `public static final int DEVICE_NOT_CONNECTED = 109; // Device is not connected`

Resources.js

The resource functions allow you to access localized string resources.

ExternalResource.js

These functions allow you to access resources on external HTTP servers.

This shows an example of the UPDATE function:

```
function update() {
    // Using json to update a value
    var url = // URL of your external resource;
    var webResponse;
    var options = {
        method: "PUT",
        data: "{\"Value\":\"Value A Updated\"}",
        headers: {
            "Content-type": "application/json"
        },
        async: false,
        complete: function(response) { webResponse = response; }
    };

    getExternalResource(url, options);

    if (webResponse.status === 200)
        alert("Update successful");
    else
        alert("Update Failed");
}
```

This shows an example of the DELETE function:

```
function delete() {
    // Delete a value
    var url = // URL of your external resource;
    var webResponse;
    var options = {
        method: "DELETE",
        async: false,
        complete: function(response) { webResponse = response; }
    };

    getExternalResource(url, options);
}
```

```
    if (webResponse.status === 200)
        alert("Delete successful");
    else
        alert("Delete Failed");
}
```

SUPStorage.js

Functions to store results from online requests in a specified cache.

Storage functions enable you to:

- Name the cached result sets
- Enumerate the cached result sets
- Read, delete, and modify cached contents individually for each cached result set

Usage Notes: PhoneGap must be initialized before a storage function is called. The initialization happens automatically when you generate code using the Hybrid App Designer; if you do not use Designer, you must detect PhoneGap initialization in your own code. See *Implementing PhoneGap*.

Cached result sets must be stored as strings (before deserialization to an `xmlWorkflowMessage` structure).

Calls to these methods do not trigger events.

Example: Constructors

```
// These constructors create two local storage instances with their
// own domain
var store1 = new hwc.SUPStorage("mydomain");
var store2 = new hwc.SUPStorage("myotherdomain");

// This constructor creates a shared storage instance whose key is
// the one set in the
// packaging tool for generated JavaScript API, or in the Hybrid App
// Designer.
var storeS = new hwc.SharedStorage();
```

Example: length

```
// Displays the current number of elements in the storage
var store = new hwc.SUPStorage();
alert(store.length());
```

Example: key(index)

```
// Displays the value at the provided index in the storage
var store = new hwc.SUPStorage();
alert(store.key(2));
```

Example: getItem(key)

```
// Displays the value for the provided key
```

```
var store = new hwc.SUPStorage();
alert(store.getItem("mykey"));
```

Example: setItem(key, value)

```
// Sets a key/value pair
var store = new hwc.SUPStorage();
store.setItem("mykey", "myvalue");
```

Example: emoveItem(key)

```
// Removes a key/value pair
var store = new hwc.SUPStorage();
store.removeItem("mykey");
```

Example: clear

```
// Clears the storage
var store = new hwc.SUPStorage();
store.clear();
```

SAP Mobile PlatformStorage

The SAP Mobile Platform Storage API allows you to store structured data on the client side.

You can also use these functions as an arbitrary key or value storage mechanism. Keys are strings, and any string (including the empty string) is a valid key. Keys cannot be duplicated in the same Hybrid App package. Values are also strings and values can be duplicated in the same Hybrid App package. Keys and values can contain multi-byte characters.

SUPStorage can span multiple screens in the Hybrid App, and lasts beyond the current session. This allows the storage of user data on the client, such as entire user-authored documents.

Using platform-specific mechanisms, the items stored using the SUPStorage API are encrypted according to the particular platform policies:

Platform	Encryption policy
BlackBerry	PersistentStore, which adheres to the Content Protection BES IT policy
Android	Encrypted before storing into the SQLite database
iOS	Stored in SQLite Encryption Extensions database
Windows Mobile	Unencrypted SQLite—security is deferred to Afaria Security Manager

The amount of data that can be stored on the client is limited only to the available storage space on the particular platform:

Platform	Data storage
BlackBerry	Amount of free <code>PersistentStore</code> .
iOS and Android	Amount of free file system for the SQLite database, and/or the SQLite database size limit
Windows Mobile	Amount of free file system, and the SQLite database size limit.

Limitations

- The amount of data that you can retrieve and return to the JavaScript space when using the `SUPStorage` API is limited to the JavaScript size limitation as established for each platform. See the topic *Attachment Viewer and Image Limitations* in *SAP Mobile WorkSpace - Hybrid App Package Development*.
- On Windows Mobile devices, there is a 500K limitation for the length of the shared storage item. If the length of a shared item is more than 500K, the JavaScript does not accept anything.
- Physical SAP Mobile Platform storage is tied to a Hybrid App package. When the Hybrid App package is uninstalled, the corresponding SAP Mobile Platform storage for the Hybrid App package is removed immediately.
- Items stored using the `SUPStorage` API are persisted, and therefore, survive soft device resets.
- `SUPStorage` persists through invocations of the Hybrid App.
- The `SUPStorage` API does not restrict reading or writing of the storage data from different domains. For example, if a Hybrid App loads some code from an external HTTP server that attempts to access the `SUPStorage` API, it is allowed.
- The `SUPStorage` API does not take into account the current locale or language of the device. You can, however, access the global JavaScript variable called *lang* and implement this in your custom code.

Shared Storage

All Hybrid Apps with a shared storage key assigned share the storage with other Hybrid Apps that have the same storage key assigned.

- When the last Hybrid App with the shared storage key is removed from the device, the storage data is also removed.
- Since shared storage data is loaded into JavaScript, the same limitations apply to it as that which applies to the JavaScript size limitation as established for each platform. See the topic *Attachment Viewer and Image Limitations*. If a large amount of data is involved in the operation, the shared storage should be used only to store the reference or location of the data, not the data itself. This helps to ensure you stay within the JavaScript size limitations.

For example, if data for an image needs to be saved in shared storage for later use, the image data should be stored in the device file system or the persistent store, and then store only the file path to the shared storage.

- Shared storage items are removed when the last Hybrid App using the same shared storage key is removed from the device (it happens on unassignment)
- On Windows Mobile devices, there is a 500K limitation for the length of the shared storage item. If the length of a shared item is more than 500K, the JavaScript does not accept anything.

Timezone.js

The date/time functions allow you to extract and format the date and time for the Hybrid App.

WorkflowMessage.js

Use these functions to access message resources.

Using Third-Party JavaScript Files

To include your own files in Hybrid Web Container, copy them into the appropriate place in the Generated Hybrid Apps folder.

To load external JavaScript and CSS files dynamically, copy the relevant third-party JavaScript and CSS files to the Generated Hybrid Apps\`<package_name>`\html and js or css folders. If the files are JavaScript files, and are in the html\js folder, they are automatically included in the HTML as script.

Note: On Android, individual HTML, JavaScript, and CSS files cannot exceed 1MB.

These files will be included in the Hybrid App `manifest.xml` and ZIP files automatically when the Hybrid App package is regenerated.

Repackaging Hybrid App Package Files

After modifying the `Custom.js` file, you must redeploy the Hybrid App package to SAP Mobile Server.

1. Save and close the modified files after adding your custom code.
2. In WorkSpace Navigator, right-click the `<hybrid_app_name>.xbw` file and select **Generate Hybrid App**.
3. In the Hybrid App generation wizard, select the connection profile.
4. In Generation Options, choose:
 - Generate Package
 - Update generated code
 - Deploy to an SAP Mobile Server as a replacement

5. Click **Finish**.

Common Customizations

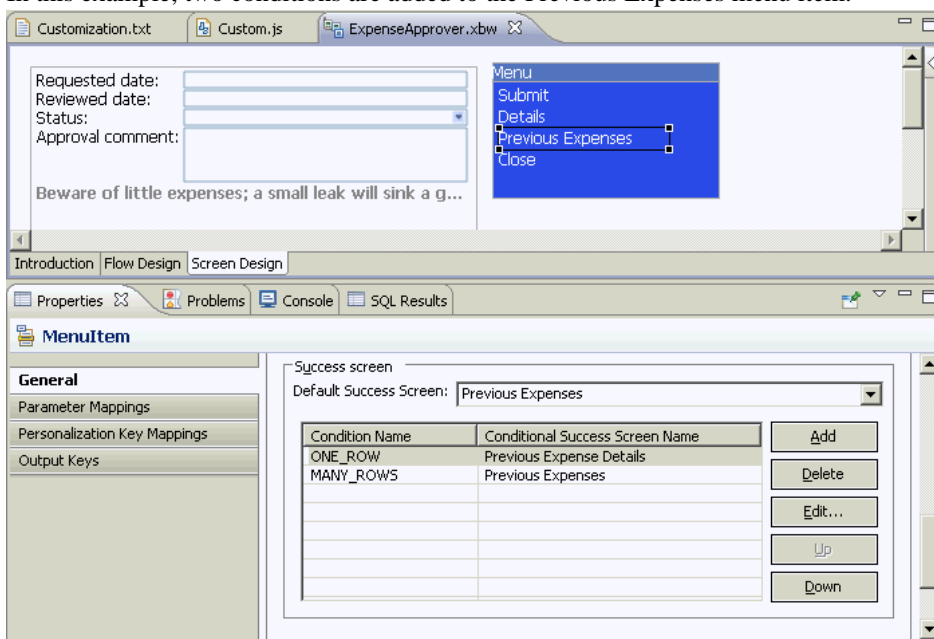
Implementing Conditional Navigation

Conditional navigation allows you to implement a custom function that allows you to override navigation behavior between screens.

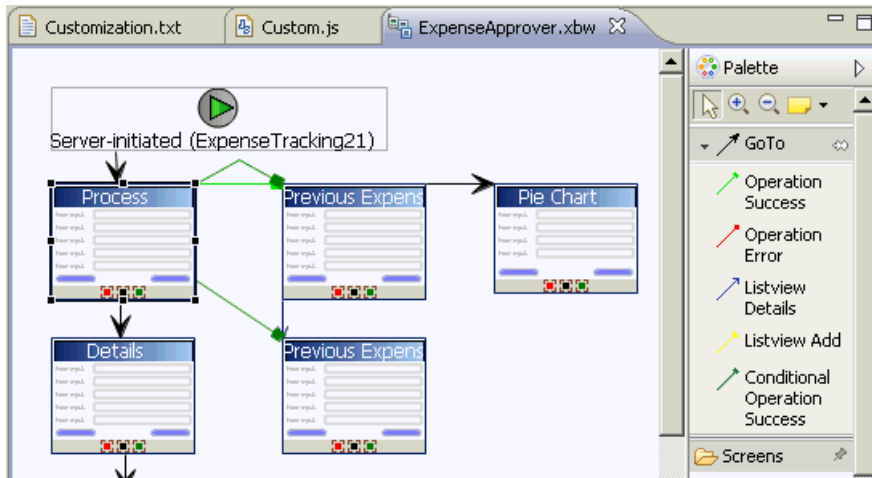
This procedure gives an example of how you can use conditional navigation to skip a screen.

1. In the Screen Design page, modify the menu item by adding conditions.

In this example, two conditions are added to the Previous Expenses menu item.



2. Go to the Flow Design page to see the conditional navigation paths in the flow.



3. In the Custom.js file, add the custom code for conditional navigation.

```
//This example demonstrates the conditional navigation
functionality for an online request.
//In this example we skip the list view screen and go directly to
the details screen if there is only one item in the list
function customConditionalNavigation(currentScreenKey,
actionName, defaultNextScreen, conditionName, workflowMessage) {
    if ((currentScreenKey === 'Process') && (actionName ===
'Previous Expenses')) {
        if (conditionName === 'ONE_ROW') {
            var values = workflowMessage.getValues();
            var m = workflowMessage.serializeToString();
            var expenseTracking =
values.getData("ExpenseTracking21View");
            var etList = expenseTracking.getValue();
            var count = etList.length;
            if (count == 1) {
                var etRow1 = etList[0];
                workflowMessage.updateValues(etRow1);
                return true;
            }
        }
        else if (conditionName === 'MANY_ROWS') {
            return false; //ie do the normal navigation which is
to go to the listview screen
        }
    }
    // default case is to NOT change the flow
    return false;
}
```

4. Use the Hybrid App Generation wizard to re-generate the Hybrid App package with a new hybridapp_jQueryMobileLookAndFeel.html file that contains the newly added conditional navigations.
5. Use a browser to debug the code.

Implementing a Conditional Start Screen

Add conditions that determine which start screen the user sees based on the conditions.

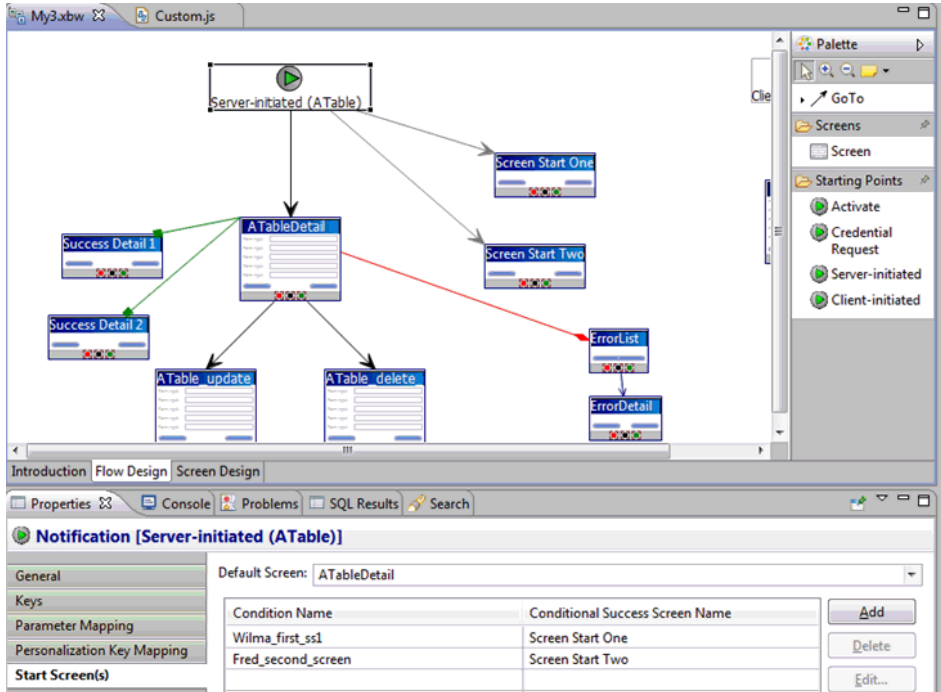
Like the conditional success navigation feature, there is a table of condition names with the matching Start screen. If all of the conditions are evaluated as false (or if they are absent), the default navigation is executed.

1. In the Flow Design page, select the server-initiated starting point to see the Properties.
2. In the Properties view, click **Start Screen(s)**.
3. Click **Add** to add a condition.
4. In the dialog, enter the condition name, select the target screen with which to associate the condition, and click **OK**.

This means that if the defined condition is found to be true, the screen you choose here will be the start screen. Condition names can include:

- Letters A-Z and a-z
- Numbers 0-9
- Embedded spaces (beginning and ending spaces are trimmed off)
- Special characters in the set \$._-+

In the Flow Design page, you can see the flow line for the conditional start is a shade of gray to differentiate it from the default GoTo line.



5. Add you custom code to the Custom.js file. For example:

```
function customConditionalNavigation( currentScreenKey,
actionName,
  defaultNextScreen,  conditionName,
  workflowMessage ) {
  if((currentScreenKey === SERVERINITIATEDFLAG) && (actionName
=== '')) {
    // conditional start screen uses this magic screen key and
the empty action name.
    if( conditionName === 'Wilma_first_ssl') {
      // custom logic
      return true;
    }
    else if(conditionName === 'Fred_second_screen'){
      // custom logic
      // return true or false
      return false;
    }
  }
  // default case is to NOT change the flow
  return false;
}
```

6. Regenerate the Hybrid App package.

When you regenerate the Hybrid App package, the `hybridapp.js` file is regenerated. The conditional start screen method is shown in the `hybridapp.js` file similar to this:

```
function customNavigationEntry() {
  this.condition;
  this.screen;
}
function customNavigationEntry( a_condition, a_screen ) {
  this.condition = a_condition;
  this.screen = a_screen;
}

/**
 * For the specific pair - screen named 'currentScreenKey' and the
action 'actionName', return
 * the list of custom navigation condition-names and their
destination screens.
 */
function getCustomNavigations( currentScreenKey, actionName ) {
  var customNavigations = new Array();
  if((currentScreenKey === SERVERINITIATEDFLAG) && (actionName
=== '')) {
    customNavigations[0] = new
customNavigationEntry( 'Wilma_first_ssl',
'Screen_Start_One' );
    customNavigations[1] = new
customNavigationEntry( 'Fred_second_screen',
'Screen_Start_Two' );
  }
  return customNavigations;
}
```

```
    return customNavigations;
}
```

Clearing the Contents of the Signature Control

Add JavaScript to clear the contents of a signature control.

1. Use the Hybrid App Generation wizard to generate the Hybrid App package and its files.

When the Hybrid App package is generated, the `Custom.js` file is generated if not already present in the project. The `Custom.js` file is located in `Generated Hybrid App\<project_name>\html\js`.

2. Open the `Custom.js` file and add your JavaScript code to the click event of a menu or button.

For example:

```
function customAfterMenuItemClick(screen, menuItem) {
    if (menuItem === "Clear Signature") {
        $.data(document.getElementById('sigKey'),
            'signature').clearSignature();
    }
}
```

3. Save and close the `Custom.js` file.
4. Re-generate the Hybrid App package and deploy it to SAP Mobile Server.

Security

Set up static or dynamic authentication, and configure the Hybrid App to use credentials.

Credentials

You can use either dynamic or static credentials in a Hybrid App screen flow.

See *Security* and *System Administration* for more detailed information about implementing security and certificates.

The user name and password values are required when the Hybrid App invokes a mobile business object operation. These authentication values can be provided statically (at design time), or dynamically (by the user at runtime). For requests sent by the client with a credential screen specified, requests are always invoked on the server using the credentials specified by the user, regardless of whether static or dynamic authentication is specified.

The choice of static versus dynamic authentication applies only to requests that must be executed on the server that do not have any credentials, or that do not have valid credentials. This happens when an object query needs to be run by a server-initiated notification, for example, or if the client provides incorrect credentials. In that scenario, the decision between static and dynamic becomes important. If static was chosen, it silently uses those hard-coded

credentials. If dynamic was chosen, it sends a notification to the client and asks the user to supply the credentials.

For example, you might define a server-initiated Hybrid App with a credential screen and static authentication. When the notification first comes in, it runs an object query using the hard-coded credentials. This is then sent to the user, who opens the notification and then makes an online request. This online request, be it an operation or an object query, will be made using the credentials supplied by the user.

Dynamic credentials require the user to enter the user name and password on a screen that the credential request starting point references. Select **Credential Cache User Name** and **Password** to indicate the user name and password to be required on the client. When the user logs in, the credentials are authenticated using the stored credentials.

Note: If an e-mail triggered Hybrid App has dynamic cached credentials, the cached credentials are not cached between invocations of the Hybrid App form through an e-mail trigger.

Static credentials mean that everyone who has access to the resource uses the same user name and password. By default, static credentials are used. The static credential user name and password for the Hybrid App can be extracted from the selected SAP Mobile Platform profile user name and password when the Hybrid App is generated, or they can be hard-coded using the Properties view. After deployment, you can change static credentials in SAP Control Center.

The application can also have a credential screen (Credential Request) that appears if the Hybrid App detects that the cached credentials are empty or incorrect.

Setting Up Static Authentication

With static authentication, everyone who has access to the resource uses the same user name and password.

Set up static credentials in the Authentication section of the Properties tab. To see the Properties page, verify there are no objects selected on the Flow Design page.

1. In the Properties view, click **Authentication**.
2. Select **Use static credentials**.
3. Select from these options:
 - **Use SAP Mobile Server connection profile authentication** – specifies that the user name and password associated with the connection profile are used when code is generated for the Hybrid App. Selected by default.
 - **Use hard-coded credentials** – sets the user name and password. When you select this option, the User name and Password fields are activated.
 - **Use certificate-based credentials** – enables you to use a certificate to generate authentication credentials.

4. (Optional) If you select **Use hard-coded credentials** in the previous step, enter the **User name** and **Password** that are to be used for authentication.
5. Select **File > Save**.

Setting Up Static Authentication Using a Certificate

Set up static authentication credentials generated from a certificate.

1. In the Properties view, click **Authentication**.
2. Select **Use static credentials** and Use certificate-based credentials.
3. Click **Generate from Certificate** to select a certificate file from which to generate authentication.
4. In the Certificate Picker, click **Browse** to locate the certificate to use.
5. Enter a password and select an alias, then click **OK**.

The information from the certificate is shown in the Properties view.

- Issuer – the issuer of the certificate
- Subject – the value of the subject field in the metadata of the certificate as defined in the X.509 standard
- Valid from – the date the certificate is valid from
- Valid until – the date after which the certificate expires

6. Select **File > Save**.

Setting Up Dynamic Authentication

Use dynamic authentication to enable the user to set the name and password on the client.

You can create the Credential Request starting point with a Credential screen automatically when you initially create a new Hybrid App, or you can create the Credential Request starting point and associated screen manually. This procedure shows how to create the Credential Request starting point automatically when you create a new Hybrid App.

1. In the Mobile Development perspective, select **File > New > Hybrid App Designer**.
2. Follow the instructions in the Hybrid App Designer wizard:
 - **Enter or select the parent folder** – select the Hybrid App project in which to create the Hybrid App screen flow.
 - **File name** – enter a name for the Hybrid App screen flow. The extension for Hybrid App screen flows is `.xbw`.
 - **Advanced** – link the Hybrid App screen flow to an existing file in the file system.
 - **Link to file in the file system** – click **Browse** to locate the file to which to link the Hybrid App screen flow. Linked resources are files or folders that are stored in the file system outside of the project's location. If you link a resource to an editor, when you select the editor, the resource is selected in the WorkSpace Navigator. Conversely, when you select the resource in the WorkSpace Navigator, the editor is selected.

Click **Variables** to define a new path variable. Path variables specify locations on the file system.

3. In the Starting Points page, select **Credentials (authentication) may be requested dynamically from the client application**.
4. Follow the steps to create the type of Hybrid App you want. Click **Finish**.
5. In the Hybrid App Designer, open the **Flow Design** to see the Credential Request starting point and its associated Credential Request screen.
To see the two pre-defined keys, `cc_username` and `cc_password` in the Properties view, click the Credential Request starting point.
6. Double-click the **Credential Request** screen to open the Screen Design page.
The two editbox controls on the screen are bound to the pre-defined keys, `cc_username` and `cc_password`.
7. Select **Username**. In the Properties view, open the **Advanced** page.
On the Username editbox, **Credential cache username** is selected by default. Click the **Password** editbox; the associated **Credential cache password** checkbox is selected.

Note: If you create a Credential Request starting point and screen manually, you must add the editbox controls, create the keys for the username and password, and check the corresponding Credential cache username or password box.

8. (Optional) To use certificate-based authentication instead of the user name and password:
 - a) Add a **MenuItem** to the Menu box.
 - b) Select the MenuItem to see its Properties.
 - c) In the Properties view, from Type, choose **Select Certificate**.
When the user selects the menu item on the device, a dialog opens to select a certificate for credentials.
9. Select **File > Save**.

The first time the Hybrid App is started following deployment, the credential screen opens. The username and password values are cached in the credential cache.

Note: If an e-mail-triggered screen flow has dynamic cached credentials, the cached credentials are not cached between invocations of the screen flow through an e-mail trigger.

Basic Authentication

On iOS, Android, and BlackBerry platforms, each Hybrid Web Container has a default basic authentication screen to enter credentials if challenged for basic authentication when Hybrid Web Container connects with the server.

The entered credentials are persisted, so any time the application restarts, the previously accepted credentials are used.

If the basic authentication screen is canceled, it is shown again only under these circumstances:

Develop a Hybrid App Using the Hybrid App Designer

- New connection information is entered and saved on the settings screen
- The restart engine menu item is pressed on the settings screen
- The application is restarted (device restart or force stop)

See *HTTP Authentication Security Provider* in *Security* for more information.

Single Sign-on

Android, BlackBerry, and iOS Hybrid Apps can provide a single sign-on (SSO) token.

Cookie-based Network Edge Authentication

Unlike standard credential cache authentication, network edge authentication is global to the Hybrid Web Container, not specific to each Hybrid App. Each Hybrid Web Container has a dialog to prompt for HTTP basic authentication credentials when challenged, and a session header or cookie is returned if the system is so configured for SSO. See *HTTP Authentication Security Provider* in *Security* for more information.

The sequence of authentication is as follows:

1. Client Network Edge authentication – The client begins a session by sending an HTTP(S) request to the Reverse Proxy. The Reverse Proxy detects the un-authenticated request and challenges for Basic authentication. After the 401 challenge, the client may already have network credentials configured, or perhaps there is a callback to prompt for credentials.
2. The client sends another HTTP request with the credentials, which the Reverse Proxy validates, and if valid issues a Cookie with an SSO token value. The HTTP headers will be added to the request that is created and sent to SAP Mobile Platform.
3. SAP Mobile Platform receives the request and uses an enhanced CSI LoginModule to authenticate. This login module is configured to extract HTTP Headers from the request (Cookie values are a subset).
4. SAP Mobile Platform processes the request and a response is sent back to the client. The client is still waiting on the original HTTP request from the Reverse Proxy. When the response comes back, the Reverse Proxy typically adds the setCookie response header at this time to pass the SSO data back to the client to use in subsequent HTTP requests.
 - If the SSO token is valid, everything proceeds.
 - If the SSO token is invalid, a server to device method instructs the Hybrid Web Container to prompt for credentials again.

Configuring the Hybrid App to Use Credentials

Configure a Hybrid App to pass user credentials, which are authenticated by SAP Mobile Server and the EIS.

For information about configuring and implementing X.509 and SSO2 on the server, see *Security*.

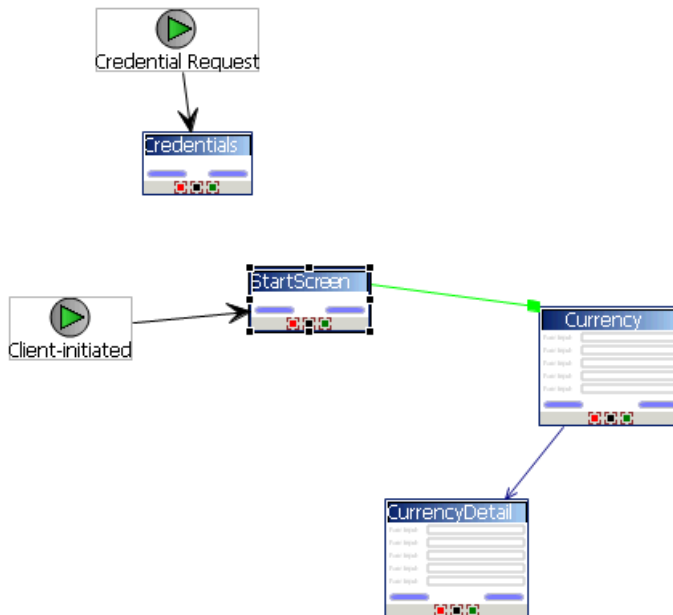
Configuring the Hybrid App to Use X.509 Credentials

Add a screen that contains a Specify Certificate Credentials menu item to the Credential Request starting point from which a Hybrid App user selects a certificate to gain access to the MBO and related resources.

1. In the Hybrid App Designer, add a **Credential Request** starting point to the Hybrid App.
2. Add a screen named **Credentials** and connect it to the Credential Request starting point.
3. Double-click **Credentials** to open it in the Screen Design. Add a **Select Certificate** menu item of the Submit type.

On the device, the Specify Certificate Credentials action prompts the user for a *.p12 certificate and passes it to SAP Mobile Server for validation.

4. Add a **Client-initiated** starting point to which you add screens that contain the Submit menu items used to run MBO operations and object queries, return and display results, and so on. These actions all use the same credentials created in the previous steps.



Configuring the Hybrid App to Use Static X.509 Credentials

When using static credentials, the Hybrid App does not prompt the user for credentials, instead it passes the credentials to SAP Mobile Server automatically and displays the Hybrid App's start screen.

1. Remove the Credential Request starting point and screen from the Hybrid App (so the client is no longer prompted for credentials).
2. From Flow Design, select **Authentication, Use static credentials, and Use certificate-based credentials**.
3. Click **Generate from Certificate**.
4. Browse to the location of the *.p12 certificate file.
5. Enter the certificate's password, select the alias, and click **OK**.
6. Save and regenerate the Hybrid App, and reassign it to a device.

Propagating a Client's Credentials to the Back-end Data Source

Use client credentials (including certificates and SSO tokens on EIS types that support them) to establish enterprise information system (EIS) connections on the client's behalf for all data source types.

To use client credentials, map an EIS connection's user name and password properties to system-defined "user name" and "password" personalization keys respectively. This creates a new connection for each client and the connection is established for each request (no connection pooling.)

1. During development of the mobile business object MBO/operation, from the data source definition page (available either in the Creation wizard or from the Properties view), in the **Runtime Data Source Credential** section (or **HTTP Basic Authentication** section for a Web Service, RESTful Web Service, or SOAP MBO), enter the client credentials in the User name and Password fields. The runtime data source credential values (user name and password) that SAP Mobile WorkSpace uses for refresh or preview operations is taken in this order:
 - a) Any literal value entered in the User name and Password fields.
 - b) User-defined personalization keys that have non-empty default values.
 - c) System personalization keys 'user name' and 'password'.
 - d) User name and password property values contained in the connection profile.
2. During deployment of the package that contains such MBOs, map the design-time connection profiles to the existing or new server connections, but be aware that the user name and password portions for the selected server connection is replaced by the user name and password propagated from the device application.

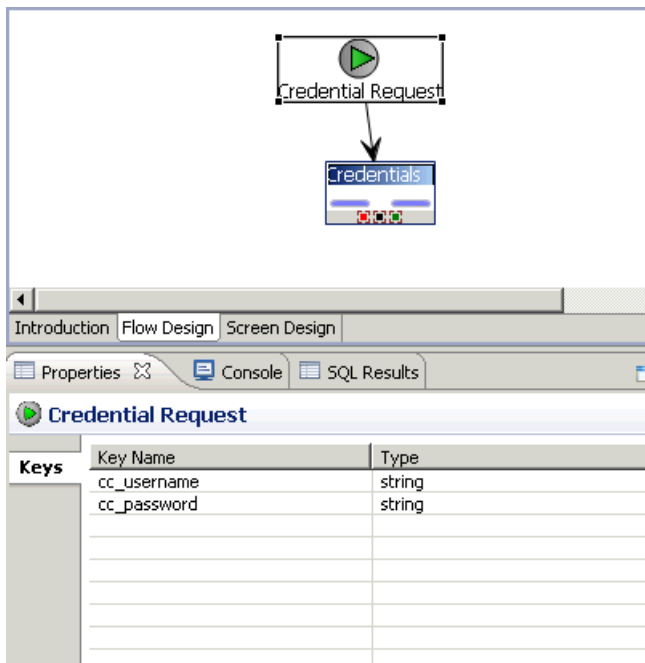
Note:

- Do not set client credentials using the Runtime Data Source Credential option for MBO's that belong to a cache group that uses a Scheduled policy, since this is unsupported.
 - In general, a MBO operation that uses data source credential settings as connection properties cannot have these settings mapped to an enterprise information system (EIS) during deployment. Instead, they maintain their original settings, which you can map after deployment using SAP Control Center.
 - When you create a new security configuration that includes the SAPSSOTokenLoginModule, and deploy it to a new domain, if the Hybrid App uses the MBOs associated with the new security configuration, you must specify an SAP Mobile Server domain that corresponds to the domain using the security configuration. See *Security* for more information about security configurations
-

Configuring a Hybrid App to Use SSO2 Tokens

Configure a Credential Request starting point from which a Hybrid App user can pass a user name and password to gain access to the MBO and related resources.

1. In the Hybrid App Designer, add a **Credential Request** starting point to the Hybrid App.
2. Add two keys to the Credential Request named `cc_username` and `cc_password`.
3. Add a screen named `Credentials` and connect it to the Credential Request starting point.

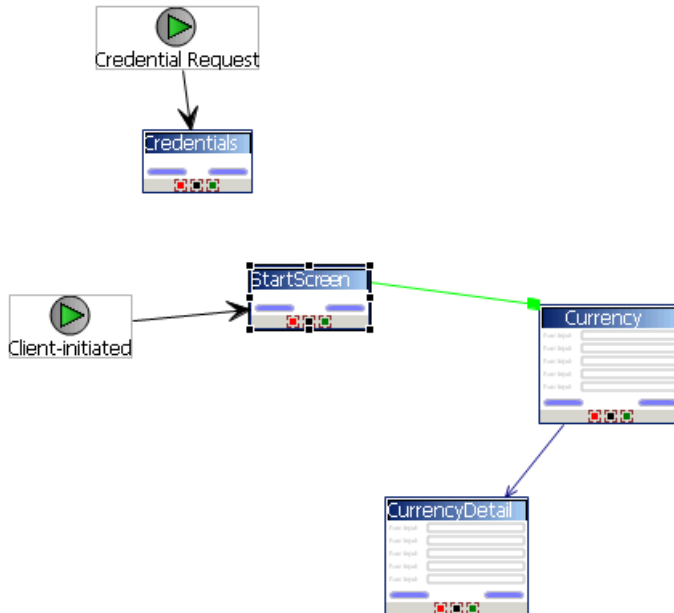


Develop a Hybrid App Using the Hybrid App Designer

4. Double-click **Credentials** to open it in the Screen Design.
5. Add a **Save screen** menu item to the Menu, and two edit boxes (Username and Password).

The Save screen saves the Username and Password entered by the Hybrid App. You could also add a **Submit** menu item instead of **Save screen**.

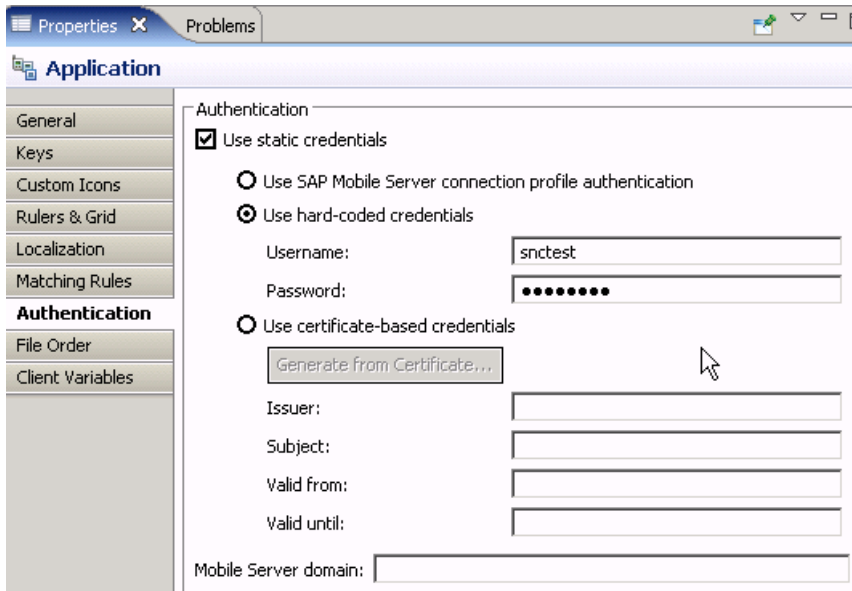
6. Add a Client-initiated starting point to which you add screens that contain the Submit menu items used to run MBO operations and object queries, return and display results, and so on. These actions all use the same credentials created in the previous steps.



Configuring the Hybrid App to Use a Static SSO2 Token

When using static credentials, the Hybrid App does not prompt the user for credentials, instead it passes the credentials to SAP Mobile Server automatically and displays the Hybrid App's start screen.

1. Remove the Credential Request starting point and screen from the Hybrid App (so the client is no longer prompted for credentials).
2. From Flow Design, select **Authentication, Use static credentials, and Use hard-coded credentials**. Enter a username and password that corresponds to those defined in SAP Control Center for the server connection (for example: sntctest/sntctest).



3. Save and regenerate the Hybrid App package, and reassign it to a device.

Modify Certificate Information for Hybrid App Packages

If using static credentials, either SSO token or static x.509 certification, you can replace the Hybrid App package certificate using either SAP Control Center or the `SUPMobileHybridApp.replaceMobileHybridAppCertificate()` API. To replace a certificate, you must have access to the certificate file and password.

Replacing the Hybrid App Certificate Through SAP Control Center

If using static credentials, you can set or modify the context variable certificate settings for a Hybrid App package from SAP Control Center.

The Hybrid App certificate password context variable is read-only. You can modify this only by using the Admin Java API method `SUPMobileHybridApp.replaceMobileHybridAppCertificate()`.

1. From SAP Control Center, navigate to **Hybrid Apps** > **<Hybrid_App_Name>**, where *Hybrid_App_Name* is the name of the Hybrid App package.
2. On the Context Variables tab, verify that `SupUser` and `SupPassword` contain valid credentials for the specified security configuration, for Hybrid App packages that do not use certificate-based authentication.
3. For Hybrid App packages that use certificate based authentication, you can view these context variables:
 - `SupCertificateIssuer`

Develop a Hybrid App Using the Hybrid App Designer

- SupCertificateSubject
- SupCertificateNotAfter
- SupCertificateNotBefore

Replacing the Hybrid App Certificate Using the Admin API

Use the `SUPMobileHybridApp.replaceMobileHybridAppCertificate()` method to set or modify the certificate password context variable for the Hybrid App package.

```
InputStream is = getClass().getResourceAsStream("sybase101.pl2");
ByteArrayOutputStream baos = new ByteArrayOutputStream();
byte[] buf = new byte[512];
int count;
while ((count = is.read(buf)) != -1) {
    baos.write(buf, 0, count);
}
is.close();
baos.flush();
baos.close();
MobileHybridAppIDVO hybridAppID = new MobileHybridAppIDVO();
hybridAppID.setWID(4);
hybridAppID.setVersion(1);

mobileHybridApp.replaceMobileHybridAppCertificate(hybridAppID,
    baos.toByteArray(), "password");
```

Content Security on Devices

This explains how the files that make up the Hybrid Web Container are protected when stored on the device, and under what circumstances the files are stored in plain text.

Content Security on Android Devices

On Android operating systems, all Hybrid Web Container files, and extra data entered by the user or retrieved from the server, are encrypted before being stored in the application's sandbox and SQLite database. You can turn off the encryption of Hybrid Web Container files to decrease the load times for Hybrid Apps by using the `disableFileEncryption` customization point.

The cryptographic libraries provided by Google/Android are used. Specifically, the encryption algorithm used is AES-256 symmetric encryption.

Hybrid Web Container Files

Hybrid Web Container files include all the files contained in the `<Hybrid_App_package_name>.zip` that is deployed to the device, including all HTML, JavaScript, CSS, and any other files that may be included as part of the ZIP package.

- When the platform's browser control requests these Web files, they are read from the device's sandbox, stored unencrypted on the file system temporarily, and then passed to the browser control through a content provider.

- These temporary files are removed from the content provider immediately after the last of them are requested by the browser control.

Note: Prepackaged files are not secured on Android. They are stored in the `assets` directory unencrypted.

Attachments

If attachments, such as *.docx, *.pdf, and so on, are part of the <Hybrid_App_package_name>.zip deployed to the device, they are stored in the application's sandbox after they have been encrypted through the Google/Android crypto libraries.

- When the JavaScript requests these attachments for viewing, they are read from the application's sandbox, and temporarily written unencrypted to the device's flash memory for the external viewers to display them.
- Once the application closes, these temporary attachment files are immediately removed.

Note: The Android operating system enforces the sandboxing of these temporary files.

Attachments that are downloaded through an online request using an object query are stored unencrypted in the device's flash memory for the file viewers to display them. Once the application closes, these temporary attachment files are immediately removed.

Images

The image is saved, unencrypted on the file system, into the Gallery application, (ImageOptions.CAMERA, ImageOptions.BOTH).

Note: The Android operating system enforces the sandboxing of these image files.

Cached Online Requests

The results of online requests that are specified to be cached are stored on the device's SQLite database (after they are encrypted through the Google/Android cryptographic libraries). Cached results are removed when the Hybrid Web Container is unassigned from the device, or uninstalled from the server.

Notifications From the Server

Notifications from the server are stored in the same SQLite database after they have been encrypted through the Google/Android cryptographic libraries, including the payload that makes up the notification. When the notification is acted upon, the JavaScript makes a request for the notification contents. This is read from the SQLite database, unencrypted, and passed to the browser in memory.

User Input Sent to the Server

When the device has no network connectivity, and the user submits a Hybrid App for the server to process, the data destined for the server is queued up on the device. The contents of this queue are again encrypted through the Google/Android cryptographic libraries before it is stored into the SQLite database.

Encryption Keys

- How the encryption key is generated:
 - A generated GUID is used as the key for encrypting the data ("data password")
 - A user-provided password (PIN) is used to secure/encrypt the "data password," which is persisted in its encrypted form. In order to have access to the "data password", one must know the user password.
 - The salt is a different persisted, generated GUID.
 - Encryption of data is done with the "data password."
- Where is the encryption key stored?
 - The "data password" is persisted in its encrypted form in a separate table in the SQLite database.

Content Security on BlackBerry Devices

In general, all Hybrid Web Container files and extra data entered by the user, or retrieved from the server, are stored on the BlackBerry device's PersistentStore.

This is the same storage area used by e-mail, calendar entries, and applications. See your BlackBerry documentation for information about persistent store APIs.

The BlackBerry Hybrid Web Container uses the RIM PersistentContent APIs when reading and writing of data from PersistentStore is required. This ensures that the content being written is stored at the device's current encryption level. See your BlackBerry documentation for information about content protection strength settings.

When content protection is turned on, content on the BlackBerry device is protected using the 256-bit Advanced Encryption Standard (AES) encryption algorithm.

- Use 256-bit AES encryption to encrypt stored data when the BlackBerry device is locked
- Use an Elliptic Curve Cryptography (ECC) public key to encrypt data that the BlackBerry device receives when it is locked

These settings apply to the encryption of data that the BlackBerry device receives while locked:

Content protection strength setting	ECC encryption key length (in bits)
Strong	160
Stronger	283
Strongest	571

The BlackBerry Hybrid Web Container also registers a PersistentContentListener, which allows it to be notified when the device's encryption level changes. This also enables previously stored content to be re-encoded to the new encryption level setting. The device's

encryption level setting can be changed by a BlackBerry Enterprise Server Administrator remotely, or by the user, from the device.

Hybrid Web Container Files

Hybrid Web Container files include all the files contained in the `<hybrid_app_name>.zip` that is deployed to the device, including all HTML, JavaScript, CSS, and any other files that may be included as part of the Hybrid App ZIP package. When the platform's browser control requests these Web files, they are read from the device's PersistentStore and passed to the browser control in memory, which means there are no temp files.

Attachments

If attachments, such as *.docx, *.pdf, and so on, are part of the `<hybrid_app_name>.zip` deployed to the device, they are stored on the device's PersistentStore:

- When the JavaScript requests to display these attachments, they are read from the PersistentStore, and temporarily written unencrypted to the device's flash memory for the external viewers to display them.
- Once the mobile Hybrid App closes, these temporary attachment files are immediately removed.

Attachments that are downloaded using an online request that use an object query are stored unencrypted in the device's flash memory for the file viewers to display them. Once the Hybrid App closes, these temporary attachment files are immediately removed.

Images

Images are stored unencrypted on the file system and saved into the `Pictures` folder (`ImageOptions.BOTH`).

Cached Online Requests

The results of online requests that are specified to be cached are stored on the device's PersistentStore. Cached results are removed when the Hybrid Web Container is unassigned from the device, or uninstalled from the server.

Notifications From the Server

Notifications from the server are stored in the same PersistentStore area, including the payload that makes up the notification. When the notification is acted upon, the JavaScript makes a request for the notification contents. This is read from the PersistentStore and passed to the browser in memory.

User Input Sent to the Server

When the device has no network connectivity, and the user submits a Hybrid App for the server to process, the data destined for the server is queued up on the device. This queue is part of the device's PersistentStore.

Content Security on iOS Devices

On iOS devices, all Hybrid Web Container files and extra data entered by the user or retrieved from the server, are stored in a SQLite database that uses the SQLite Encryption Extensions (AES-128).

Hybrid Web Container Files

Hybrid Web Container files include all the files contained in the <Hybrid_App_package_name>.zip that is deployed to the device, including all HTML, JavaScript, CSS, and any other files that may be included as part of the ZIP package. When the iOS device's browser control requests these Web files, they are read from the encrypted SQLite database. The data is temporarily written to the file system under the application sandbox, after which the browser control reads the file contents into memory. The temp files are removed when the Hybrid App closes.

Note: When using a prepackaged Hybrid App, all of the files associated with the prepackaged Hybrid App (HTML, JavaScript, CSS, and so on) exist within the sandbox in clear text.

Attachments

If attachments, such as *.docx, *.pdf, and so on, are part of the <Hybrid_App_package_name>.zip deployed to the device, they are stored in the encrypted SQLite database.

- When the JavaScript requests the attachments for viewing, they are read from the database, and temporarily written, unencrypted, to the Hybrid Web Container's sandbox for the viewer to display them.
- Once the application closes, these temporary attachment files are immediately removed.

Attachments that are downloaded using an online request that uses an object query are stored unencrypted in the Hybrid Web Container's sandbox for the file viewers to display them. Once the application closes, these temporary attachment files are removed immediately.

Images

Images are stored unencrypted in the Hybrid Web Container's sandbox, then removed when the application closes.

Cached Online Requests

The results of online requests that are specified to be cached are stored in the encrypted SQLite Database. Cached results are removed when the Hybrid Web Container is unassigned from the device, or uninstalled from the server.

Notifications From the Server

Notifications from the server are stored in the same encrypted SQLite database, including the payload that makes up the notification. When the notification is acted upon, the JavaScript makes a request for the notification contents. This is read from the SQLite database and passed to the browser in memory.

User Input Sent to the Server

When the device has no network connectivity, and the user submits an application for the server to process, the data destined for the server is queued up on the device. This queue is again part of the encrypted SQLite database.

Encryption Keys

- The Hybrid Web Container generates a hash from the password entered by the user, and a salt, combined
- The Hybrid Web Container generates a random key
- The Hybrid Web Container encrypts the key with the hash and stores it in the app area of the keychain

Content Security on Windows Mobile Devices

On Windows Mobile Professional, Hybrid Web Container files are stored unencrypted on the device's file system, and Hybrid Web Container settings are stored unencrypted in the device's registry.

Note: The Windows Mobile Hybrid Web Container defers all security and encryption responsibilities to the Afaria® Security Manager; therefore, SAP strongly recommends that you use Afaria Security Manager.

If you do not use Afaria Security Manager, you must:

- Protect these files through alternative means. The `\Program Files\SAP\Messaging\AMP` folder (and all of its sub folders) must be secured on the device.
- To protect the Hybrid Web Container settings, the `[HKEY_LOCAL_MACHINE\Software\SAP\MessagingClientLib]` registry key (and all of its sub keys) must be secured on the device.

Hybrid Web Container Files

Hybrid Web Container files include all the files contained in the `<hybrid_app_name>.zip` that is deployed to the device, including all HTML, JavaScript, CSS, and any other files that may be included as part of the Hybrid App zip package. These are all stored unencrypted on the file system of the device.

Attachments

If attachments, such as *.docx, *.pdf, and so on, are part of the `<hybrid_app_name>.zip` deployed to the device, they are stored unencrypted on the file system of the device.

- When the JavaScript requests these attachments for viewing, a file URI is constructed for a suitable external viewer to display these files.
- Once the Hybrid App closes, these temporary attachment files are immediately removed.

Images

Images are stored unencrypted on the file system, then removed when the Hybrid App closes.

Cached Online Requests

The results of online requests that are specified to be cached are stored unencrypted on the device's file system. Cached results are removed when the Hybrid Web Container is unassigned from the device, or uninstalled from the server.

Notifications From the Server

Server notifications are stored unencrypted in the Inbox database of the device (the same database that houses the device's regular e-mail messages). When the notification is acted upon, the JavaScript makes a request for the notification contents. This is read from the Inbox database and passed to the browser in memory. If you are not using Afaria Security Manager, the Windows Mobile Inbox database must be secured.

User Input Sent to the Server

When the device has no network connectivity, and the user submits a Hybrid App for the server to process, the data destined for the server is queued up on the device. The contents of this queue are stored in an unencrypted SQLite database.

Localization and Internationalization

You can localize different objects in the Hybrid App Designer, such as the names of screen controls, screens, and mobile business objects.

You can localize the Hybrid App by creating locale properties files. You can then load, update, and generate localized Hybrid Apps.

All the localizable strings in the Hybrid App Designer XML model work as resource keys in the localization properties file. All the localization properties files are in the same directory as the Hybrid App packages (.xbw files).

Resource keys are divided into these categories, which include all the elements of the Hybrid App Designer XML model:

- Menus
- Controls
- Screens

Localization consists of two levels of localization—the Hybrid App Designer XML model localization and the Hybrid App client localization.

All locale properties files are saved in the same directory as the Hybrid App package.

To ensure that the correct locale is picked up for the Hybrid Web Container, the following mechanism is used:

1. If a precise match is found for language and country, for example, English - United States (en-us) is the locale and the file exists in `html\en-us\hybridapp*.html`, that file is used and the HTTP lang parameter is set to "en-us."
2. If a precise match for country is not found, the language is used. For example, English (en). If the file exists in `html\en\hybridapp*.html`, that file is used and the HTTP lang parameter is set to "en."
3. If a language match is not found, the default locale is used. If the file exists in `html\default\hybridapp*.html`, that file is used and the HTTP lang parameter is set to "default";
4. If a default match is not found, no locale is used. If the file exists in `html\hybridapp*.html`, that file is used and the HTTP lang parameter is set to "".

Localization Limitations

Locale properties files have some restrictions.

These restrictions apply:

- Traditional Chinese characters are not supported on iOS.
- Hybrid Apps that have names that begin with numbers or special characters cannot be localized; you will receive an error when you generate the code. Make sure that any Hybrid App you want to localize does not have a file name that begins with a number or special character.
- When you specify a country for the language, the basic language locale must also be available. For example, if you create a locale and specify English as the language and the United States as the country, then a locale for English (the basic language) must also be available.
- If you create a locale that specifies language, country, and variant, the locale for the basic language and the locale for the basic language and the country must be available. For example, if you create a locale and specify English as the language, United States as the country, and WIN as the variant, then English (United States) and English locales must also be available.
- The language code must be a 2-letter code, and the country code can be either a 2-letter or 3-letter code.

Note: BlackBerry 9800 Asia simulators do not have a place to specify a country name, so you can specify only a language.

- If you specify a variant, the country code must be a 2-letter code.

Localizing a Hybrid App Package

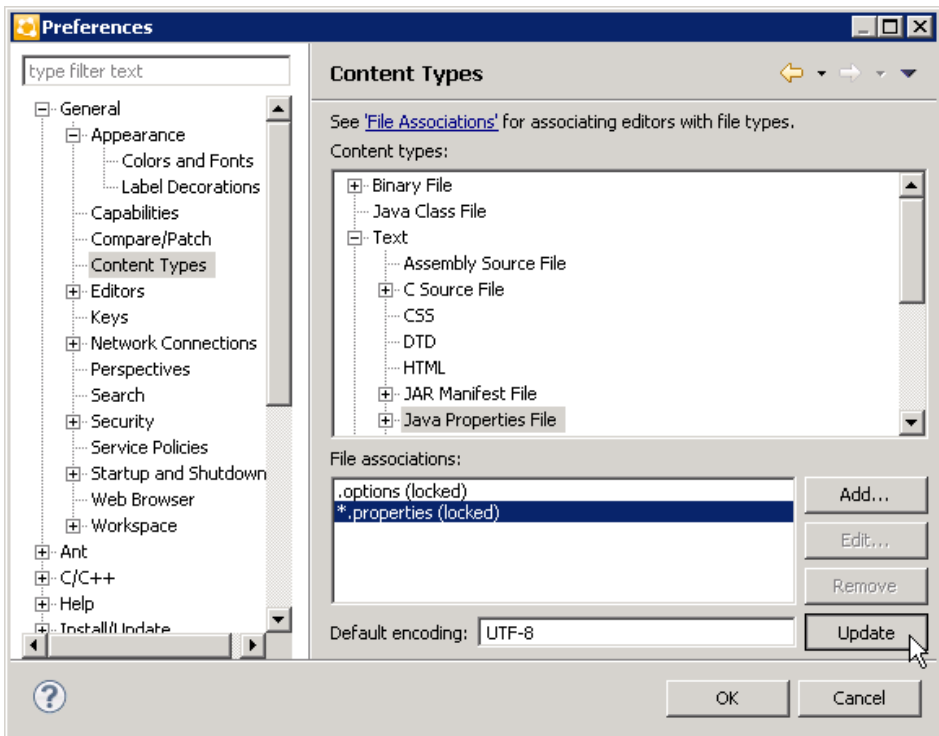
Use the Hybrid App Designer to complete these tasks to localize Hybrid App packages (.xbw files).

Changing the Encoding Type

Change the encoding type in Preferences.

If you manually localize the locale properties file using an external editor, you must make sure the file is encoded in ASCII, so that the content can be correctly read and converted to Unicode. The localization file is encoded in standard ISO-8859-1. All non-ASCII character values are converted to escaped Unicode hexadecimal values before they are written to the properties files. Before translating the localization file, select the correct file encoding option, for example UTF-8.

1. In SAP Mobile Platform, select **Window > Preferences**.
2. Expand **General > Content Types**.
3. In the right pane, select, **Text > Java Properties File**.
4. In the **File Associations** list, select `*.properties`.
5. In the Default encoding field, change ISO-8859-1 to **UTF-8**, and click **Update**.



Creating and Validating a New Locale Properties File

Create a locale properties file as the default locale.

Prerequisites

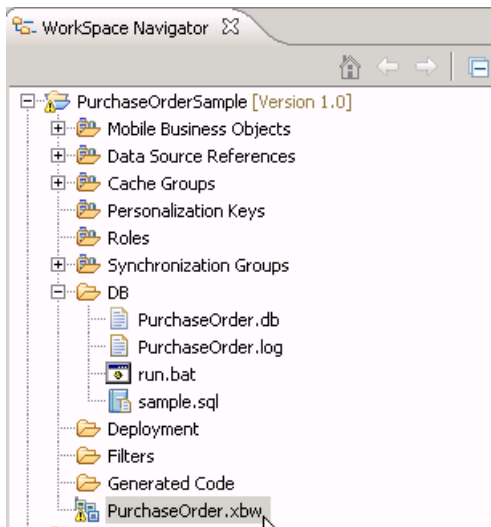
You must have an existing Hybrid App package before you create the locale properties file.

Task

When you create a new locale, keep in mind:

- When you specify a country for the language, the basic language locale must also be available. For example, if you create a locale and specify English as the language, then there must also be a locale for English (the basic language).
- If you create a locale that specifies language, country, and variant, the locale for the basic language and the locale for the basic language and the country must be available. For example, if you create a locale and specify English as the language, United States as the country, and WIN as the variant, then English (United States) and English locales must also be available.

1. In Workspace Navigator, double-click the *Hybrid App.xbw* file to open the Hybrid App Designer.



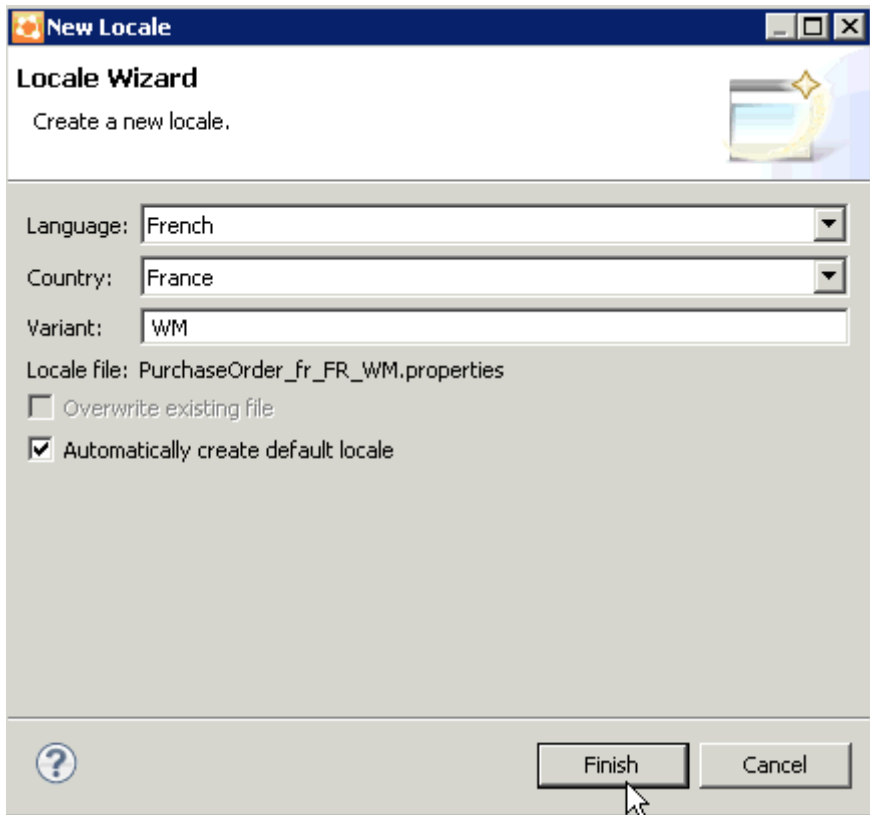
2. Click the **Flow Design** tab.
3. Right-click in a blank area on the Flow Design page, and select **Show Properties View**.
4. In the Properties view, on the left, click the **Localization** tab.
5. In the right pane, click **New**.
6. Select or enter the information for the new locale, select **Automatically create default locale**, and click **Finish**.

Develop a Hybrid App Using the Hybrid App Designer

Option	Description
Language	Select the language.
Country	Select the country.
Variant	Enter the variant, which is the vendor or browser-specific code. For example, enter WIN for Windows, MAC for Macintosh and POSIX for POSIX. If there are two variants, separate them with an underscore, and put the most important one first. For example, a Traditional Spanish collation might construct a locale with parameters for language, country, and variant as: es, ES, Traditional_WIN.
Overwrite existing file	Overwrite an existing localization file.
Automatically create default locale	Automatically create the default locale properties file. For example, if you specify the language as English and the country as United States for a device application called test, then both test_en_us.properties and test.properties files are created.

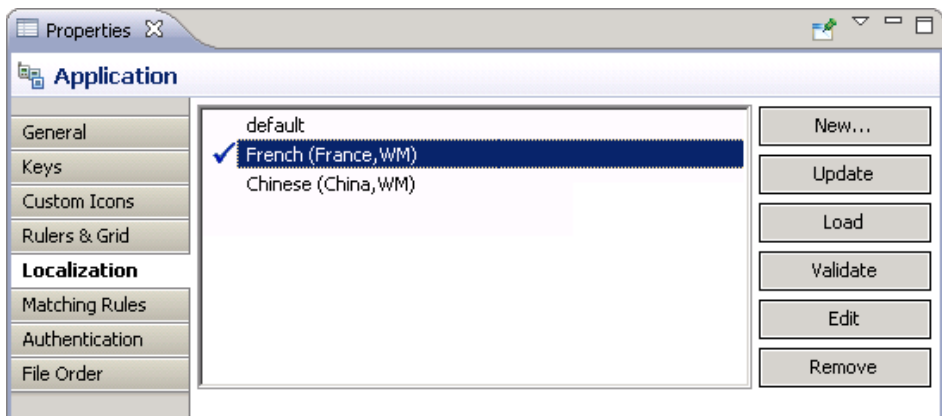
For example:

- Language – select **French**.
- Country – select **France**.
- Variant – enter a value to make this locale file unique from others, for example, WM for Windows Mobile.



This locale file is now the default locale file, and will be used when the regional setting of the device does not match that of any supplied locale file.

7. In the Properties view, in the Localization page, select the file to validate and click **Validate**.



Develop a Hybrid App Using the Hybrid App Designer

The properties file is scanned and if there are any errors, a dialog appears. Click **Yes** to correct the errors automatically; click **No** to see the errors in the Problems view.

Editing the Locale Properties File

Edit the locale properties file.

1. In WorkSpace Navigator, under the Generated Code folder, right-click the locale properties file you created, and select **Open With > Properties File Editor**.
2. You can make and save changes to the file in the Properties File editor, for example, you can replace all the values of the resource keys with Chinese characters.
3. Select **File > Save**.
The next time you open the locale properties file, notice that all of the ASCII characters have been changed.
4. In the Localization pane, select the localization file you edited, and click **Load**.
The elements of the application in the editor are translated into the language you specified if the localization file passes the loading validation.

Removing a Locale

Remove locale properties files.

1. In the Screen Design page Properties view, click **Localization**.
2. Select the locale to remove and click **Remove**.
3. Click **Yes** to confirm the deletion.

Updating the Current Locale

Update the currently loaded locale properties file with the resource keys from the current Hybrid App Designer.

If the locale properties file does not already exist, it is created. If the current locale is not defined in the Hybrid App file, the updated locale is used as the default, and the file name is *{device_application}.properties*. Otherwise, the locale defined in the Hybrid App file is updated.

Note: When you update the localization bundle, it removes all resources that are not explicitly bound to existing UI elements (screens, menuitems, controls, and so on). If you want to manually supply resources, you must do so after updating, and be careful not to update the resource bundles afterwards, or you will have to re-add those manually-supplied resources after updating.

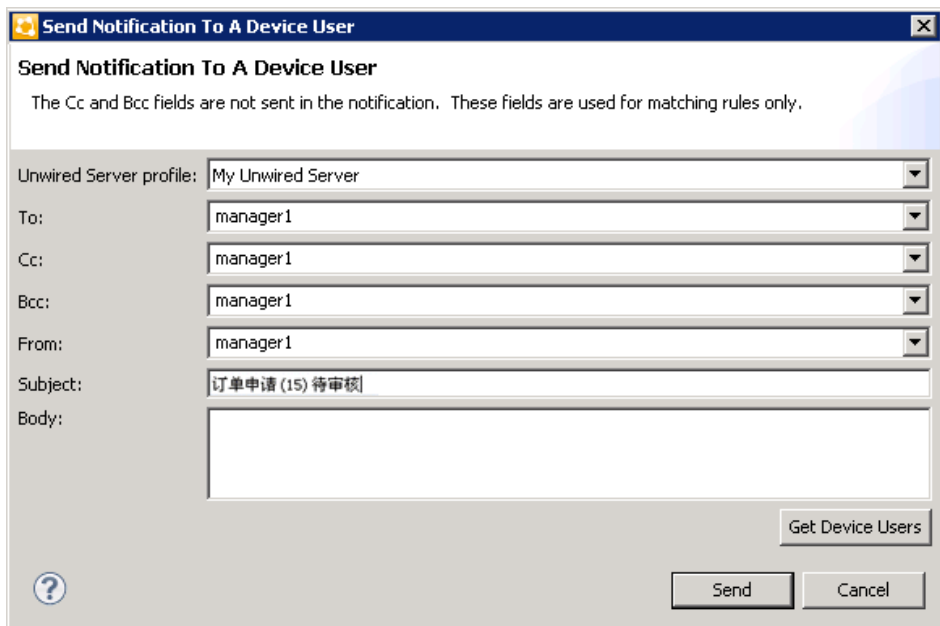
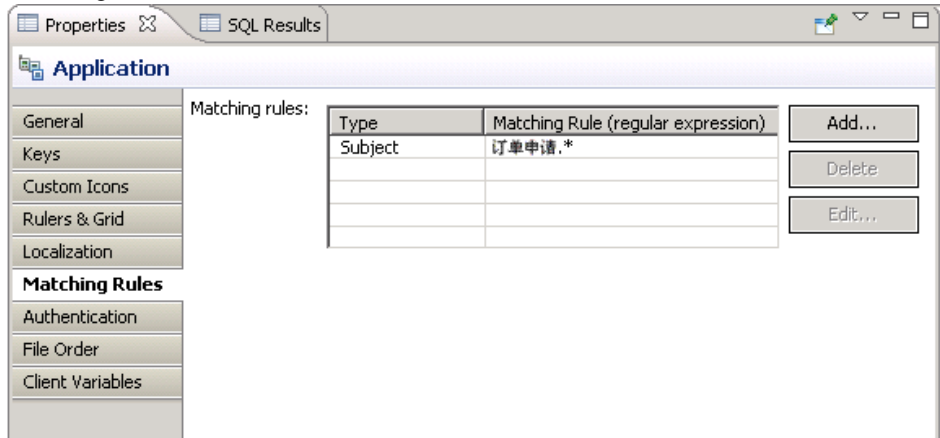
1. In the Screen Design page Properties view, click **Localization**.
2. Click **Update**.

Hybrid App Package Internationalization

The internationalization feature depends on the internationalization setting on the operating system where SAP Mobile Platform Hybrid App is running.

In the Hybrid App Designer, you can use international data in:

- Matching rules for notifications.



- Key names – you can create keys with names in other languages and map them to mobile business object parameters.

Develop a Hybrid App Using the Hybrid App Designer

Key

Specify the name of the key and, optionally, the input data binding for the key as well.

Name:

Type:

Sent by server

Input Data Binding

Mobile business object:

Mobile business object attribute

Name:

Convert to UTC

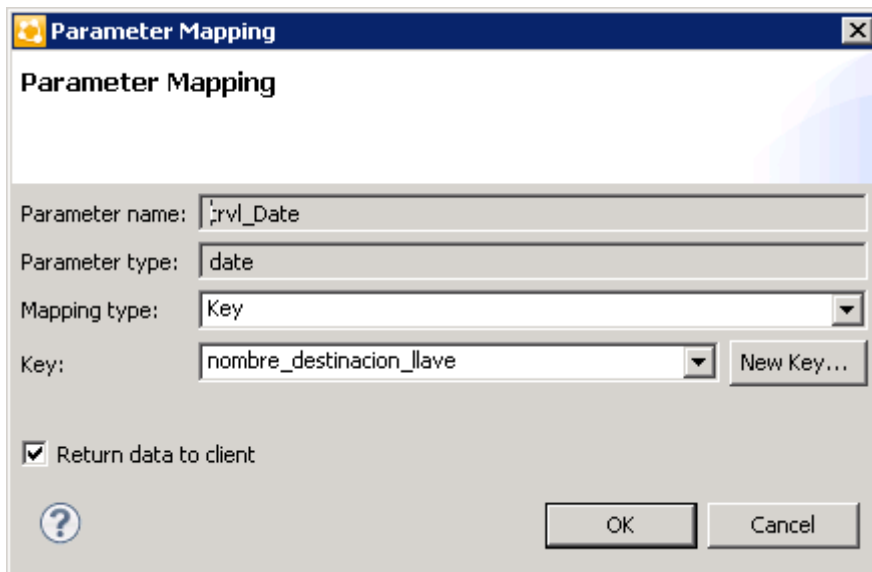
Mobile business object relationship

MBO object query results

Hard-coded value

User-defined

Extraction rule



- Generated Code folder – you can include languages other than English in the code generation path based on the name of the selected language.

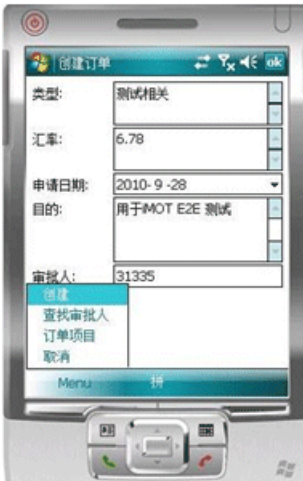
Internationalization on the Device

On the device, e-mail messages and data can include languages other than English.

The internationalization feature depends on the internationalization setting on the device where the Hybrid App client running.

E-mail messages can be sent and received using Chinese, for example, which can then be used to extract the parameter. You can also create and update records in using international data, such as Chinese. For example:

Develop a Hybrid App Using the Hybrid App Designer



Test Hybrid App Packages

Test a Hybrid App on a device or simulator.

1. Launch and/or connect to the mobile device or emulator.
2. Deploy the Hybrid App package to the device.
3. Establish the connection to the server on the device.
4. For user-initiated Hybrid App packages, go to the Hybrid Apps menu and click on the appropriate Hybrid App.

5. For e-mail subscription Hybrid App packages, send the e-mail to the device, either automatically, for example, database trigger, or manually, through the Send E-mail dialog; then open that e-mail on the device.
6. Enter data and execute menu items appropriately.
7. Verify that the backend is updated correctly.
8. Check the logs.

Testing Server-Initiated Hybrid App Packages

Test a server-initiated Hybrid App package.

1. In the Hybrid App Designer, open the Hybrid App <hybridapp>.xbw.
2. Click **Flow Design**.
3. Right-click in the editor, and select **Send a notification**.
4. In the Send a Notification window:
 - a) Select the SAP Mobile Server profile and click **Get Device Users**.
 - b) Choose the desired user and fill in the fields according to the matching rules specified when creating the Hybrid App.
5. Click **Send**.
6. On the client, from the applications screen, open the Hybrid Web Container.
7. In the client application, click **Hybrid Apps**. This contains the server-initiated Hybrid App.

Viewing Hybrid App Messages on the Device

Where Hybrid App messages that are sent to the device appear varies by platform.

Note: Registration must be successfully completed either through providing an activation code or a password for automatic registration in the connection settings before any Hybrid App packages appear on the device.

Android and BlackBerry

To see Hybrid App messages on BlackBerry devices and simulators:

1. In the applications screen, open **Hybrid Web Container**.
2. Messages appear in the Messages screen.

iOS

To see Hybrid App messages on iOS devices and simulators:

1. Open the **Hybrid Web Container**.
2. Click **Messages** to view messages.

Windows Mobile

To see Hybrid App messages on Windows Mobile devices and emulators:

Develop a Hybrid App Using the Hybrid App Designer

1. In the Programs screen, open the **Hybrid Web Container**.
2. Messages appear in the Messages screen.

Launching a Server-initiated Hybrid App on the Device

Server-initiated Hybrid App messages are sent to the Hybrid Web Container that is installed on the device.

When you click the **Hybrid Apps** menu item in the Hybrid Web Container, only the latest version of the Hybrid Apps appear. When you click the icon for a particular Hybrid App, the Hybrid App version that is associated with the notification is launched, whether it is the latest version or not.

Example

You develop a Hybrid App that has both client-initiated and server-initiated starting points. You deploy the initial version, which is called version 1, and a notification is sent.

Next, make some changes and deploy a second version, version 2. Again, a notification is sent.

There are now three ways that this Hybrid App can be launched, and the way that it is launched determines which version of the Hybrid App is launched:

- If you launch the application from the **Hybrid Apps** menu item, the last deployed version of the Hybrid App, in this case, version 2, is launched. Although version 1 of the Hybrid App still exists somewhere on the device, it is never used as long as you launch the Hybrid App from the Hybrid Apps menu.
- If you launch the Hybrid App by opening the initial notification, the version that corresponds with the latest version that existed at the time the notification was sent, is used. In this case, that is version 1; it does not matter that a later version (version 2) exists.
- If you launch the Hybrid App by opening the second notification, the version that corresponds with the latest version that existed at the time the notification was sent is used. In this case, that is version 2.

Debugging Custom Code

Debug the Hybrid App package HTML and JavaScript files using a Windows desktop browser.

This procedure uses Google Chrome as an example, but you can use any browser that supports JavaScript debugging.

1. Change the tracing level of Hybrid App to Debug.
2. Open the browser to use for debugging and open the Java Console.

If you are using Chrome:

- a) Add this command line option to the shortcut used to start Chrome:

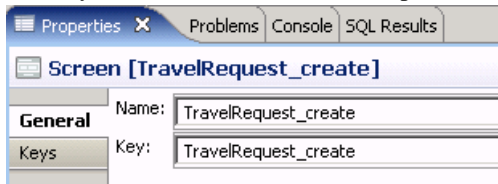
```
..\chrome.exe" --allow-file-access-from-files
```

3. You can debug a client-initiated Hybrid App up until the point where a menu item of the Submit type is performed. If the menu item action is an Online Request, place the XMLWidgetMessage (available in the WorkflowClient trace log located in `SMP_HOME\Servers\UnwiredServer\logs\WorkflowClient`) that is the expected response message into an `rmi.xml` file and place it at the same level as the generated `hybridapp.html` file.

Note: Control characters are not parsed correctly when using `rmi.xml` and Chrome to debug the Hybrid App. Do not format the content of the `rmi.xml` when debugging the Hybrid App using Chrome. If you want a formatted look at the `rmi.xml` file, make a copy of the file for that purpose.

4. From WorkSpace Navigator, drag and drop the `hybridapp.html` file for the Hybrid App to debug onto the browser window.
5. Find the name of the key to debug:
 - a) In Flow Design, click the screen to debug.
 - b) In the Properties view, click **General** in the left pane.

The key name is shown, in this example, that is `TravelRequest_create`.



6. In the URL, add the `?screenToShow=<Screen_name>` parameter to the end of the URL, for example:


```
file:///C:/Documents%20and%20Settings/<user_name>/workspace/HybridApp101/Generated%20HybridApp/travelrequest/html/hybridapp.html?screenToShow=TravelRequest_create
```
7. To simulate an e-mail message triggered Hybrid App:
 - a) Create a file called `transform.xml` and place the contents of the XMLWidgetMessage into it.

The contents of the XMLWidgetMessage are in the WorkflowClient trace log in `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\logs\WorkflowClient`.
 - b) To provide data to the Hybrid App you are debugging, place the `transform.xml` file at the same level as the generated `hybridapp.html` file (Generated Hybrid App\`<Hybrid_App_name>`\html).
 - c) Add a `?loadtransformdata=true` parameter to load the data into the Hybrid App.

Configuring Messaging Server Log Settings

Messaging Server logs create trace configurations for messaging modules, and retrieve trace data for all or specific messages. Configure trace configuration properties for modules to

specify the amount of detail that is written to the log. You can configure trace settings for the primary server cluster in SAP Control Center for each module. The settings are available to cluster servers through the shared data folder.

Note: The default settings may only need to change in case of technical support situations where, for diagnostic reasons, a request is made to configure the specific module(s) settings, and provide the request log. In all other cases, the administrator or developer should not need to change the settings.

Additionally, you should always use SAP Control Center to configure server logs. If you manually edit the configuration file, especially on secondary servers in a cluster, the servers may not restart correctly once shut down.

1. In the SAP Control Center left navigation pane, click **Configuration**.
2. In the right administration pane, click the **Log Setting** tab and select **Messaging Server**.
3. Select Default, or one or more of the messaging service modules. Click **Show All** to show all modules.

Module	Description
Default	Represents the default configuration. The default values are used if optional fields are left blank in a module trace configuration. Required.
Device Management	Miscellaneous functions related to device registration, event notification, and device administration. Enable tracing for problems in these areas.
JMSBridge	This module handles communications from the SAP Mobile Server to the messaging server. Enable tracing to view the detailed messaging exchange.
MO	This module handles the delivery of messages between the client and server, including synchronous function calls from client to server. Enable tracing for MO errors and message delivery issues.
SUPBridge	This module handles communications from the messaging server to the SAP Mobile Server. Enable tracing to view the detailed messaging exchange.

Module	Description
TM	This module handles the wire protocol, including encryption, compression, and authentication, between the messaging server and clients. All communication between the client and the messaging server passes through TM. Enable tracing for authentication issues, TM errors, and general connectivity issues.
WorkflowClient	The WorkflowClient module.

4. Click **Properties**.

- a) Enter trace configuration properties. If you selected multiple modules, a string of asterisks is used to indicate settings differ for the selected modules. You can select the option to view or change the property value for any module.

Property	Description
Module	Display only. Default, module name, or list of module names selected.
Description	(Optional) Custom description of the server module.
Level	Trace level for the module - DISABLED, ERROR, WARN, INFO, DEBUG, DEFAULT. If the default trace level is specified for the module, the module uses the trace level defined for Default. Required.
Max trace file size	(Optional) Maximum trace file size in MB. If the trace file size grows larger than the specified value, the trace file data is backed up automatically.
User name	(Optional) Only data for the specified user name is traced.
Application Connection ID	(Optional) Only data for the specified Application ID is traced.

- b) Click **OK**.

Log files for each module are stored in folders of the same name located in:
`SMP_HOME\Servers\UnwiredServer\logs.`

Develop a Hybrid App Using the Hybrid App Designer

Manage a Hybrid App Package

The Hybrid Apps node in SAP Control Center allows administrators to view and manage deployed Hybrid App packages, including display name, module name, and module version.

Administrators deploy Hybrid App packages into the SAP Mobile Platform cluster through this node, as well as manage notification settings configuration.

Registering or Reregistering Application Connections

Registering an application connection groups the user, device, and application to create a unique connection in SAP Control Center, so the registered connection activity can be monitored. Use SAP Control Center to manually register an application connection. You can also reregister an application connection when the association between the user, device and application breaks or requires a different pairing.

For more information on registering and reregistering application connections, see *How Connections Are Registered in Mobile Application Life Cycle*.

1. In the left navigation pane, click the **Applications** node.
2. In the right administration pane, click the **Application Connections** tab.
3. Choose an action:
 - Click **Register** to register a new application connection. Using the Activation Code, this application is then paired with a user and a device.
 - Click **Reregister** to associate the application with a new device and user pairing. For example, reregister the application connection if someone loses their device. By reregistering the application connection, the user then receives the same applications and workflows as the previous device.

Note: If the client application does not support reregistration, you cannot reregister the application connection. To determine if the client application supports reregistration, review the **Capabilities** properties for the application connection. If the **Application Supports Client Callable Components** property has a value of `False`, reregistration is not supported.

4. In the Register Application Connection or the Reregister Application Connection dialog.
 - a) For new device registration only, type the name of the user that will activate and register the device. For reregistrations or clones, the same name is used and cannot be changed.
 - b) (Not applicable to reregistration.) Select the name of the template for initial application connection registration. The template you use supplies initial values in the subsequent fields.

- Default – a default template that you can use as is, or customize.
- HWC – a default template for Hybrid Web Container. Use as is, or customize. If you use the HWC template, Application ID must be set to HWC.
- Custom - customized templates are listed.

Note: You cannot change the application connection template for an application connection after registration.

5. Change the default field values for the template you have chosen.

If you are using Relay Server, ensure the correct values are used.

- **Application ID**- the application ID registered for the application. The value differs according to application client type - native application, Hybrid App, or Online Data Proxy client. See *Application ID Overview* for guidelines.

Note: If the template you have chosen supplies the Application ID, then this field is read-only.

- **Security Configuration**- select the security configuration relevant for the application connection.
- **Logical Role**- (not applicable to reregistration) select the logical role that users must belong to in order to access the application.
- **Domain**- select the domain to which the application connection is assigned. A domain is not required for registering application connections for Hybrid Web Container applications.

Note: This value is sent to and used by the device application, and is automatically derived from the application ID you select. Therefore, you must set this value correctly when using a domain with an application ID. If you set a domain, ensure it matches the domain of the packages needed by the application; otherwise, the application generates a `Package not found` error.

- **Activation code length** - the number of characters in the activation code.
- **Activation expiration**- the number of hours the activation code is valid.

6. (Optional) Select the check box adjacent to **Specify activation code** to enter the code sent to the user in the activation e-mail. This value can contain letter A - Z (uppercase or lowercase), numbers 0 - 9, or a combination of both. Acceptable range: 1 to 10 characters.

7. Click **OK**

The application is registered or reregistered. SAP applications that have connections registered with SAP Mobile Server, can also have licenses counted by SAP License Audit service. For a list of SAP applications for which licenses are counted, see *SAP Applications Tracked with SAP License Audit* in *System Administration*.

Setting General Application Properties

Provide general application properties such as the application ID, description, security configuration and domain details while registering the application.

1. In the Application Creation Wizard, enter a unique **Application ID**.

Note:

- SAP recommends that application IDs contain a minimum of two dots ("."). For example, the following ID is valid: `com.sybase.mobile.appl`.
 - Application IDs cannot start with a dot ("."). For example, the following ID is invalid: `.com.sybase.mobile.appl`.
 - Application IDs cannot have two consecutive dots ("."). For example, the following ID is invalid: `com..sybase.mobile.appl`.
-

2. Enter a **Display name** and **Description** for the application.
3. Select the appropriate security configuration from the **Security Configuration** drop-down list.

For applications that do not require authentication, select the **anonymous** security configuration or the **Anonymous access** checkbox.

4. Select the appropriate domain from the **Domain** drop-down list.
5. (Optional) Assign one or more packages as desired.

Note: When an application ID is intended for use by Online Data Proxy, packages do not need to be assigned. .

6. (Optional) Modify application connection template settings.
 - a) Select **Configure additional settings**, and click **Next**.
 - b) To reuse the configuration of an existing template, select a **Base template** from the drop-down list.
 - c) Configure the application connection template properties as required.

Note: ODP applications require a proxy type connection endpoint. When modifying application connection template settings for an ODP application, you can automatically create the proxy connection endpoint by entering an OData URL as the Application Endpoint value in the connection template Proxy properties. This creates a proxy connection endpoint with the same name as the Application ID. If the ODP application uses an anonymous security configuration, the newly created connection endpoint will have the Allow Anonymous Access property set to True and the Address (URL) property set to the Application ID. If you want to create the proxy connection endpoint manually, leave the Application Endpoint property empty. You manually create the proxy connection endpoint through the SAP Control Center Domains node.

7. Click **Finish** to register the application with the configured settings.

Application ID and Template Guidelines

Choose an appropriate application ID while registering application connection for use by native MBO, Hybrid App, or Online Data Proxy clients. Using an incorrect application ID results in failure when the client tries to activate itself.

Application Type	Guidelines
Hybrid App	<ul style="list-style-type: none"> • 2.0.1 or earlier – leave the application ID empty. • 2.1 or later – use preexisting HWC template, or, if you are using your own template, make sure that HWC is set as the application ID in the template. • iOS sample container 2.1 or later – use the template you have created. The application ID used by the iOS sample container should match the application ID specified in registration.
Native MBO application	<ul style="list-style-type: none"> • Previous to 2.1.2 – leave the application ID empty. This applies to native messaging-based application clients. • 2.1.2 or later – (recommended) use the application connection template that is automatically created for the application. Otherwise, ensure you register the application connection with the correct template by verifying that application ID matches, and that the correct security configuration and domain are selected. Also, if using replication, set other template properties (such as synchronization-related properties in Connection category) as required. For Android native MBO applications, this recommendation applies starting with version 2.1.1.
Online Data Proxy	<p>Register the application connection using the template created for the application. Existing templates can be found in the Applications > Application Connection Template tab.</p>

Enabling and Configuring the Notification Mailbox

Configure the notification mailbox settings that allow SAP Mobile Server to transform e-mail messages into Hybrid App.

The notification mailbox configuration uses a listener to scan all incoming e-mail messages delivered to the particular inbox specified during configuration. When the listener identifies an e-mail message that matches the rules specified by the administrator, it sends the message as a Hybrid App to the device that matches the rule.

Note: Saving changes to the notification mailbox configuration deletes all e-mail messages from the account. Before proceeding with configuration changes, consult your e-mail administrator if you want to back up the existing messages in the configured account.

1. Log in to SAP Control Center.
2. In the left navigation pane, click **Hybrid Apps**.
3. In the right administration pane, click **Notification Mailbox**.
4. Select **Enable**.
5. Configure these properties:
 - **Protocol** – choose between POP3 or IMAP, depending on the e-mail server used.
 - **Use SSL** – encrypt the connection between SAP Mobile Server and the e-mail server in your environment.
 - **Server** and **Port** – configure these connection properties so SAP Mobile Server can connect to the e-mail server in your environment. The defaults are localhost and port 110 (unencrypted) or 995 (encrypted).
 - **User name** and **Password** – configure these login properties so SAP Mobile Server can log in with a valid e-mail user identity.
 - **Truncation limit** – specify the maximum number of characters taken from the body text of the original e-mail message, and downloaded to the client during synchronization. If the body exceeds this number of characters, the listener truncates the body text to the number of specified characters before distributing it. The default is 5000 characters.
 - **Poll seconds** – the number of seconds the listener sleeps between polls. During each poll, the listener checks the master inbox for new e-mail messages to process. The default is 60 seconds.
6. If you have added at least one distribution rule, you can click **Test** to test your configuration. If the test is successful, click **Save**.

Assigning and Unassigning a Hybrid App to an Application Connection

Assign a Hybrid App package to an application connection to make it available to a device user. Unassign the Hybrid App package when it is no longer required.

You can also assign Hybrid App packages indirectly through the application connection template. The set of packages assigned to an application connection will be a combination of packages assigned indirectly through the application connection template and directly through the application connection.

1. In the left navigation pane of SAP Control Center, click **Hybrid Apps** and select the Hybrid App to assign.
2. In the right administration pane, click the **Application Connections** tab.
3. Click **Assign**.
4. List the activation users to assign the Hybrid App package to.
Search for users by selecting the user property you want to search on, then selecting the string to match against. Click **Go** to display the users.
5. Select the user or users from the list that you want to assign the Hybrid App package to.
6. Click **OK**.
7. To set the Hybrid App package as the default application for an application connection, select the connection and click **default**.
Set a Hybrid App package as the default to run that application on the device as a single-purpose application. Single-purpose applications launch automatically when the user opens the Hybrid Web Container. This will be the only Hybrid App available on the device. You can select only one default per application connection.
8. To unassign a Hybrid App package, select the application connection and click **Unassign**.

Note: If you unassign the Hybrid App package that is set as the default, you may want to select a new default package.

9. Click **OK**.

Activating the Hybrid App

Hybrid App screen menus contain two menu item types: **Submit Hybrid App** (asynchronous) and **Online Request** (synchronous).

To complete the Hybrid App activation process, the last screen in the Hybrid App must have a **Submit Hybrid App** menu item. This is necessary for the device and server-side to activate the Hybrid App for the device.

A Hybrid App is considered to have been processed or activated only if it is closed with a **Submit Hybrid App** menu item, which may or may not be tied to a mobile business object (MBO).

Configuring Context Variables for Hybrid App Packages

The administrator can change some of the values of a selected variable, should the design-time value need to change for a production environment.

Which values are configurable depends on whether the developer hard-coded a set of user credentials or used a certificate.

1. In the left navigation pane, expand the **Hybrid Apps** folder and select the Hybrid App package to configure context variables for.
2. In the right administration pane, click the **Context Variables** tab.
3. Select the context variable to configure, then click **Modify**.

Context Variable	Description
SupUser	The valid Hybrid App application user ID that SAP Mobile Server uses to authenticate the user. Depending on the security configuration, SAP Mobile Server may pass that authentication to an EIS.
SupDomain	The name of the domain that the Hybrid App package is deployed to.
SupUnrecoverableErrorRetryTimeout	After sending a JSON request to SAP Mobile Server, if you receive an EIS code that indicates an unrecoverable error in the response log, the Hybrid App client throws an exception. A retry attempt is made after a retry time interval, which is set to three days by default. Select this property to change the retry time interval.
SupThrowCredentialRequestOn401Error	The default is true , which means that an error code 401 throws a <code>CredentialRequestException</code> , which sends a credential request notification to the user's inbox. If this property is set to false , error code 401 is treated as a normal recoverable exception.

Context Variable	Description
SupThrowBadHttpHeadersOn412Error	The default is true , which means that an error code 412 throws a <code>BadHttpRequestException</code> . If this property is set to false , error code 412 is treated as a normal recoverable exception.
SupRecoverableErrorRetryTimeout	After sending a JSON request to SAP Mobile Server, if you receive an EIS code that indicates a recoverable error in the response log, the Hybrid App client throws an exception. A retry attempt is made after a retry time interval, which is set to 15 minutes by default. Select this property to change the retry time interval.
SupPassword	The valid Hybrid App application user password that SAP Mobile Server uses to authenticate the user. Depending on the security configuration, SAP Mobile Server may pass that authentication to an EIS. An administrator must change development/test values to those required for a production environment.
SupPackages	The name and version of the MBO packages that are used in the Hybrid App. This cannot be changed.
SupMaximumMessageLength	Use this property to increase the allowed maximum Hybrid App message size. Limitations vary depending on device platform: <ul style="list-style-type: none"> • For BlackBerry 5, the limit is 512 bytes. • For Windows Mobile the limit is 500 bytes. • For BlackBerry 6 and Android, the limit depends on the memory condition of the device. Large message may result in an out of memory error.
SupWorkflowVersion	The version number of the Hybrid App package.

4. In the Context Variable dialog, change the value of the named variable and click **OK**.

Changing Hard Coded User Credentials

The administrator can change hard coded user credentials assigned at design time if the design time value needs to change for a production environment.

1. In the left navigation pane, expand the **Hybrid Apps** folder and select the Hybrid App package to configure context variables for.
2. In the right administration pane, click the **Context Variables** tab.
3. Select one or both of the variables: SupUser or SupPassword, and click **Modify**.
4. Type the new value and click **OK**.

Adding a Certificate File to the Hybrid App Package

The administrator can change the credential certificate assigned at design time.

Note: SAP recommends that you use Internet Explorer to perform this procedure.

1. In the left navigation pane, expand the **Hybrid Apps** folder and select the Hybrid App package to configure context variables for.
2. In the right administration pane, click the **Context Variables** tab.
3. Select **SupPassword** and click **Modify**.
4. Select **Use certificate-base credentials** and click **Browse** to find and upload a certificate file.
5. Set the value for **Certificate password** and click **OK**.

On the Context Variables page, the read-only values of SupUser, SupCertificateSubject, SupCertificateNotBefore, SupCertificateNotAfter, and SupCertificateIssuer change to reflect values of the new certificate and password you set.

End to End Trace and Performance

The SAP passport handling functionality allows for an end to end trace of data communication from the client to the back-end.

The `hwc.e2eTrace` JavaScript APIs enable or disable end-to-end trace and the ability to upload and view the trace file. SAP Mobile Server must be configured with SAP Solution Manager to upload and view this trace. See *Configuring SAP Mobile Server Performance Properties* in *SAP Control Center for SAP Mobile Platform*.

Note: End to end trace is supported on Android and iOS only.

The performance library provides the ability to capture performance metrics of the device while the Hybrid Web Container is running. Administrators can use this information to troubleshoot performance related issues.

These metrics are collected when the performance agent is enabled:

- totalTime [ms]
- networkTime [ms]
- totalCpuTime [ms]

- roundTrips
- totalBytes
- sentBytes
- receivedBytes
- memMax

Enabling the Performance Agent on the Device

The performance setting on the device gives administrators a mechanism to collect performance counters when running Hybrid Apps.

Note: The performance agent is not supported on Windows Mobile devices.

Note: To enable the performance setting on BlackBerry and Android, an SD card must be installed on the device.

1. Go to the Hybrid App settings screen.
2. Click the menu key and select **Advanced**.
3. Select **Performance** to start the performance agent.
4. Unselect **Performance** to create the performance log.

The performance numbers are stored in memory and saved to a file when you stop the performance library, either on the device or through the `stopInteraction` JavaScript API. View the performance logs in SAP Control Center. See *Tracing Application Connections*.

Tracing Application Connections

Send a request to the device to retrieve log files for an application connection.

1. In the left navigation pane, select the **Applications** node.
2. In the right administration pane, click **Application Connections** tab.
3. Select an application connection, and click **Get Trace**.

Note: If the client application does not support tracing, you cannot trace the application connection. To determine if the client application supports tracing, review the **Capabilities** properties for the application connection. If the **Application Supports Client Callable Components** property has a value of `False`, tracing is not supported.

The application connection status must be "online" to retrieve the logs.

4. Click **OK**.
5. When the application connection is online, you can view the log contents in SAP Control Center by retrieving the server log for the domain that the application connection belongs to. The trace logs will be identified by one of the following values in the Category column of the Server log tab: PerformanceAgent, MOCA, or SQLTrace. Trace logs can also be

viewed in the file system. The default location for single node and cluster installations is `SMP_HOME\Servers\UnwiredServer\logs\ClientTrace`.

Build a Customized Hybrid Web Container Using the Provided Source Code

Use the provided source code to build your own customized user interface and configure other resources in the development environment of your choice.

You must first *Build a Customized Hybrid Web Container Using the Provided Source Code* before creating a prepackaged Hybrid App.

Building the Android Hybrid Web Container Using the Provided Source Code

The Hybrid Web Container in this procedure is a sample container provided with the SAP Mobile Platform Mobile SDK installation.

Prerequisites

- Install the Android SDK version 2.2, API Level 8. You can get the Android SDK at <http://developer.android.com/sdk/index.html>.
- If you are developing in Eclipse, install the ADT Plug-in for Eclipse.

Task

This example uses Eclipse as the development environment, but you can use any development environment.

1. Open Eclipse and select **File > Import**.
2. Expand the **General** folder, choose **Existing Projects into Workspace**, and click **Next**.
3. Choose **Select archive file**, browse to `SMP_HOME\MobileSDK<version>\HybridApp\Containers\Android\`, and select `Android_HWC_<version>.zip`.
4. Click **Finish**.

A Hybrid Web Container project folder is added to Workspace Navigator. You may receive an error indicating that the source folder `gen` is missing. If so, add an empty folder named `gen` to the `src` folder in the project.

5. Open the `local.properties` file in the main directory of the project. This file contains a non-commented line, `sdk.dir = <filepath>`. Verify the `<filepath>` matches the filepath to your installation of the Android SDK.

Build a Customized Hybrid Web Container Using the Provided Source Code

6. If you receive an `Android requires compiler compliance level 5.0 or 6.0. Found '1.4' instead`. Please use `Android Tools > Fix Project Properties` error, follow the instructions and then clean the project.
7. If you receive errors of the type `... must override a superclass method`, make sure the Java compiler has its compliance set to 1.6.
 - a) Right-click the **HybridWebContainer** project and select **Properties**.
 - b) Go to the Java Compiler section and set the Compiler compliance level to 1.6.
 - c) Rebuild the project.

Building the Android Hybrid Web Container Outside of Eclipse

You can build the Android Hybrid Web Container independent from SAP Mobile Platform.

1. Open a command prompt and navigate to the base directory of the Hybrid Web Container project.
2. Run either the **ant debug** or **ant release** command, depending on whether you want to debug or release the Hybrid Web Container.

You can download Apache Ant from <http://ant.apache.org/bindownload.cgi>, if necessary.

A file named either `HybridWebContainer-debug.apk` or

`HybridWebContainer-release-unsigned.apk` (depending on the command you used) is added to the `bin` folder. If a file already exists with that name, it is overwritten.

3. Use Android Debug Bridge (ADB), which is included in the Android SDK installation, to install the `.apk` to the emulator.
 - a) Launch an Android Virtual Device (AVD) that does not have the Hybrid Web Container installed (or uninstall it if it is installed).
 - b) In the Command Prompt window, navigate to the folder that contains the `adb.exe` file, which should be in the `.../android-sdk/platform-tools/` folder.
 - c) Execute: **adb install <path>**, where *<path>* is the full filepath to the `HybridWebContainer.apk` file.

Building the BlackBerry Hybrid Web Container Using the Provided Source Code

You can use the provided BlackBerry Hybrid Web Container template to build a custom user interface and configure other resources.

Prerequisites

- Install the BlackBerry Plug-in for Eclipse. See <https://developer.blackberry.com/java/download/eclipse?IID=DEVJVA1223>.
- Register the device in SAP Control Center.

Task

This example uses Eclipse as the development environment. If you use another development environment, the steps might vary.

1. Extract the files from `SMP_HOME\MobileSDK<version>\HybridApp\Containers\BB\BB_HWC_<version>.zip`
2. In Eclipse, import the BlackBerry Hybrid Web Container template as a legacy BlackBerry project:
 - a) Select **File > Import**.
 - b) Expand the **BlackBerry** folder.
 - c) Select **Import Legacy BlackBerry Projects**.
 - d) Click **Next**.
 - e) Specify the JRE and, in the BlackBerry Workspace field, browse to the `HWCtemplate.jdw` file and select the project to import.
 - f) Select **Copy BlackBerry projects into workspace** to create a copy of the imported project in the Eclipse workspace.
 - g) Click **Finish**.
3. Supply a signing key.

Supplying a Signing Key

You must supply a BlackBerry code signing key from BlackBerry to access the persistent store.

1. Go to <https://www.blackberry.com/SignedKeys/codesigning.html> to obtain a signing key and import into Eclipse following BlackBerry's instructions.

Once you import your signing key, you must change some code to let the Hybrid Web Container know which keys you are using.

Build a Customized Hybrid Web Container Using the Provided Source Code

2. Open the `CustomizationHelper.java` file for editing.
3. Find the method named `getCodeSignerId()` and update it to return the name of your key.

If there is no key file and believe it is not needed, return `NULL` from `getCodeSignerId()`. The key file is used to protect data in the persistent store. If there is no key file and you want to create one, install and use *BlackBerry Signing Authority Tool*.

Once you have created a key file, add it to your project, so it is included in the `.cod` file. `getCodeSignerId()` then needs to return the name of the key file without an extension.

4. Add the key file to your project so it is included in the `.cod` file.

Building the iOS Hybrid Web Container Using the Provided Source Code

Build a sample Hybrid Web Container.

Prerequisites

- Register the device in SAP Control Center.
- Have access to a Mac with a supported version of Xcode and the iOS SDK.

See *Supported Hardware and Software* for the most current version information for mobile device platforms and third-party development environments.

Task

1. On your Mac, connect to the Microsoft Windows machine where SAP Mobile Platform is installed:
 - a) In the Apple menu, click **Go > Connect to Server**.
 - b) Enter the name or IP address of the machine.
For example, `smb://machine DNS name` or `smb://IP Address`.
2. Copy the `iOS_HWC_version.tar.gz` archive from `SMP_HOME\MobileSDKversion\HybridApp\Containers\iOS\` to a location on your Mac.
In the archive file name, *version* is the current SAP Mobile Server version number. For example, `iOS_HWC_2.3.2.tar.gz`.
3. Unpack `iOS_HWC_version.tar.gz`.
The extraction creates a `HybridWebContainer` directory.

Build a Customized Hybrid Web Container Using the Provided Source Code

4. In the `HybridWebContainer` directory, double-click **HWC.xcodeproj** to open it in the Xcode IDE.
5. If you are building for a device, you must add provisioning profiles to the project to be able to sign the executable.
 - a) In Xcode, click the **HWC** project and select the HWC target.
 - b) Select the **Build Settings** tab.
 - c) Under the Code Signing section, add code-signing identities for each configuration (Debug, Release, or Distribution) you want to build, depending on how you will deploy the app.

When you build the Hybrid Web Container using your provisioning profile, you are creating your own version of the application. You can reuse the bundle ID that is distributed with the HWC template project, but you cannot upgrade your custom-built application through the normal means.

The reason for this is because on iOS the Keychain is used to store information and Keychain rights depend on the provisioning profile used to sign your application. Therefore, you should consistently use the same provisioning profile across different versions of your application. Follow the instructions in *Using Multiple Hybrid Web Containers on the Same iOS Device* when you build the HWC template source.

6. In Xcode, click **Product > Build** to build the project.

Building the Windows Mobile Hybrid Web Container Using the Provided Source Code

Use the provided Windows Mobile Hybrid Web Container template to build your own customized user interface and configure other resources.

1. Unpack `SMP_HOME\MobileSDK<version>\HybridApp\Containers\WM\WM_HWC_<version>.zip` into a local folder.
2. Include custom code files in your template project:
 - a) In Visual Studio, open Solution Explorer and select the template project.
 - b) Click the **Show All Files** button and select all files in the CustomCode folder.
 - c) With all files selected, right-click and choose **Include In Project**.
3. Specify the signing for the template project:
 - a) Right-click the project in the Solution Explorer and choose **Properties**.
 - b) Open the Signing tab, and select an existing key file or create a new one.
4. Right-click the project and choose **Add Reference**.
5. Click **Browse**, select **HybridAppLib.dll**, and click **OK**.

Build a Customized Hybrid Web Container Using the Provided Source Code

Install and Configure the Hybrid Web Container On the Device

To enable deploying Hybrid App packages to a device, you must download, install, and configure the Hybrid Web Container on the device.

Deploy the Hybrid Web Container to devices and register the devices with SAP Mobile Server. You can use Afaria® to install the container on devices for enterprise deployment. For information on setting up an Afaria environment, see *Provisioning With Afaria* in *Mobile Application Life Cycle*.

See the configuration procedure for your device type.

Preparing Android Devices for the Hybrid Web Container

Install the Hybrid Web Container on the Android device using the Android SDK. In the Settings for your Android device, disable all keyboards except the Android keyboard.

Installing the Hybrid Web Container on Android Devices

Use the Android SDK Manager to install Hybrid Web Container application files.

To install the Android Hybrid Web Container on your Android device:

1. Connect the device.
2. Install the Android SDK.
3. Run `platform-tools\adb` and install `SMP_HOME\MobileSDK<version>\HybridApp\Containers\Android\HybridWebContainer.apk`.

For example:

```
C:\Android\android-sdk\platform-tools\adb install ^
SMP_HOME\MobileSDK<version>\HybridApp\Containers\Android
\HybridWebContainer.apk
```

Configuring the Android Emulator

Configure an Android emulator for testing a Hybrid App package.

Note: The steps or interface may be different depending on the Android SDK version you are using.

1. Install the Android SDK.
 - a) Go to <http://developer.android.com/sdk/>.
 - b) Download the Android SDK (for example, `installer_r21-windows.exe`).

Install and Configure the Hybrid Web Container On the Device

Note: Do not download the larger SDK starter package (ADT Bundle for Windows). The starter package includes not only the SDK but also the ADT plug-in for Eclipse and a more recent platform than the one shown in this tutorial.

- c) In Windows Explorer, double-click the downloaded installer to run it.

Note where the SDK is installed on your system, for example,

C:\Program Files\Android\android-sdk.

2. Install the SDK platform tools:

- a) Run the Android SDK Manager, *android-sdk\SDK Manager.exe*.

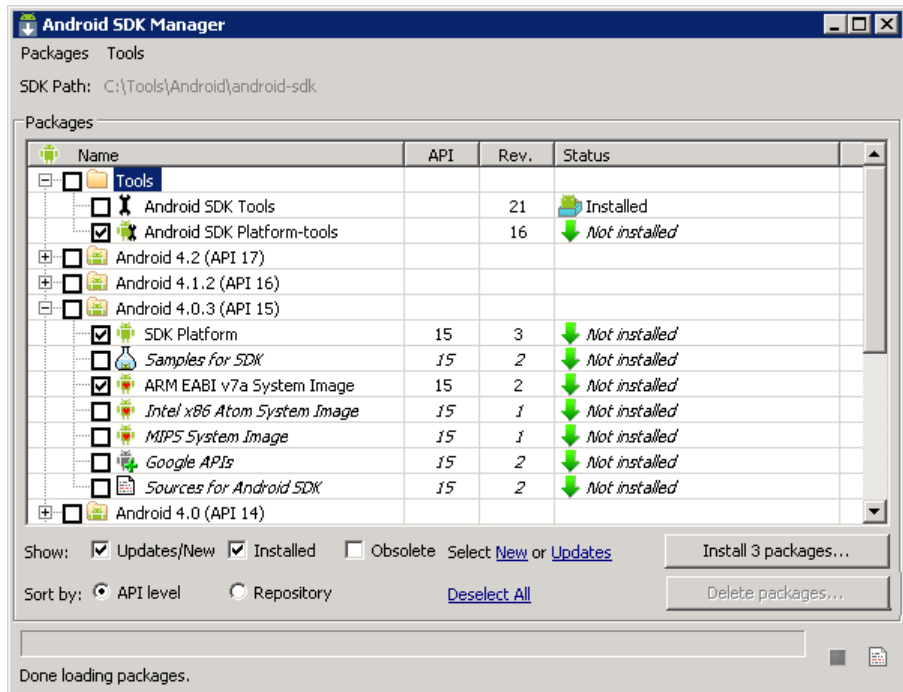
- b) In the Android SDK Manager, expand Tools and select **Android SDK Platform-tools**.

Android SDK Tools should already be installed.

- c) Expand **Android 4.0.3 (API 15)** and select these packages:

- **SDK Platform.**
- **ARM EABI v7a System Image.**

- d) Click the **Install *n* packages** button.



- e) In Choose Packages to Install, select **Accept All**, then click **Install**. Close the log window when done.

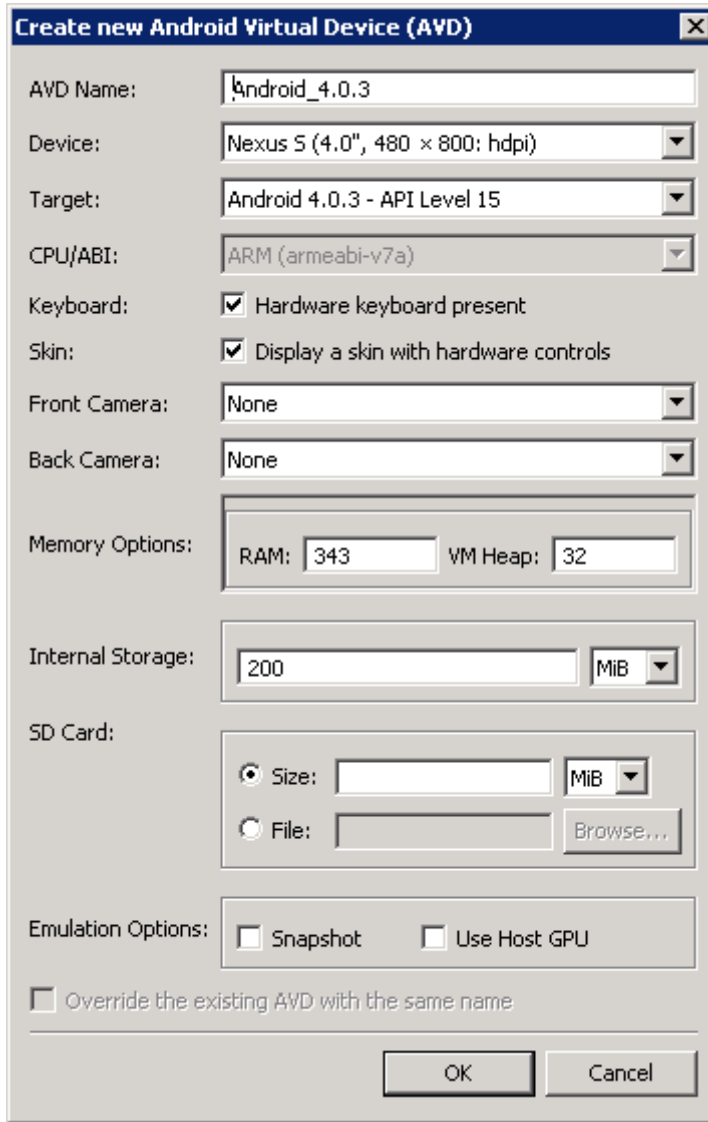
- f) Close the Android SDK Manager.

3. Run the Android Virtual Device Manager, *android-sdk\AVD Manager.exe*.

4. Configure and start an Android emulator instance.

- a) In the AVD Manager, click **New**.
- b) In the Create new Android Virtual Device window, enter an AVD name and select a supported Android device for this instance.

For example:



Create new Android Virtual Device (AVD)

AVD Name:

Device:

Target:

CPU/ABI:

Keyboard: Hardware keyboard present

Skin: Display a skin with hardware controls

Front Camera:

Back Camera:

Memory Options: RAM: VM Heap:

Internal Storage:

SD Card:

Size:

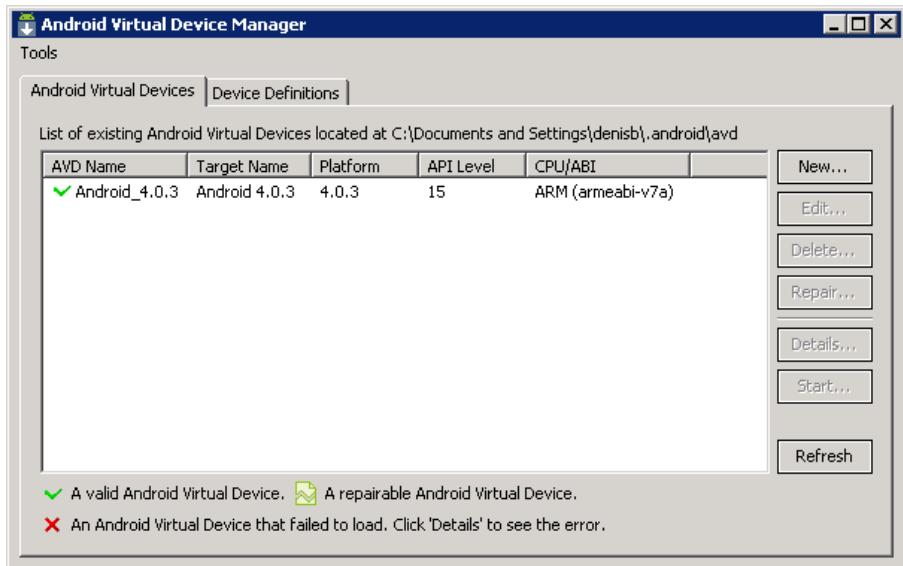
File:

Emulation Options: Snapshot Use Host GPU

Override the existing AVD with the same name

- c) Click **OK** to add the instance to the AVD Manager.

Install and Configure the Hybrid Web Container On the Device



- d) Select the new virtual device and click **Start**.
 - e) In Launch Options, click **Launch** to open the Android emulator screen.
5. Install the Hybrid Web Container on the emulator instance:
- a) With the Android emulator running, open a command prompt window.
 - b) Run `android-sdk\platform-tools install SMP_HOME \MobileSDKversion\HybridApp\Containers\Android\HybridWebContainer.apk`.

For example:

```
C:\Android\android-sdk\platform-tools>adb install ^
C:\SAP\MobilePlatform\MobileSDKversion\HybridApp\Containers
\Android\HybridWebContainer.apk
```

Preparing BlackBerry Devices for the Hybrid Web Container

Install the Hybrid Web Container on the BlackBerry device using BlackBerry Desktop Manager.

Prerequisites

For prerequisites and complete information about provisioning BlackBerry devices see *Setting Up BES Environments for SAP Mobile Platform Applications* in *Mobile Application Life Cycle*.

Task

1. Connect the BlackBerry device to the computer that contains the Hybrid Web Container for BlackBerry.
2. Run the BlackBerry Desktop Manager following the instructions in the RIM documentation.
3. In the BlackBerry Desktop Software, select **Application Loader**.
4. Under Add/Remove Applications, select **Start**.
5. Browse to the location on your local machine or network that contains the Hybrid Web Container `HybridWebContainer.cod` and `HybridWebContainer.alx` container files, `<SMP_HOME>\MobileSDK<version>\HybridApp\Containers\BB`.
6. Select the files and click **Open**.
The application is listed on the Application Loader wizard.
7. Click **Next**.
8. Click **Finish**.
9. Restart your BlackBerry device.

Installing the Hybrid Web Container on BlackBerry Devices Over the Air

Your system administrator must provide the appropriate information for installing the Hybrid Web Container over the air, and the BlackBerry Exchange Server (BES) must be available.

Note: For information about provisioning BlackBerry devices see *Setting Up BES Environments for SAP Mobile Platform Applications* in *Mobile Application Life Cycle*.

The administrator stages the OTA files in a Web-accessible location and notifies BlackBerry device users through an e-mail message with a link, or A URL to the Hybrid Web Container installation file. This can be accomplished by pointing the BlackBerry browser to the `HybridWebContainer.jad` file. This single JAD and associated files for this type of deployment are located in `<SMP_HOME>\MobileSDK<version>\HybridApp\Containers\BB\OTA`.

Enabling Hybrid Web Container Message Notification

On each BlackBerry device, customize the alert profile to notify users when a new Hybrid Web Container message is received.

By default, Hybrid Web Container messages do not trigger BlackBerry sounds and alerts. The only indication of a new message is a change to the home screen icon. To add notifications, each BlackBerry user can create a new alert profile.

This topic describes how to configure alert profiles for Hybrid Web Container messages on a BlackBerry 9800 running BlackBerry 6 or a BlackBerry 9900 running BlackBerry 7.1: The

Install and Configure the Hybrid Web Container On the Device

steps are similar for other supported BlackBerry devices. For information about all devices, see the BlackBerry Manuals page at <http://docs.blackberry.com>.

1. On the home screen, select the **Sound and Alert** application.
2. Select **Change Sounds and Alerts**.
3. Select **Profile Management**.
4. Select **Add Custom Profile**.
5. In New Custom Profile, enter a name for the new profile in **Name**.
6. Expand **Other Applications - Notifiers** and choose **Hybrid Web Container**.
7. Configure the sound, visual, and other alert options that you want.
8. Save your changes and close the profile.
For example, open the menu and choose **Close**. When prompted, choose the **Save** option.
9. Activate the customized profile: return to the home screen, select the **Sound and Alert** application again, and choose the new profile.

Configuring the BlackBerry Simulator for Hybrid Web Containers

Copy the `HybridWebContainer.cod` file to the BlackBerry Simulator directory.

Prerequisites

MDS must be running.

Task

1. Start the BlackBerry simulator.
2. From **File > Load BlackBerry Application or Theme**.
3. Navigate to `SMP_HOME\MobileSDK<version>\HybridApp\Containers\BB`.
4. Select the `HybridWebContainer.cod` file, then click **OK**.

Preparing iOS Devices for the Hybrid Web Container

Install the Hybrid Web Container on the device using the App Store, or use the source code provided for the Hybrid Web Container to deploy to the iOS simulator from the Xcode project.

Complete these prerequisites before provisioning the Hybrid Web Container:

- Determine your security policy – SAP Mobile Platform provides a single administration console, SAP Control Center, which allows you to centrally manage, secure, and deploy applications and devices. Device user involvement is not required and you can maintain the authorization methods you already have in place. See *Security > Device Security*.

- Register each application connection using SAP Control Center – application connections pair an application with a device. See *SAP Control Center for SAP Mobile Platform* documentation.

Installing the Hybrid Web Container on the iOS Device

How you install the Hybrid Web Container on your iOS device depends on how your company provisions the application.

Your company will choose a method for provisioning the application. Your system administrator determines how you obtain and install the Hybrid Web Container. The possible methods include:

- Downloading and installing the free version of the Hybrid Web Container from the Apple App Store. The free version should not be used for enterprise deployment.
- Obtaining a copy of the application on your corporate network or through a link in an e-mail message, then using iTunes to install and synchronize it to your device. This mechanism should be used for enterprise deployment and is based on the application built using the XCode project, which is included as part of SAP Mobile Platform installation.

Installing the Hybrid Web Container from the Apple App Store

Install the Hybrid Web Container from the Apple App Store.

This is a free version of the Hybrid Web Container and should not be used for enterprise deployment.

1. On your device, on the iOS home page, tap **App Store**.
2. Search for **SAP Hybrid Web Container**.
3. In the search results, find the version of the Hybrid Web Container to install and click **Free**.
4. Tap **Install** to download the application.
5. In **Settings > HWC<version>**, for Connection Info, enter:
 - Server Name – the machine that hosts the server where the mobile application project is deployed.
 - Server Port – SAP Mobile Server port number. The default is 5001.
 - Farm ID – the farm ID you entered when you registered the application connection in SAP Control Center.
 - Protocol – HTTP or HTTPS. The protocol with which to connect to the Relay Server or the reverse proxy server. The default is HTTP.
 - (Optional) URL Suffix – the URL suffix used to connect to a Relay Server or the reverse proxy server. Get this information from your administrator. See *Device Advanced Properties* in *System Administration*.
6. Scroll to the page that contains the **HWC** icon, then tap to launch.
7. Enter your personal identification number (PIN).

This PIN is a security measure to safeguard your company's data.

Install and Configure the Hybrid Web Container On the Device

- The PIN must be at least six digits and cannot be consecutive digits (for example, 123456), or same digit (for example, 111111).
- (First time/reinstallation) Create a PIN in the **Password** field, then verify it in the second field.
- (Second or subsequent logins) Enter the PIN in the **Password** field. Select **Change Password** to change the PIN. You can change the PIN once you enter the current PIN. The **HWC** page appears.

8. Tap **Messages** to view messages/notifications.
9. (Optional) If instructed by your system administrator, enable notifications on your device.

Installing the Hybrid Web Container Using iTunes

Install the Hybrid Web Container using iTunes.

1. Launch iTunes.
2. Download the application from your corporate network to your Applications library.
3. Sync the application to your Apple mobile device.
4. Specify the connection settings in **Settings > Hybrid App**.

Preparing Windows Mobile Devices for the Hybrid Web Container

Install the Hybrid Web Container on the Windows Mobile device.

Installing the Hybrid Web Container on Windows Mobile Devices

Install and configure the Hybrid Web Container required to prepare a Windows Mobile device to run Hybrid Apps.

1. Navigate to `SMP_HOME\MobileSDK\HybridApp\Containers\WM`.
2. Copy the Windows Mobile Professional device file, `HybridWebContainer.cab`, to the device's **My Documents** folder.
3. Cradle the Windows Mobile device.
4. Connect a USB cable between the PC and device, and transfer the `.cab` file.
5. Open the `HybridWebContainer.cab` file from the Windows Mobile device. This installs the container.
6. In Programs, click the Hybrid Web Container icon and click **Settings**.
7. In the Connection screen, enter the connection settings. These settings should match the values you used when you registered the device in SAP Control Center.

Note: Click **Menu** and select **Show Log** to view the container log. This is useful for checking the connection, or retrieving other debugging information.

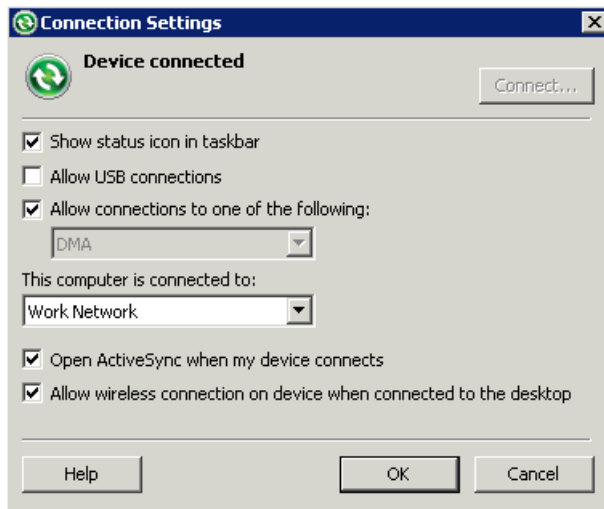
Installing Microsoft Synchronization Software

Install and configure Microsoft synchronization software so you can deploy and run an application on a Windows Mobile emulator.

Note: These instructions describe how to install Microsoft ActiveSync for Windows XP. If you are using Windows Vista, Windows 7, or Windows 2008, install Virtual PC 2007 SP1 and Windows Mobile Device Center to manage synchronization settings. Download the Windows Mobile Device Center from <http://www.microsoft.com/en-us/download/details.aspx?id=15> and follow Microsoft instructions for installing and using that software instead of this procedure.

1. Download Microsoft ActiveSync:
 - a) In a Web browser, open the Windows Phone page at <http://www.microsoft.com/windowsmobile/en-us/help/synchronize/device-synch.mspx>.
 - b) Follow the instructions to select and download the sync software for the system's operating system. Windows XP requires ActiveSync version 4.5.
 - c) In the Windows Phone downloads page, click the **ActiveSync** button.
 - d) Download the ActiveSync installation file and save it to your local system.
2. Run the downloaded installation file.
For example, double-click **setup.msi** in Windows Explorer.
3. When the installation is complete, restart the system.
4. Start ActiveSync if it does not start automatically.
For example, click **Start > Programs > Microsoft ActiveSync**.
5. Click **File > Connection Settings**.
6. Select **Allow connections to one of the following**, then select **DMA**.
7. Select **Work Network** for "This computer is connected to".

Install and Configure the Hybrid Web Container On the Device



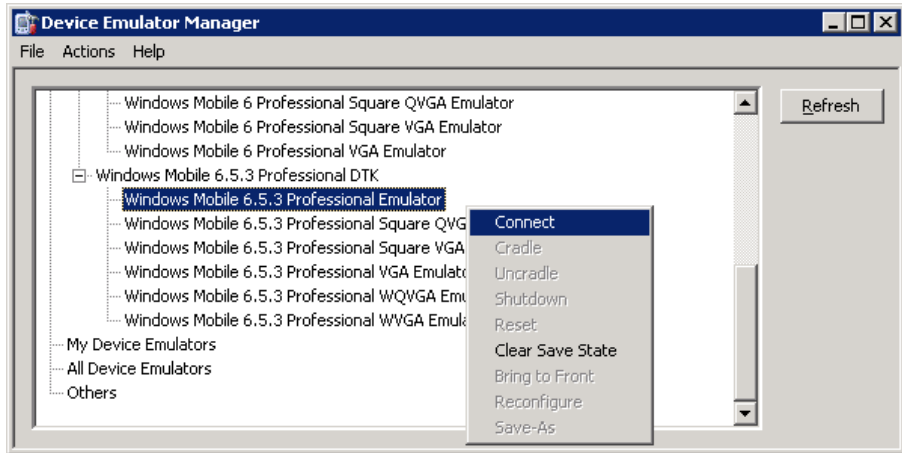
8. Click **OK**.

Installing the Hybrid Web Container on the Windows Mobile Emulator

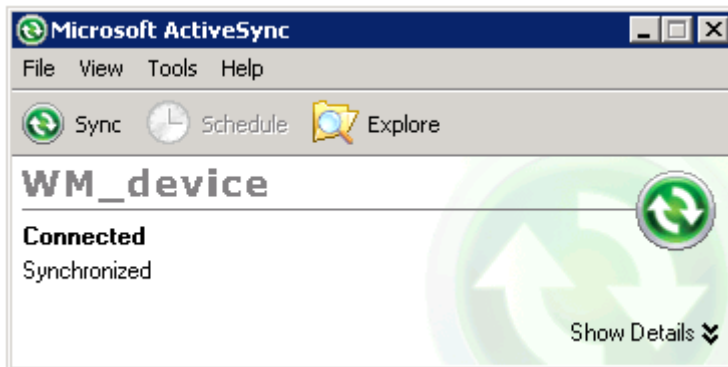
Install the Hybrid Web Container software on your emulator.

1. Start the synchronization software.
For example, on Windows XP, start Microsoft ActiveSync. On Windows Vista, Windows 7, or Windows 2008, start the Windows Mobile Device Center.
2. Start the Device Emulator Manager and select an emulator to run.
For example:
 - a. Double-click `C:\Program Files\Microsoft Device Emulator\1.0\dvcemumanager.exe`.
 - b. In the Device Emulator Manager, right-click the device you want to run and choose **Connect** to open the emulator.

Install and Configure the Hybrid Web Container On the Device



- c. In the Device Emulator Manager, right-click the device again and click **Cradle**.
3. The synchronization software runs and connects to your device. If the Synchronization Setup wizard opens, follow the instructions and click **Finish**.

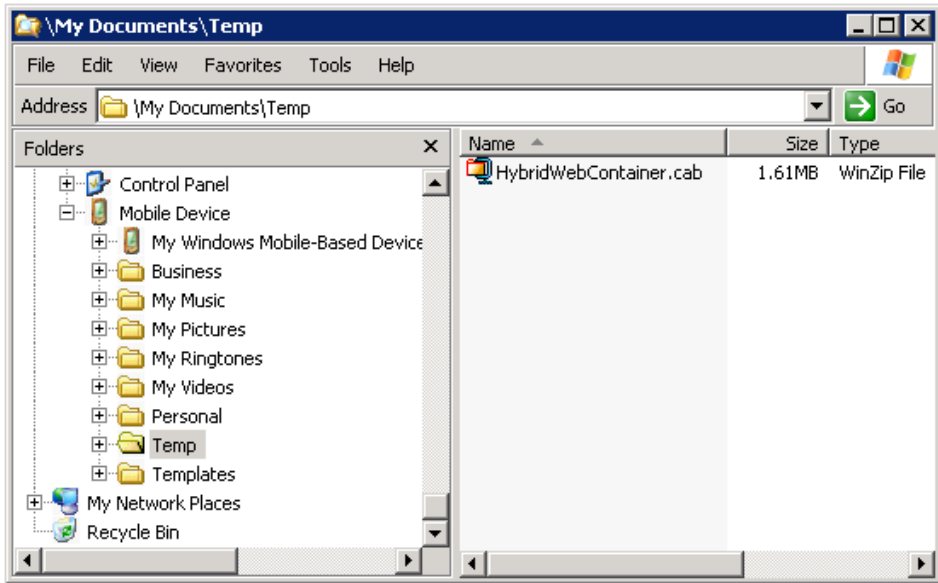


4. Run the downloaded Microsoft .NET Compact Framework Redistributable file to install the .NET Compact Framework on your running emulator. Follow the setup wizard instructions, and click **Finish** to close the wizard when you are done.

Note: Be sure to run the installer while your emulator is running; otherwise the .NET Compact Framework Redistributable is not installed correctly.

5. Go to `SMP_HOME\MobileSDK<version>\HybridApp\Containers\WM` and copy the `HybridWebContainer.cab` file to a folder on mobile device folder on your system.
For example:

Install and Configure the Hybrid Web Container On the Device



6. On the device emulator, open File Explorer and browse to the folder to which you copied the CAB file. Click the file once to install the Hybrid Web Container on your emulator.

Configure Connection Settings on the Device

Configure the connection settings for the Hybrid Web Container on the device.

See the topic for your platform.

Configuring Android Connection Settings

Configure the connection settings for the Hybrid Web Container.

1. Click the **HWC** icon on the applications screen, then select **Settings**.
2. In the basic authentication screen, enter the user name and password if you are prompted.
3. Click **Registration** to choose from the registration options:
 - Manual – enter connection settings and register manually.
 - Automatic (Password) – enter the password for automatic registration.
 - Automatic (Afaría Certificate) – register using an Afaria certificate.
 - Automatic (Local Certificate) – register using a local certificate.
4. Enter the settings for the Hybrid Web Container:
 - Server Name – the machine that hosts the server where the mobile application project is deployed.

Install and Configure the Hybrid Web Container On the Device

- Server Port – SAP Mobile Server port number. The default is 5001.
- Farm ID – the farm ID you entered when you registered the application connection in SAP Control Center.
- Protocol – HTTP or HTTPS. The protocol with which to connect to the Relay Server or the reverse proxy server. The default is HTTP.
- (Optional) URL Suffix – the URL suffix used to connect to a Relay Server or the reverse proxy server. Get this information from your administrator. See *Device Advanced Properties* in *System Administration*.

Select **Save** to save the settings.

5. (Optional) Configure trace and performance settings:

Note: To enable the performance agent, an SD card must be installed.

- a) In the Settings screen, click the menu key and select **Advanced**.
 - b) Select **Trace** to enable SAP Passport end to end trace.
 - c) Click **Level** to choose the log level.
 - Low – focuses on response-time-distribution analysis, in other words, how much time is spent on each server component, or the specific location of a bottleneck.
 - Medium – (default) gives performance analysis. Performance traces are triggered on the server-side.
 - High – gives functional analysis and has detailed functional logging and tracing.
 - d) Select **Performance** to enable the performance agent.
6. Start the application, then view the settings log to verify that the connection is active. From the application, tap **Settings > Show Log**.

Configuring BlackBerry Connection Settings

Configure the connection settings for the Hybrid Web Container.

1. Click the **Hybrid Web Container** icon on the applications screen, then press the **Menu** key and select **Settings**.
2. Enter the settings for the Hybrid Web Container:
 - Registration – choose from:
 - Manual – enter connection settings and register manually.
 - Auto (Password) – when you select this option, the Password field is enabled. Enter your password.

Note: The Activation Code and Enable Automatic Registration options are mutually exclusive. If you use a password for automatic registration, you cannot enter an activation code, and vice versa.

- Auto (Afaría Cert) – register using an Afaría certificate. When you choose this option, these fields are enabled:
 - Common name

Install and Configure the Hybrid Web Container On the Device

- Challenge code
- Auto (Local Cert) – register using a local certificate.
- Server Name – the machine that hosts the server where the mobile application project is deployed.
- Server Port – SAP Mobile Server port number. The default is 5001.
- Farm ID – the farm ID you entered when you registered the device in SAP Control Center.
- User Name – the user you registered in SAP Control Center.

Note: When there are multiple application connection templates for the same APP ID, and you need to establish a connection using the anonymous security configuration, you must include the security configuration in the user name, in this format:
`anonymous@anonymous.`

- Activation Code – the activation code for the user, for example, 123.
 - Protocol – the protocol with which to connect to the Relay Server or the reverse proxy server. Choose from:
 - HTTP
 - HTTPS
 - (Optional) URL Suffix – the URL suffix used to connect to a Relay Server or the reverse proxy server. Get this information from your administrator. See *Device Advanced Properties in System Administration*.
3. Select **Menu > Save** to save the settings.
 4. (Optional) In the settings screen, click the menu key and select **Advanced** to turn on the performance agent.

Note: To enable the performance agent, an SD card must be installed.

5. Start the application, then view the settings log to verify that the connection is active.
In the Hybrid Web Container, select **Settings**. On the connection settings screen, select **Show Log**.

Configuring iOS Connection Settings

Configure the settings for the Hybrid Web Container.

1. Go to the device Settings screen and click **HWC**.
2. In the basic authentication screen, enter the user name and password if you are prompted.
3. Enter the settings for the Hybrid Web Container:
 - Server Name – the machine that hosts the server where the mobile application project is deployed.
 - Server Port – SAP Mobile Server port number. The default is 5001.

Install and Configure the Hybrid Web Container On the Device

- Farm ID – the farm ID you entered when you registered the application connection in SAP Control Center.
 - Protocol – HTTP or HTTPS. The protocol with which to connect to the Relay Server or the reverse proxy server. The default is HTTP.
 - (Optional) URL Suffix – the URL suffix used to connect to a Relay Server or the reverse proxy server. Get this information from your administrator. See *Device Advanced Properties* in *System Administration*.
4. Click in the **Registration Method** field to choose a registration method:
 - Manual – enter connection settings and register manually.
 - Automatic (Password) – when you select this option, the Password field is enabled.
 - Automatic (Afaria Certificate) – allows you to register using an Afaria certificate.
 5. Click the **HWC** icon to go back to the settings screen.
 6. If you chose manual registration, enter your user name and activation code.

Note: When there are multiple application connection templates for the same APP ID, and you need to establish a connection using the anonymous security configuration, you must include the security configuration in the user name, in this format:

anonymous@anonymous.

The activation code and password for automatic registration are mutually exclusive. If you use a password for automatic registration, you cannot enter an activation code, and vice versa.

7. If you chose automatic registration, enter your user name and password.
8. If you chose automatic registration with an Afaria certificate, enter the common name and challenge code for the Afaria certificate.

Configuring Windows Mobile Connection Settings

Configure the connection settings.

Prerequisites

Install the Hybrid Web Container CAB file.

Task

1. Select **Start > Programs**.
2. Click the Hybrid Web Container icon.
3. Click **Settings**.
4. In the Connection screen, enter the connection settings:
 - Server Name – the machine that hosts the server where the mobile application project is deployed.
 - Server Port – SAP Mobile Server port number. The default is 5001.

Install and Configure the Hybrid Web Container On the Device

- Farm ID – the farm ID you entered when you registered the device in SAP Control Center.
- User Name – the user you registered in SAP Control Center.

Note: When there are multiple application connection templates for the same APP ID, and you need to establish a connection using the anonymous security configuration, you must include the security configuration in the user name, in this format:
`anonymous@anonymous.`

- Registration – choose from:
 - Manual – enter connection settings and register manually.
 - Automatic – when you select this option, the Password field is enabled.

Note: The Activation Code and Enable Automatic Registration options are mutually exclusive. If you use a password for automatic registration, you cannot enter an activation code, and vice versa.

- Certificate – allows you to register using a certificate.
 - Activation Code – the activation code for the user, for example, 123.
 - Password – this field is enabled if you chose Automatic registration. Enter your password.
 - Certificate – this field is enabled if you chose Certificate as the registration type. Choose your certificate. The User Name field is populated with the certificate name.
 - Protocol – the protocol with which to connect to the Relay Server or the reverse proxy server. Choose from:
 - HTTP
 - HTTPS
5. Click **Advanced** for these options:
- Allow roaming – the device is allowed to connect to server while roaming. By default, this is set to true.
 - (Optional) URL Suffix – used to connect to a Relay Server or the reverse proxy server. Get this information from your administrator. See *System Administration > System Reference > Application Connection Properties > Device Advanced Properties*.
 - Keep alive – the frequency used to maintain the wireless connection, in seconds. Acceptable values: 30 to 1800. The default is 240.
6. Click **Save**.
7. Start the Hybrid App, then view the settings log to verify that the connection is active. In the Settings screen, click **Menu > Show Log**.

Install and Test Certificates on Simulators and Devices

Install and test certificates on various types of simulators and devices.

Note: The supported algorithm for the public-key cryptography used in the X.509 certificates is RSA.

Copy the generated .p12 certificate to the device on which you are installing.

See the User Guide for your device or simulator for instructions.

Installing X.509 Certificates on Windows Mobile Devices and Emulators

Install the *.p12 certificate on a Windows Mobile device or simulator and select it during authentication.

1. Launch the simulator or device.
2. Start the Windows synchronization software and cradle the device.
3. Use File Explorer to copy the *.p12 certificate to the simulator or device.
4. Navigate to and double-click the certificate.
5. Enter the password at the prompt and click **Done**.

An informational window indicates the certificate installed successfully.

Testing X.509 Certificates on Windows Mobile Devices and Emulators

Select an X.509 certificate to use for user authentication.

Prerequisites

1. Create a Hybrid App that prompts the user to specify a certificate as credentials.
2. Package and assign the Hybrid App to a Windows Mobile device user.

Task

1. In the Programs screen, open the Hybrid Web Container and select the Hybrid App to test.
2. Select the **Specify Certificate Credentials** menu item from the Certificate Picker.
3. Select the certificate and continue with the Hybrid App.

Installing X.509 Certificates on Android Devices and Emulators

Install the *.p12 certificate on an Android device or emulator.

Prerequisites

- Java SE Development Kit (JDK) must be installed.
- The Android SDK must be installed.

Task

1. Connect the Android device to your computer with the USB cable.
2. To install using Eclipse with the ADT plugin:

Note: USB debugging must be enabled.

- a) Open the Windows File Explorer view. From the menu bar, navigate to **Window > Show View > Other**.
 - b) In the Show View dialog, expand the Android folder and select **File Explorer**.
 - c) Expand **mnt > sdcard** and select the **sdcard** folder.
 - d) In the top right of the File Explorer view, click **Push a file onto the device**.
 - e) In the Put File on Device dialog, select the certificate and click **Open**.
3. To install using Windows Explorer:

Note: USB debugging must be disabled.

- a) Open **Windows Explorer**
 - b) Under your computer, click the Android device to expand the folder.
 - c) Click **Device Storage**, navigate to and select the certificate.
 - d) Import the certificate to the Device Storage folder.
4. To install using the Android Debug Bridge (adb):

Note: USB debugging must be enabled.

- a) Open the command line directory to the `adb.exe` file, for example, `C:\Program Files\android-sdk-windows\tools`, or `C:\Program Files\android-sdk-windows\platform-tools`
- b) Run the command: `adb push %PathToCert%\MyCert.p12 /sdcard/MyCert.p12`

Testing X.509 Certificates on Android Devices and Emulators

Select an X.509 certificate for user authentication.

Prerequisites

1. Create a Hybrid App that prompts the user to specify a certificate as credentials.
2. Package and assign the Hybrid App to an Android device user.

Task

1. On the Android device or emulator, in applications, click **Hybrid Web Container**.
2. Select the Hybrid App on which to test the installed certificate.
3. From the Certificate Picker, select the **Specify Certificate Credentials** menu item.
4. Select the certificate and click **OK**.
5. Enter the password and click **OK**.

Installing X.509 Certificates on BlackBerry Simulators and Devices

Install the .p12 certificate on the BlackBerry device or simulator and select it during authentication.

1. Install the certificate on a device:
 - a) Connect to the device with a USB cable.
 - b) Browse to the SD Card folder on the computer to which the device is connected.
 - c) Navigate to and select the certificate. Enter the password.
 - d) Import the certificate.

You can also use the BlackBerry Desktop Manager to install the certificate on the device, but you may need to perform a custom installation to access the Synchronize Certificates option.
2. Install the certificate on a simulator:
 - a) From the simulator, select **Simulate > Change SD Card**.
 - b) Add/or select the directory that contains the certificate.
 - c) Open the media application on the device, and select **Menu > Application > Files > MyFile > MediaCard**.
 - d) Navigate to and select the certificate. Enter the password.
 - e) Check the certificate and select **Menu > Import Certificate**. Click **Import Certificate** then enter the data vault password.

Testing X.509 Certificates on BlackBerry Devices and Simulators

Select an X.509 certificate to use for user authentication.

Prerequisites

1. Create a Hybrid App that prompts the user to specify a certificate as credentials.
2. Package and assign the Hybrid App to a BlackBerry device user.

Task

1. From the applications screen, open the Hybrid Web Container.
2. Select the Hybrid App for which to test the certificate.
3. From the Certificate Picker, select the **Specify Certificate Credentials** menu item.
4. Select the certificate and continue with the Hybrid App.

Installing X.509 Certificates on iOS Devices

Use Afaria to get an X.509 certificate on to an iOS device.

See the topic *Provisioning with Afaria in Mobile Application Life Cycle* for details.

Apple Push Notification Service

SAP Mobile Platform provides support for Apple Push Notification Service by pushing notifications to Hybrid Apps when the Hybrid App is offline.

With APNS, each device establishes encrypted IP connections to the service and receives notifications about availability of new items awaiting retrieval on SAP Mobile Server. This feature overcomes network issues with always-on connectivity and battery life consumption on 3G networks.

For more information on end-to-end iPhone application development and provisioning, see *Mobile Application Life Cycle*.

Note: APNS cannot be used on a simulator.

Examples of cases when notifications are sent include:

- The server identifies that a new message needs to be sent to the device. This could include:
 - A new Hybrid App is assigned to the device.
 - A DCN message is sent to SAP Mobile Server, targeting a particular user and the Hybrid App is not running.

If you want to use APNs for the Hybrid App, use the Apple Provisioning Portal to create your own .p12 certificate if you build your own Hybrid App using the source code included in `<SMP_HOME>\MobileSDK<version>\HybridApp\Containers\iOS`.

After creating the .p12 certificate, you must configure the APNs settings in SAP Control Center.

Provisioning iOS Devices

Use this procedure to provision your iOS device for APNs if you build your own application using the source code provided in `<SMP_HOME>\MobileSDK<version>\HybridApp\Containers\iOS\iOS_HWC_<version>.tar.gz`.

See the Apple developer documentation for Provisioning and Development. These procedures are documented in detail there. Applications developed for distribution must be digitally signed with a certificate issued by Apple. You must also provide a distribution provisioning profile that allows user devices to execute the application.

1. Register with Apple to download and use the iOS SDK. A free account allows you to download the SDK and develop with the simulator. To deploy Hybrid Apps to devices, you must create a certificate in your developer account and provision your device. See *Apple Local and Push Notifications in Depth* for details.
2. Use the iOS Provisioning Portal at <http://developer.apple.com/devcenter/ios/index.action> (you must log on or register as an Apple developer) to create the SSL certificate and Keys. Configure the certificate to enable for Apple Push Notification service.
3. On your Mac, launch the Keychain Access program. This is located in the `Utilities` folder.
 - a) In Keychain Access, select **Keychain Access > Certificate Assistant > Request a Certificate from Certificate Authority**.
 - b) In the Certificate Information window, enter the information. Use a unique common name.

Note: Make sure you use a different common name than a development certificate you already have. This creates a private key with the name you enter here.

A certificate request is created and saved in the Desktop folder by default.

4. In the Apple Provisioning Portal, continue with the App ID provisioning and browse to the certificate request file created in Keychain Access in the previous step, then click **Generate**.
5. Click **Continue**.
6. Click **Download Now**.

The certificate is downloaded onto your machine, the Keychain utility appears, and the certificate is imported into the "login" keychain.
7. Verify that the certificate is associated with a private key.
8. Create and install a Provisioning profile for the application.
9. In Xcode, open the `HWC.xcodeproj` project.

Note: Note the product name. This is used to configure the Hybrid Web Container in SAP Control Center and corresponds to the Application Name property in SAP Control Center.

Install and Configure the Hybrid Web Container On the Device

By default, the application name is HWC. This needs to be configured in the properties for the target. There is a 15-character limit for the product name.

10. Change `AppName` and `AppId` in the `Branding.strings` file for the necessary language resources.

This file is available under the **Resources** folder of the HWC Xcode project.

Note: The Bundle Identifier must correspond to the Bundle identifier specified in the App ID. Change it to something unique.

11. Copy the exported `<certificate_name>.p12` certificate to the machine where SAP Control Center is installed and follow the instructions in *Configuring Apple Push Settings for the Hybrid Web Container* and use the certificate you just created.

Note: Make sure you select only the certificate in the Keychain tool before exporting.

Configuring Apple Push Settings for the Hybrid Web Container

The certificate that was exported from the keychain corresponding to Apple Push settings must be configured with the correct application name in SAP Control Center.

Note: When configuring the Apple Push Notification Service, change the push gateway, push gateway port, feedback gateway, and feedback gateway port values only when configuring notifications in a development environment. To enable Apple push notifications, the firewall must allow outbound connections to Apple push notification servers on default ports 2195 and 2196. The default URL is for production and should be changed to `gateway.sandbox.push.apple.com` for development. After making these changes, you must restart your machine.

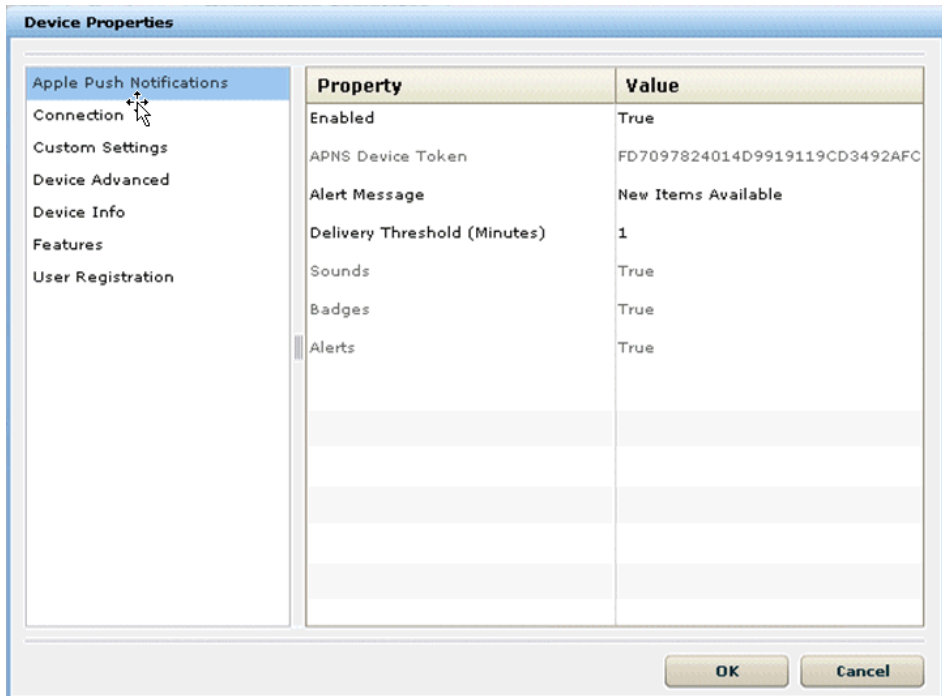
1. In the left navigation pane, select **Applications**.
2. In the right pane, select the **Applications** tab.
3. Select the **Application ID** for which you are configuring native notification and select **Properties**.
4. Select the **Push Configurations** tab and click **Add**.
5. Enter the **Application name**. Make sure this name matches the `AppId` entered in the `Branding.strings` file.

Enter:

Property	Description
Server	The push notification server.
Port	Push notification server port.
Feedback server	If a feedback service is enabled, the server to which APNS routes feedback information.
Feedback port	The feedback service port.

Property	Description
Certificate (encoded)	The security certificate used for authentication.
Certificate password	The security certificate password.

6. Click **Browse** to use a security certificate file that already exists on the server.
 - a) Select the desired certificate from the list.
 - b) Enter and confirm the certificate password.
7. Click **OK**.
8. You can verify that the device is configured for APNS correctly by verifying that the device token has been passed from the application after the application runs once on the device.



Use the **Send a Notification** tool inside the Hybrid App Designer to send a test notification.

Apple Push Notification Properties

Apple push notification properties allow iOS users to install client software on their devices.

- **APNS Device Token** – the Apple push notification service token. An application must register with Apple push notification service for the iOS to receive remote notifications

Install and Configure the Hybrid Web Container On the Device

sent by the application's provider. After the device is registered for push properly, this should contain a valid device token. See the iOS developer documentation.

- **Alert Message** – the message that appears on the client device when alerts are enabled. Default: `New items available`.
- **Delivery Threshold** – the frequency, in minutes, with which groupware notifications are sent to the device. Valid values: 0 – 65535. Default: 1.
- **Sounds** – indicates if a sound is made when a notification is received. The sound files must reside in the main bundle of the client application. Because custom alert sounds are played by the iOS system-sound facility, they must be in one of the supported audio data formats. See the iOS developer documentation.

Acceptable values: true and false.

Default: true

- **Badges** – the badge of the application icon.

Acceptable values: true and false

Default: true

- **Alerts** – the iOS standard alert. Acceptable values: true and false. Default: true.
- **Enabled** – indicates if push notification using APNs is enabled or not.

Acceptable values: true and false.

Default: true

Uninstall the Hybrid Web Container from the Device

Remove the Hybrid Web Container from the device.

Removing the Hybrid Web Container From the BlackBerry Device

Remove the Hybrid Web Container from the BlackBerry device.

You can remove the Hybrid Web Container using either the delete function on the device, or by using RIM Desktop Manager.

1. To remove the Hybrid Web Container using the delete function on the device;
 - a) On your BlackBerry device, navigate to **Options > Advanced Options > Applications**.
 - b) Scroll through the list of applications, highlight the Hybrid Web Container you want to remove and choose **Delete**.
 - c) When the confirmation dialog asks if you are sure, choose **Delete**. It may ask you to reset your device after removing the program

Install and Configure the Hybrid Web Container On the Device

When you delete the Hybrid Web Container from the device using this method, the data is removed by the `CodeModuleListener` method.

2. Use the RIM Desktop Manager to remove the Hybrid Web Container from the BlackBerry device.

See your BlackBerry documentation for how to remove applications using RIM Desktop Manager.

Note: If you delete the Hybrid Web Container using Desktop Manager or JavaLoader, the data is not deleted, as the `CodeModuleListener` is not used.

Install and Configure the Hybrid Web Container On the Device

Hybrid Web Container Customization

The Hybrid Web Container project is accompanied by libraries and the source code necessary for you to build the Hybrid Web Container.

You can customize the Hybrid Web Container in a variety of ways. Whenever a customization requires a source code modification, there is a reference to “touch points” in the code. These references are annotated with

`<PLATFORM>_CUSTOMIZATION_POINT_<descriptor>` and a descriptor identifying the customization to which they belong.

For example, all code areas associated with changing the About screen are annotated with `<PLATFORM>_CUSTOMIZATION_POINT_BRAND`. The touch points are typically accompanied by brief comments in the code explaining the necessary changes. Only source code files contain these touch points. Many of the customizations are done in the `CustomizationHelper` file.

Note: After performing any customizations, you must rebuild the container. You can customize the Hybrid Web Container in a variety of ways. SAP recommends that you always test your changes before using the resulting application.

Adding a Custom Icon for the Hybrid App Package Using the Packaging Tool

Use the packaging tool to add a custom icon to the Hybrid App package.

1. Navigate to `SMP_HOME\MobileSDK23\HybridApp\PackagingTool` and double-click the `packagingtool.bat` file if you are using a 32-bit JDK, or `packagingtool64.bat` if you are using a 64-bit JDK.
2. Select the output directory for the Hybrid App package and click **OK**.
3. In Project Explorer, choose the project to which to add the custom icon.
4. Click the **Custom Icons** tab.
5. Click **Add** to add a custom icon.

When you add a custom icon, the `manifest.xml` file is updated when you generate the package.

6. Click **Save**.
7. Click **Generate** to generate the Hybrid App package.

Manually Adding a Custom Icon to the Manifest.xml File

The simplest way to add a custom icon for the Hybrid App package is by using the packaging tool, but you can also manually update the `manifest.xml` file to include a custom icon.

1. Open `manifest.xml` for editing.
2. Specify the custom icon image files in the `<Icons></Icons>` section of the file, for example:

The `<Icons>` element should be added under the root `<Manifest>` node.

```
<Icons>
  <Icon width="32" height="32" type="png" name="ambulance" />
  <Icon width="64" height="64" type="png" name="ambulance" />
  <Icon width="32" height="32" type="png" name="car" path="html/
car.png" processedpath="html/carp.png"/>
  <Icon width="32" height="32" type="png" name="train"
path="html/train.png" />
  <Icon width="48" height="48" type="gif" name="van" path="html/
image/van.gif" processedpath="html/image/vanp.gif"/>
</Icons>
```

The unique key of the icon element in the Icons collection is the combination of width, height, type, and name.

- `width` – (required) indicates the width of the image.
- `height` – (required) indicates the height of the image.
- `type` – (required) indicates the image type. The value should be same as image file suffix.
- `name` – (required) indicates the name of the icon. You can set it as an empty string.
- `path` – (optional) indicates the path of the normal icon image saved in the package. If the path attribute is missing or empty, the image for the normal icon is saved in the `html\icon` folder. The image file name is a combination of name, width, height and type. For example, the above ambulance icon file path is `html/icon/ambulance32x32.png`.
- `processedpath` – (optional) indicates the path of the processed icon image saved in the package. If the `processedpath` attribute is missing or empty, the image for the processed icon is saved in the `html\icon` folder. The image file name is a combination of name, width, height and type with the letter `p` appended. For example, the above ambulance processed icon file path is `html/icon/ambulance32x32p.png`.

Certain image formats, such as `.ico` files, might contain multiple resolutions in a single image file. Make sure that the `manifest.xml` file includes multiple entries for each of the different resolutions that all point to the same file through the `path` and `processedpath` attributes, as shown below:

```

<Icons>
<Icon width="32" height="32" type="ico" name="car" path="html/
car.ico" processedpath="html/carp.ico">
<Icon width="64" height="64" type="ico" name="car" path="html/
car.ico" processedpath="html/carp.ico">
<Icon width="128" height="128" type="ico" name="car" path="html/
car.ico" processedpath="html/carp.ico">
</Icons>

```

When there are multiple icon files declared, the Hybrid Web Container chooses the best matched icon based on the device's capability.

3. Add the icon file reference under the `<HtmlFiles>` element, for example:


```
<HtmlFile>html/icon/ambulance32x32.png</HtmlFile>
```
4. Save the `manifest.xml` file.

Changing the Hybrid App Package Icon

Modify the Hybrid App package application icon.

You cannot add new icons to the folder, but you can replace the existing icon images, using the same file name. The Hybrid App icons are named `ampicon<index>.png`, where `<index>` is a number between 30 and 116. The icon `ampicon48.png` is the default Hybrid App icon. This is also the icon that is shown on the menu item that shows all the Hybrid Apps.

Each Hybrid App icon has two associated image files that contain images for processed and unprocessed messages; `ampicon<index>.png` (unprocessed messages) and `ampicon<index>p.png` (processed messages). Processed means the message has been submitted to the server.

When you build the Hybrid Web Container with custom icons, the original icons still appear in SAP Control Center and in SAP Mobile WorkSpace. You must remember the original icon, so you can select it in SAP Mobile WorkSpace and in SAP Control Center.

1. Identify the image currently used by the Hybrid App package that you want to replace:
 - a) Log in to SAP Control Center.
 - b) In **Workflows**, select the Hybrid App package for which to replace the image.
 - c) Click the **General** tab.

The icon is shown in **Display icon**.

2. Go to the `...\HybridWebContainer\res\drawable\` folder and find and replace the `ampicon<index>.png` and `ampicon<index>p.png` image files with the new images.

Note: The new image files must use the same name as those you replaced, including the file extension, and they must have the same resolution as the original images.

If you do not want to overwrite the icon entirely, make a copy of it using another name and move it out of the folder. Extra files in the `drawable` folder may interfere with resource indexing.

3. Rebuild the Hybrid Web Container project.

Android Hybrid Web Container Customization

Customize the look and feel and default behavior of the Android Hybrid Web Container.

Before getting started:

- Install the Android Development Tools (ADT) plug-in for Eclipse. See <http://developer.android.com/sdk/installing/installing-adt.html>.

Note: If you are also developing for BlackBerry, it is recommended that you do not install the BlackBerry Java Plug-in for Eclipse and the ADT plug-in in the same Eclipse environment.

- Build the Hybrid Web Container project as described in *Building the Android Hybrid Web Container Using the Provided Source Code*. The `HybridWebContainer` directory contains directories such as `libs`, as well as `images` and other files.

Documentation for the application (`com.sybase.hwc`) and the library (`com.sybase.hybridApp`) are included in the `docs` directory of the `HybridWebContainer` project.

Android Customization Touch Points

All code areas associated with Hybrid Web Container customizations are annotated with `ANDROID_CUSTOMIZATION_POINT_<customization>` comment tags, or touch points.

Touch Point	Description
<code>ANDROID_CUSTOMIZATION_POINT_COLORS</code>	Use custom colors for the Hybrid Web Container.
<code>ANDROID_CUSTOMIZATION_POINT_FONTS</code>	Use custom fonts in the Hybrid Web Container.
<code>ANDROID_CUSTOMIZATION_POINT_BRAND</code>	Change application name, copyright, and developer information
<code>ANDROID_CUSTOMIZATION_POINT_SPLASHSCREEN</code>	Add a splash screen to the Hybrid Web Container.

Touch Point	Description
ANDROID_CUSTOMIZATION_POINT_DEFAULTSETTINGS	Set the defaults for the Settings screen.
ANDROID_CUSTOMIZATION_POINT_PRESETSETTINGS	Hard code settings for the Settings screen so they do not show up on the device. This prevents the user from changing the settings.
ANDROID_CUSTOMIZATION_POINT_PREPACKAGED_APP	Run the Hybrid Web Container as a single Hybrid App.
ANDROID_CUSTOMIZATION_POINT_PIN	Use for PIN screen customizations, or to remove the PIN screen.
ANDROID_CUSTOMIZATION_POINT_SORTING	Sort Hybrid App messages based on different criteria.
ANDROID_CUSTOMIZATION_POINT_FILTERING	Filter the list of Hybrid App messages so only messages meeting certain criteria are shown.
ANDROID_CUSTOMIZATION_POINT_HYBRIDAPPSORT	Customize the criteria for how the Hybrid App list is sorted.
ANDROID_CUSTOMIZATION_POINT_HYBRIDAPPSEARCH	Make the list of Hybrid App packages searchable.
ANDROID_CUSTOMIZATION_POINT_HYBRIDAPPLIST	Customize the Hybrid App package list appearance.
ANDROID_CUSTOMIZATION_POINT_CATEGORIZEDVIEWS	Create categorized views of the Hybrid App packages.
ANDROID_CUSTOMIZATION_POINT_HTTPHEADERS	Set HTTP headers for the Android Hybrid Web Container to include authentication tokens.
ANDROID_CUSTOMIZATION_POINT_PUSH_NOTIFICATION	Customize how the Hybrid Web Container handles the push notification.
ANDROID_CUSTOMIZATION_POINT_ANONYMOUS_USER	<p>Returns whether or not anonymous user support is being used. Change to YES to allow clients to register anonymously.</p> <hr/> <p>Note: For this to work, the HWC application connection template must be configured to use the anonymous security configuration. See <i>Application Connection Templates</i> in <i>SAP Control Center for SAP Mobile Platform</i>.</p>

Look and Feel Customization of the Android Hybrid Web Container

Customizations you can make to the look and feel include changing the splash screen, changing the Hybrid App icons and name, changing the Hybrid App package icons, changing labels and text, adding support for new languages, and so on.

Changing the Android Hybrid Web Container Icon

Modify the icon shown on the home screen by replacing the icon image files.

Changing this icon also changes the image used on the About screen, and the image that sometimes shows up in the title bar.

The icon image files are located in these directories:

- ...\`HybridWebContainer\res\drawable-hdpi`
- ...\`HybridWebContainer\res\drawable-ldpi`
- ...\`HybridWebContainer\res\drawable-mdpi`

Go to each directory and replace the `icon.png` image file with another `.png` image of your choice.

Note: The new image files must use the same name as those you replaced, including the file extension, and they must have the same resolution as the original images.

Customizing the About Screen and Other Branding

Customize the About screen.

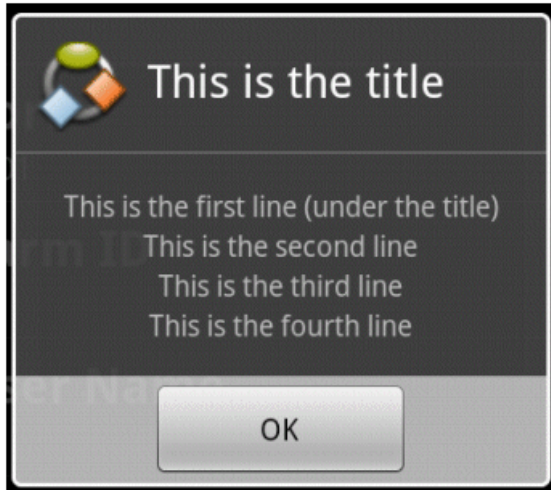
In some parts of the code, branding information is retrieved not from `strings.xml`, but from a constant in the `Brand` class. You cannot change these constants, but they are used only in a small number of places, and you can replace them where they are used. The `Brand` class is used mostly in the About screen, but there are a few other cases (all marked by the `ANDROID_CUSTOMIZATION_POINT_BRAND` comment tag).

1. Open the `CustomizationHelper.java` file, which is located in ...
 \`HybridWebContainer\src\com\sybase\hwc`.

This is where the strings in the About screen are set.

2. Locate the `customAbout` method.

Sample code is shown in this method. The default behavior is for the method to return false. The sample code produces the below dialog.



3. Uncomment the sample code, change the text to what you want to display, and change `return false;` to `return true;`.

Adding a Splash Screen

Add a splash screen to the Hybrid Web Container.

This procedure shows an example of a splash screen, which is the first screen that you see in the Hybrid Web Container. The related comment tag is

`ANDROID_CUSTOMIZATION_POINT_SPLASHSCREEN`.

1. Open the `SplashScreenActivity.java` file, which is located in the `... \HybridWebContainer\src\com\sybase\hwc` folder.
2. Edit `SplashScreenActivity.java`.
 - a) You must call **finish()** on the splash screen as soon as you are finished displaying the screen.
Currently this is done in the `onStart` method, so you must remove it from there.
 - b) Create an intent that launches the **EnterPasswordActivity** after **finish()** is called. You must do this even if you disable the PIN screen.
It is important that **finish()** is called first. Currently this is done in the `onStop` method.

Changing Labels and Text

You can customize most of the text found in labels, dialogs, or error messages used by the Hybrid Web Container.

1. Open the `strings.xml` file, which is located in `... \HybridWebContainer\res \values` for editing.

Hybrid Web Container Customization

This file contains the text for error messages, screen titles, screen labels, validation messages, and so on.

2. Make your changes and save the file.

Keep in mind that for any change you make, you must also make the same change for each language if you want your changes to translate across other languages. You must edit the `strings.xml` files located in the `values-<language_code>` folder for each language.

Adding a New Language

Add support to the Hybrid Web Container for a new language.

1. In the `...\HybridWebContainer\res` folder, create a new folder named `values-<xx>`, where `<xx>` is the ISO 639 code of the language, for example, `values-it`, for Italian.
2. Add a file called `strings.xml` to the new folder. Use the `strings.xml` file from the `values` folder as a template for the new `strings.xml` file.
3. Open the default `strings.xml` file, which is located in `...\HybridWebContainer\res\values` and use it as a template for the new `strings.xml` file.

You need not include strings that do not require localization in the new `strings.xml` file. Strings that are missing from a localization are pulled from the default `strings.xml` file.

The new language is used automatically by a device that is set to that language.

Using Custom Colors

Use custom colors to change the look of Hybrid App messages and the Hybrid Web Container.

These examples modify the colors of the Hybrid App messages. You can also use custom colors for the Hybrid Web Container using similar steps. The related comment tag for customizing colors is `ANDROID_CUSTOMIZATION_POINT_COLORS`.

1. Open the `colors.xml` file, which is located in `...\HybridWebContainer\res\values`, for editing.
2. Find the `ANDROID_CUSTOMIZATION_POINT_COLORS` comment tag and add these tags inside the resources tag:

```
<color name="hybridapp_message_title_color">#F23431</color>
<color name="hybridapp_message_from_color">#FF1111</color>
<color name="hybridapp_message_date_color">#3234F1</color>
```

3. Open the `workflowmessages.xml` file, which is located in `...\HybridWebContainer\res\layout`, for editing.
4. In the `msg_datetime` `TextView` tag, modify the `android:textColor` attribute to:

```
android:textColor="@color/hybridapp_message_date_color"
```

5. Make similar changes to the `msg_from` and the `msg_title` tags, using the color resource defined in step 2.

If you build the Hybrid Web Container without making any more changes, notice that the custom colors are used for `msg_datetime` and `msg_title`, but not for `msg_from`. This is because the color for `msg_from` is overridden by the Java code. To stop a custom attribute from being overridden:

- a) Select **Search > File** from the menu.

- b) For Containing text, enter `msg_from` and click **Search**.

The search result shows two files: `workflowmessages.xml` and `UiHybridAppMessagesScreen.java`.

- c) Open the `UiHybridAppMessagesScreen.java` file for editing.

- d) Search the file for "`msg_from`."

You will find this line: `TextView tf = (TextView) v.findViewById(R.id.msg_from);`

The `TextView` object `tf` represents `msg_from`.

- e) You are changing the color, so search for "`tf.setText`."

The search results return two occurrences because the color is set depending on whether the message has been read or not.

- f) Comment out both lines to ensure that `msg_from` is always the color you set in the `workflowmessages.xml` file. Save the file.

Using Custom Fonts

Customize fonts for Hybrid App messages and the Hybrid Web Container.

This example customizes the fonts for Hybrid App messages.

1. Create a new XML file named `attrs.xml` in the `...\HybridWebContainer\res\values\` folder.

2. Open the `attrs.xml` and add this code:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <declare-styleable name="com.sybase.hwc.CustomFontTextView" >
        <attr name="customFont" format="string"/>
    </declare-styleable>
</resources>
```

3. You cannot set the font attribute using the standard `TextView` control, so you must extend the `TextView` object by creating a new file named `CustomFontTextView.java`.

4. Add this code to the `CustomFontTextView.java` file:

```
package com.sybase.hwc;

import android.content.Context;
```

```
import android.widget.TextView;
import android.text.TextUtils;
import android.util.AttributeSet;
import android.content.res.TypedArray;
import android.graphics.Typeface;

public class CustomFontTextView extends TextView {

    public CustomFontTextView( Context oContext )
    {
        super( oContext );
    }

    public CustomFontTextView( Context oContext, AttributeSet
oAttrs )
    {
        super( oContext, oAttrs );
        setCustomFont( oContext, oAttrs,
R.styleable.com_sybase_hwc_CustomFontTextView,
R.styleable.com_sybase_hwc_CustomFontTextView_customFont );
    }

    private void setCustomFont( Context oContext, AttributeSet
oAttrs, int[] aiAttributeSet, int iFontId)
    {
        TypedArray taStyledAttributes =
oContext.obtainStyledAttributes( oAttrs, aiAttributeSet );
        String sCustomFont =
taStyledAttributes.getString( iFontId );
        if( !TextUtils.isEmpty( sCustomFont ) )
        {
            Typeface oTypeFace = null;

            try
            {
                oTypeFace = getFont( oContext, sCustomFont );
                setTypeface( oTypeFace );
            }
            catch (Exception e)
            {
                System.out.println( "Count not set font!" );
                // can't set the font
            }
        }
        else
        {
            System.out.println( "Custom font string was empty!" );
        }
    }

    private Typeface getFont( Context oContext, String
sCustomFont )
    {
        String sFullCustomFont = "fonts/" + sCustomFont;
        Typeface oTypeFace =
Typeface.createFromAsset( oContext.getAssets(),
```

```
sFullCustomFont );
    return oTypeFace;
}
}
```

5. Create a fonts folder in `... \HybridWebContainer \assets` and add the TTF font file to this new folder.

For example, Windows fonts are usually in `C: \Windows \Fonts \` if you want to use one of those.

6. Open the `workflowmessages.xml` file for editing and add this attribute to the `RelativeLayout` tag:

```
xmlns:custom="http://schemas.android.com/apk/res/com.sybase.hwc"
```

7. Find the `TextView` tag with the "ID `msg_from`" and change the tag from a `TextView` tag to a "`com.sybase.hwc.CustomFontTextView`" tag.

8. Add this attribute to the `com.sybase.hwc.CustomFontTextView` tag:

```
custom:customFont="<NAME_OF_YOUR_FONT_FILE.TTF>"
```

9. Repeat the above steps for tags with the "id `msg_title`" and "id `msg_datetime`."

If you build the Hybrid Web Container without making any more changes, you see that "msg_title" and "msg_datetime" are shown with the custom font, but "msg_from" is not. This is because the font for "msg_from" is overridden in the Java code.

10. To prevent the font from being overridden:

- a) Select **Search > File** from the menu.
- b) For **Containing text**, enter `msg_from` and click **Search**.

The search result shows two files: `workflowmessages.xml` and `UiHybridAppMessagesScreen.java`.

- c) Open the `UiHybridAppMessagesScreen.java` file for editing.
- d) Search the file for "msg_from."

You will find this line: `TextView tf = (TextView) v.findViewById(R.id.msg_from);`

The `TextView` object `tf` represents `msg_from`.

- e) You are changing the font, so search for "tf.setTypeface."

The search results return two occurrences because the text is either bolded or not depending on whether the message has been read. Set bold, italic, or normal style for the text in the same way you specify the font.

- f) To ensure your custom font is used, make these modifications to the two occurrences of the method calls to **setTypeface**:

```
tf.setTypeface( tf.getTypeface(), Typeface.BOLD );
tf.setTypeface( tf.getTypeface(), Typeface.NORMAL );
```

Default Behavior Customization for the Android Hybrid Web Container

Default behavior that you can change includes removing a PIN screen, configuring default values for the Settings screen, sorting Hybrid App messages, and so on.

Removing Fields from the Settings Screen

You can hard-code settings for the Settings screen so they do not appear on the Settings screen on the device.

The comment tag associated with the fields on the Settings screen is

ANDROID_CUSTOMIZATION_POINT_DEFAULTSETTINGS.

1. Open the `CustomizationHelper.java` file, which is located in the . . .
 `\HybridWebContainer\src\com\sybase\hwc` folder.
2. All of the settings screen customization functionality is grouped together under this comment in the file:

```
//-----  
-----  
    // Setting screen customization methods  
    //-----  
-----
```

3. To remove a field, set the associated property to false.

For example, if you want to remove the user name field, change:

```
public boolean isConnectionUserNameVisible()  
{  
    return true;  
}
```

to

```
public boolean isConnectionUserNameVisible()  
{  
    return false;  
}
```

Configuring Default Values for the Settings Screen

Set default values for the Settings screen.

The comment tag associated with customizations of the default settings is

ANDROID_CUSTOMIZATION_POINT_DEFAULTSETTINGS.

1. Open the `CustomizationHelper.java` file, which is located in the . . .
 `\HybridWebContainer\src\com\sybase\hwc` folder.
2. Find the collection of methods named with the pattern
 `getDefaultConnection<setting_name>` or
 `isDefaultConnect<setting_name>`, where `<setting_name>` is the name of the
 setting.

3. Edit the methods to return the specific value you require.

The save button on the settings screen is enabled only when all of the fields requiring values are populated and a field is changed by the user, so if you change the return value for all of the methods to values that users do not have to modify on the device, you can run into a problem. To avoid this issue:

- a) Find the method in CustomizationHelper named `isSettingsSaveButtonAlwaysEnabled()`, which, by default, returns **false**.
- b) Change the method to return **true** so the save button is always enabled if all of the fields requiring values are populated.

Removing the PIN Screen

Remove the PIN screen (password screen) from the Hybrid Web Container.

The related comment tag is `ANDROID_CUSTOMIZATION_POINT_PIN`.

Note: Removing the PIN screen leaves data that is stored on the device less secure. You should remove the PIN screen only if you are not concerned about keeping your data secure.

1. Open the `CustomizationHelper.java` file, which is located in the `... \HybridWebContainer\src\com\sybase\hwc` folder.
2. Find the `enablePIN` method.
By default it returns **true** and shows the password screen.
3. Change the `enablePIN` method to return **false**.
The application does not show a password screen if it has been idle and is reactivated.
4. Test the application.

Using Multiple Hybrid Web Containers on the Same Android Device

Configure the Hybrid Web Container so that two or more Hybrid Web Containers co-exist on the same Android device.

1. Open the `AndroidManifest.xml` file, which is located under the HybridWebContainer project folder.
2. In the `manifest` tag, change the `"com.sybase.hwc"` package attribute to something else.
3. Search the file and change any references to `"com.sybase.hwc"` to the new package from step 2.

Note: Do not change any references to `com.sybase.hybridApp`, as these refer to the library jar files.

4. Save the file and choose **Yes** when asked if you want to change your launch configuration.
5. Change to the Eclipse Java perspective.

6. Right-click the package under `src` (it will be the old package name, `com.sybase.hwc`) and choose **Refactor > Rename**.
7. Set the name to be the package name you set in step 2.
8. Open the `CustomizationHelper.java` file, which is located in `... \HybridWebContainer\src\com\sybase\hwc`, and find the method named `getAppId()`:
By default `getAppId()` returns `Brand.OEM_HYBRIDAPP_APPID`. Change it to return a `String` that uniquely identifies your application.
9. You must now add an application with a matching App id in SAP Control Center, and if you want to use the automatic registration option, you must also add an Application Connection Template.
Now when you build the Hybrid Web Container, you can install it on a device that already has a Hybrid Web Container installed (but with a different package name). You should make other changes to your new Hybrid Web Container, such as `app_short_name` in the `strings.xml` file, or the icon `.png` image, to differentiate the Hybrid Web Containers on the device.

Sorting the List of Hybrid Apps

You can sort and filter the list of Hybrid Apps.

By default, the Hybrid Web Container displays Hybrid App packages in alphabetical order by package name. This procedure shows how to change the list so that it is case-sensitive. The related comment tag is `ANDROID_CUSTOMIZATION_POINT_HYBRIDAPPSORT`.

1. Open the `CustomizationHelper.java` file, which is located in the `... \HybridWebContainer\src\com\sybase\hwc` folder.
2. Find the `getHybridAppComparator()` method.
The comparator is used to order application (`HybridApp`) objects and is called by `sort`.
3. Modify the comparator to order the applications to meet your requirements.
4. Save the file.

Sorting Hybrid App Messages

Sort Hybrid App messages based on different criteria.

The comment tag associated with sorting Hybrid App messages is `ANDROID_CUSTOMIZATION_POINT_SORTING`.

1. Open the `CustomizationHelper.java` file, which is located in the `... \HybridWebContainer\src\com\sybase\hwc` folder.
2. Find the `getMessageComparator()` method.
The comparator is used to order `Message` objects and is called by `sort`.
3. Modify the comparator to order the messages to meet your requirements.

4. Save the file.

Filtering the Hybrid App Messages

Filter the list of Hybrid App messages so only messages that meet specified criteria are shown.

The comment tag associated with Hybrid App messages is

ANDROID_CUSTOMIZATION_POINT_FILTERING.

1. Open the `CustomizationHelper.java` file, which is located in the `...`
`\HybridWebContainer\src\com\sybase\hwc` folder.

2. Find the `getFilteredMessages()` method.

The default behavior is to return all messages.

3. To return a subset of messages, you can modify `getFilteredMessages()` to return a list of messages based on your criteria.

For example, if you want only high priority messages to appear in the message list, you can change the code to the following:

```
// Display high priority messages only.
ArrayList<Message> filteredMessages =
MessageDb.getMessages( bCompleteList );
for( int iMessageIndex = 0; iMessageIndex <
filteredMessages.size(); iMessageIndex++ )
{
if( filteredMessages.get(iMessageIndex).getMailPriority() !=
com.sybase.mo.AmpConsts.EMAIL_STATUS_IMPORTANCE_HIGH )
{
filteredMessages.remove(iMessageIndex);
//we need to decrement the index so we don't skip an element now
iMessageIndex--;
}
}
return filteredMessages;
```

You must refresh the listview before the new messages are filtered. You can refresh the listview by switching to another view and then switching back.

Setting HTTP Headers

You can set HTTP headers for the Android Hybrid Web Container to include authentication tokens.

There are three sample methods showing how to do this in the Android Hybrid Web Container template source code, which include:

- `setHttpHeaders()` – use this method to set the authentication tokens. The tokens you set are used until `setHttpHeaders` is called again.
- `setHybridAppTokenErrorListener()` – use this method to call `setHttpHeaders()` to put the authentication tokens back in a good state, if, for example, they have expired.

Hybrid Web Container Customization

- `setHttpErrorListener()` – use this method to handle HTTP errors.

The comment tag associated with setting HTTP headers is `ANDROID_CUSTOMIZATION_POINT_HTTPHEADERS`.

1. Open the `CustomizationHelper.java` file and make your changes.
2. Save the file.

Modifying the Hybrid App List Appearance

Change how the Hybrid Apps are shown on the device.

The comment tag associated with customizing the Hybrid App list appearance is `ANDROID_CUSTOMIZATION_POINT_HYBRIDAPPLIST`.

To show the list of applications, the Hybrid Web Container calls the `getHybridAppScreenClass()` method in `CustomizationHelper.java`. That method returns the class that displays the list. The default class is `UiHybridAppScreen`.

1. To make small changes to the list view, open the `UiHybridAppScreen.java` file, which is located in the `...\HybridWebContainer\src\com\sybase\hwc` folder, and make your changes.

Note: Optionally, you can create your own class that extends `UiHybridAppScreen`. If you do this, you must modify the `getHybridAppScreenClass()` method in the `CustomizationHelper` file to return the name of your new class.

2. Save the file.

Creating a Gallery View

Change the Hybrid App Package list view to a gallery view.

The comment tag associated with creating categorized views is `ANDROID_CUSTOMIZATION_POINT_HYBRIDAPPLIST`.

1. Add an XML layout called `hybridappgallery.xml` to the `HybridWebContainer` project.
2. Match your `hybridappgallery.xml` layout to:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <Gallery xmlns:android="http://schemas.android.com/apk/res/
android"
        android:id="@+id/gallery"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
```

```
</LinearLayout>
```

3. Create a new activity for the HybridWebContainer.
 - a) Open the `AndroidManifest.xml` file.
 - b) Click the **Application** tab.
 - c) In the Application Nodes section (at the bottom left), click **Add**.
 - d) Choose **Activity** and click **OK**.
 - e) Select the new activity and change its name to `com.sybase.hwc.HybridAppGalleryActivity`.
 - f) Click **Name*** to generate the stub Java file.
 - g) Click **Finish**.
4. Enter this code into the `HybridAppGalleryActivity.java` file:

```
package com.sybase.hwc;

import java.util.ArrayList;
import java.util.Vector;
import java.util.Arrays;

import com.sybase.hybridApp.*;
import com.sybase.hybridApp.amp.Consts;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;

public class HybridAppGalleryActivity extends Activity {

    ImageAdapter m_adapter;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.hybridappgallery);

        Gallery oGallery = (Gallery) findViewById(R.id.gallery);
        m_adapter = new ImageAdapter(this);
        oGallery.setAdapter(m_adapter);

        oGallery.setOnItemClickListener(new OnItemClickListener ()
        {
            public void onItemClick(AdapterView parent, View v, int
```

Hybrid Web Container Customization

```
position, long id)
    {
        startHybridApp(parent, v, position, id);
    }
});
}

public void startHybridApp(AdapterView oParent, View v, int
iPos, long id )
{
    Intent oIntentHybridAppContainer = new Intent( this,
UiHybridAppContainer.class );

oIntentHybridAppContainer.putExtra( Consts.INTENT_PARAM_HYBRIDAPP
_START_MODE, Consts.START_MODE_HYBRIDAPP );

oIntentHybridAppContainer.putExtra( Consts.INTENT_PARAM_HYBRIDAPP
_ID, m_adapter.getItem( iPos ).getHybridAppId() );

oIntentHybridAppContainer.putExtra( Consts.INTENT_PARAM_HYBRIDAPP
_PROGRESS_TEXT, m_adapter.getItem( iPos ).getDisplayName() );
    startActivityResult( oIntentHybridAppContainer,
Consts.INTENT_ID_HYBRIDAPP_CONTAINER );
}

@Override
public void onActivityResult( int iRequestCode, int
iResultCode, Intent relaunchData )
{
    super.onActivityResult( iRequestCode, iResultCode,
relaunchData );
    if ( iRequestCode == Consts.INTENT_ID_HYBRIDAPP_CONTAINER &&
iResultCode == Consts.RESULT_RELAUNCH )
    {
        Intent oIntentHybridAppContainer = new Intent( this,
UiHybridAppContainer.class );

oIntentHybridAppContainer.putExtra( Consts.INTENT_PARAM_HYBRIDAPP
_START_MODE, Consts.START_MODE_HYBRIDAPP );

oIntentHybridAppContainer.putExtra( Consts.INTENT_PARAM_HYBRIDAPP
_ID, relaunchData.getIntExtra( Consts.INTENT_PARAM_HYBRIDAPP_ID,
0 ));

oIntentHybridAppContainer.putExtra( Consts.INTENT_PARAM_HYBRIDAPP
_PROGRESS_TEXT,
relaunchData.getStringExtra( Consts.INTENT_PARAM_HYBRIDAPP_PROGRE
SS_TEXT ));
        startActivityResult( oIntentHybridAppContainer,
Consts.INTENT_ID_HYBRIDAPP_CONTAINER );
    }
}

public class ImageAdapter extends BaseAdapter
{
```

```

//int mGalleryItemBackground;
private Context mContext;
private Vector<HybridApp> mHybridApps;

private ArrayList<Integer> mImageIds;

public ImageAdapter(Context c)
{
    mContext = c;
    mImageIds = new ArrayList<Integer>();

    //have to get a list of all installed HybridAppss
    mHybridApps = new
Vector<HybridApp>( Arrays.asList (HybridAppDb.getInvocableHybridAp
ps(false)) );
    for(int iHybridAppIndex = 0; iHybridAppIndex <
mHybridApps.size(); iHybridAppIndex++)
    {
        HybridAppDb oHybridApp = (HybridAppDb)
mHybridApps.get(iHybridAppIndex);
        int iconIndex = oHybridApp.getIconIndex();
        if(iconIndex >= 30 &&
            iconIndex <= 116)
        {
            //luckily the icon resources are consecutive
            int iResource = 0;
            if(iconIndex < 100)
            {
                iResource = 0x7f020022;
                iResource += (iconIndex - 30)*2;
            }
            else
            {
                iResource = 0x7f020000;
                iResource += (iconIndex - 100)*2;
            }
            mImageIds.add(new Integer(iResource));
        }
    }
}

public int getHybridAppId(int position)
{
    return
((HybridAppDb)mHybridApps.get(position)).getHybridAppId();
}

public String getDisplayName(int position)
{
    return
((HybridAppDb)mHybridApps.get(position)).getDisplayName();
}

public int getCount()
{
    return mImageIds.size();
}

```

```
    }

    public HybridAppDb getItem(int position)
    {
        return (HybridAppDb)mHybridApps.get(position);
    }

    public long getItemId(int position)
    {
        return position;
    }

    public View getView(int position, View convertView, ViewGroup
parent)
    {
        ImageView imageView = new ImageView(mContext);

        imageView.setImageResource(mImageIds.get(position).intValue());
        imageView.setLayoutParams(new
Gallery.LayoutParams(150,100));
        imageView.setScaleType(ImageView.ScaleType.FIT_XY);

        return imageView;
    }
}
}
```

5. Save the file.
6. Open the `CustomizationHelper.java` file, which is located in the `... \HybridWebContainer\src\com\sybase\hwc` folder and edit the `getHybridAppScreenClass()` method, to change the class returned to your new class.
(Android only) That class must extend **Activity**.
7. (Android only) Update the `manifest.xml` file to include the new activity you create.

Creating Categorized Views

Create categories so that Hybrid Apps and messages appear in lists under a category heading.

The comment tag associated with creating categorized views is

```
ANDROID_CUSTOMIZATION_POINT_CATEGORIZEDVIEWS.
```

First, determine names for the categories. SAP recommends that you name the final category “Miscellaneous;” this adds all applications and messages that do not match a category to the Miscellaneous category. Also in this example, all applications that belong to a category must include the category name contained in their display name. For example, an application named “Financial Claim” belongs in the “Financial” category.

There are other ways to determine categories; if you know the names of the applications in advance, you can simply list all the application names that belong in each category.

1. Create a new XML layout called `category.xml` and paste the following code into the auto generated file:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    android:layout_width="fill_parent"
    android:layout_height="?android:attr/listPreferredItemHeight"
    android:padding="6dip">

    <LinearLayout xmlns:android="http://schemas.android.com/apk/res/
    android"
        android:orientation="vertical"
        android:layout_width="0dip"
        android:layout_weight="1"
        android:layout_height="fill_parent">
        <TextView
            android:id="@+id/category"
            android:layout_width="fill_parent"
            android:layout_height="0dip"
            android:layout_weight="1"
            android:singleLine="true"
            android:ellipsize="marquee"
            android:gravity="center_vertical"
            />
        </LinearLayout>
    </LinearLayout>
</LinearLayout>
```

2. Copy the `UiHybridAppScreen.java` file and rename it to your own class, for example, `CategorizedAppScreen.java`, and open it for editing.
3. Add the list of categories to the `UiHybridAppScreen` class, as a public static final member variable:

```
public static final String[] m_asHybridAppCategories =
{ "Financial", "Utilities", "Miscellaneous" };
```

4. Replace the `HybridAppAdapter` class with:

```
private class HybridAppAdapter extends ArrayAdapter<Object>
{
    private String[] m_asCategories;

    public HybridAppAdapter( Context context, int
textviewResourceId, List<Object> items, String[] categories ){
        super( context, textviewResourceId, items );

        m_asCategories = categories;

        for( int index = 0; index < m_asCategories.length; index
++ )
        {
            this.add( m_asCategories[index] );
        }
    }

    @Override
```

```

public View getView(int position, View convertView,
ViewGroup parent)
{
    Object oObject = this.getItem(position);
    View v = null;
    if( oObject instanceof HybridApp )
    {
        HybridApp oHybridApp = ( HybridApp ) oObject;
        LayoutInflater vi =
(LayoutInflater) getSystemService( Context.LAYOUT_INFLATER_SERVICE)
;
        v = vi.inflate(R.layout.workflows, null);

        if ( oHybridApp != null )
        {
            ImageView ic = (ImageView)
v.findViewById( R.id.workflow_icon );

ic.setImageResource( UiIconIndexLookup.getNormalIconIdForIndex( o
HybridApp.getIconIndex() ));
            TextView tt = (TextView)
v.findViewById(R.id.workflow_title);
            if ( tt != null ) {
                tt.setText( oHybridApp.getDisplayName() );
            }
        }
    }
    else
    { //This position is not a HybridApp, but a category
heading
        String sString = ( String ) oObject;
        LayoutInflater vi = ( LayoutInflater )
getSystemService( Context.LAYOUT_INFLATER_SERVICE );
        v = vi.inflate( R.layout.category, null );
        if( sString != null )
        {
            TextView tt = (TextView)
v.findViewById( R.id.category );
            if ( tt != null )
            {
                tt.setText( sString );
            }
        }
    }
    return v;
}

public void remove( HybridApp oApp )
{
    // The object to remove has a different pointer
    // so match it up with the one in the list
    for ( int i = 0; i < this.getCount(); i++ )
    {
        Object oObject = getItem( i );
        if( oObject instanceof HybridApp )
        {

```



```

        HybridApp oTemp = ( HybridApp ) oObject;

        if ( oApp.getModuleId() == oTemp.getModuleId()
            && oApp.getVersion() == oTemp.getVersion() )
        {
            super.remove( oTemp );
            return;
        }
    }
}

public void sort()
{
    // Sorts applications by name
    this.sort( new Comparator<Object>()
    {
        @Override
        public int compare( Object oObject1, Object
oObject2 )
        {
            if( oObject1 instanceof String && oObject2
instanceof String)
            {
                String sString1 = ( String ) oObject1;
                String sString2 = ( String ) oObject2;
                for( int index = 0; index < m_asCategories.length;
index++ )
                {
                    if( sString1.equals( m_asCategories[index] ) )
                    {
                        return -1;
                    }
                    if( sString2.equals( m_asCategories[index] ) )
                    {
                        return 1;
                    }
                }
            }
            else if( oObject1 instanceof HybridApp && oObject2
instanceof HybridApp )
            {
                HybridApp oHybridApp1 = ( HybridApp ) oObject1;
                HybridApp oHybridApp2 = ( HybridApp ) oObject2;

                int iCategoryIndex1 =
getCategoryIndex( oHybridApp1 );
                int iCategoryIndex2 =
getCategoryIndex( oHybridApp2 );

                if( iCategoryIndex1 == iCategoryIndex2 )
                {

```

```

        return
oHybridApp1.getDisplayName().toLowerCase().compareTo( oHybridApp2
.getDisplayName().toLowerCase() );
    }
    else
    {
        return iCategoryIndex1 - iCategoryIndex2;
    }
}
else
{ //we have one String (category heading) and one
HybridApp
    HybridApp oHybridApp = null;
    String sString = null;
    int iSwitch = 1;
    if( oObject1 instanceof HybridApp)
    {
        oHybridApp = ( HybridApp ) oObject1;
        sString = ( String ) oObject2;
    }
    else
    {
        oHybridApp = ( HybridApp ) oObject2;
        sString = ( String ) oObject1;
        iSwitch = -1;
    }

    int iHybridAppCategoryIndex =
getCategoryIndex( oHybridApp );
    int iCategoryIndex = getCategoryIndex( sString );
    if( iCategoryIndex <= iHybridAppCategoryIndex )
    {
        return 1*iSwitch;
    }
    else
    {
        return -1*iSwitch;
    }
}

return 0;
}

private int getCategoryIndex( String sString )
{
index++ )
    for( int index = 0; index < m_asCategories.length;
        {
if( m_asCategories[index].equalsIgnoreCase( sString ) )
    {
        return index;
    }
}
return m_asCategories.length - 1;
}

```

```

        }

        private int getCategoryIndex( HybridApp oHybridApp )
        {
            for( int index = 0; index < m_asCategories.length;
index++ )
                {
                    if( oHybridApp.getDisplayName().toLowerCase().indexOf( m_asCategories[index].toLowerCase() ) >= 0 )
                        {
                            return index;
                        }
                }
            return m_asCategories.length - 1;
        }
    });
}
}

```

5. In the `onResume` method, make modifications to the following line (changes are shown in **bold**):

```

this.m_adapter = new HybridAppAdapter( this, R.layout.workflows,
new
ArrayList<Object>(Arrays.asList( HybridAppDb.getInvocableHybridApps(false) ) ), m_asHybridAppCategories );

```

6. Modify the `onListItemClick` method as shown in the example code (changes are shown in **bold**):

```

public void onListItemClick( ListView oParent, View v, int iPos,
long id )
{
    Object oObject = m_adapter.getItem( iPos );
    if( oObject instanceof HybridApp )
    {
        HybridApp oHybridApp = ( HybridApp ) oObject;
        Intent oIntentHybridAppContainer = new Intent( this,
UiHybridAppContainer.class );

        oIntentHybridAppContainer.putExtra( Consts.INTENT_PARAM_HYBRIDAPP
_START_MODE, Consts.START_MODE_HYBRIDAPP );

        oIntentHybridAppContainer.putExtra( Consts.INTENT_PARAM_HYBRIDAPP
_ID, ((HybridAppDb) oHybridApp).getHybridAppId() );

        oIntentHybridAppContainer.putExtra( Consts.INTENT_PARAM_HYBRIDAPP
_PROGRESS_TEXT, oHybridApp.getDisplayName() );
        startActivityForResult( oIntentHybridAppContainer,
Consts.INTENT_ID_HYBRIDAPP_CONTAINER );
    }
}

```

7. Save the file.

8. Open the `UiHybridAppMessagesScreen.java` file for editing, and in the `onCreateContextMenu` method, make these modifications (changes are shown in **bold**):

```
public void onCreateContextMenu( ContextMenu oMenu, View v,
ContextMenu.ContextMenuInfo menuInfo)
{
    super.onCreateContextMenu( oMenu, v, menuInfo );

    AdapterContextMenuInfo oInfo = (AdapterContextMenuInfo)
menuInfo;
    Object oObject = m_adapter.getItem( oInfo.position );
    if( oObject instanceof Message )
    {
        Message oMsg = ( Message ) oObject;

        oMenu.setHeaderTitle( oMsg.getSubject() );
        oMenu.add( 0, CONTEXT_MENU_DELETE, 0,
R.string.Context_Menu_Delete );

        // Save the id for operations used in the context menu
        m_iContextMessageId = oMsg.getMessageId();
    }
}
```

9. In the `onContextItemSelected` method, make these modifications (changes are shown in **bold**):

```
public boolean onContextItemSelected( MenuItem oItem )
{
    if ( oItem.getItemId() == CONTEXT_MENU_DELETE )
    {
        AdapterContextMenuInfo oInfo = (AdapterContextMenuInfo)
oItem.getMenuInfo();

        // The message might have been deleted while the context
menu was open.
        // Make sure the position is still present and matches
the id we expect
        if ( oInfo.position < m_adapter.getCount() )
        {
            Object oObject = m_adapter.getItem( oInfo.position );
            if( oObject instanceof Message )
            {
                Message oMsg = ( Message ) oObject;

                if ( oMsg.getMessageId() == m_iContextMessageId )
                {
                    // Remove message from database
                    MessageDb.delete( oMsg.getMessageId() );
                }
            }
        }
        return true;
    }
    return false;
}
```

```

    }

```

10. Replace the MessageAdapter class:

```

private class MessageAdapter extends ArrayAdapter<Object>
{
    String[] m_asCategories;

    public MessageAdapter( Context context, int
textViewResourceId, ArrayList<Object> items, String[]
categories ){
        super( context, textViewResourceId, items );

        m_asCategories = categories;

        for( int index = 0; index < m_asCategories.length; index
++ )
        {
            this.add( m_asCategories[index] );
        }
    }

    @Override
    public View getView(int position, View convertView,
ViewGroup parent) {
        Object oObject = getItem( position );
        View v = null;
        if( oObject instanceof Message )
        {
            Message oMsg = (Message) oObject;
            LayoutInflater vi =
(LayoutInflater) getSystemService( Context.LAYOUT_INFLATER_SERVICE)
;
            v = vi.inflate(R.layout.workflowmessages, null);

            if ( oMsg != null )
            {
                //set the Hybrid App message priority icon
                ImageView imageForPriority = (ImageView)
v.findViewById( R.id.priority_icon );

                if ( oMsg.getMailPriority() ==
AmpConsts.EMAIL_STATUS_IMPORTANCE_HIGH )
                {
                    imageForPriority.setImageResource( R.drawable.readhi );
                    imageForPriority.setVisibility( View.VISIBLE );
                }
                else if ( oMsg.getMailPriority() ==
AmpConsts.EMAIL_STATUS_IMPORTANCE_LOW )
                {
                    imageForPriority.setImageResource( R.drawable.readlow );
                    imageForPriority.setVisibility( View.VISIBLE );
                }
            }
        }
    }
}

```

```

    }
    else
        imageForPriority.setVisibility( View.GONE );

        ImageView ic = (ImageView)
v.findViewById( R.id.msg_icon );
        if ( oMsg.isMsgProcessed() )

ic.setImageResource( UiIconIndexLookup.getProcessedIconIdForIndex
( oMsg.getIconIndex() ));
        else

ic.setImageResource( UiIconIndexLookup.getNormalIconIdForIndex( o
Msg.getIconIndex() ));
        TextView tf = (TextView)
v.findViewById(R.id.msg_from);
        TextView tt = (TextView)
v.findViewById(R.id.msg_title);
        TextView bt = (TextView)
v.findViewById(R.id.msg_datetime);
        if ( tf != null ) {
            tf.setText( oMsg.getMsgFrom() );
        }
        if (tt != null) {
            tt.setText( oMsg.getSubject());
        }
        if(bt != null){
            Calendar dtReceived =
Calendar.getInstance();

dtReceived.setTime( oMsg.getReceivedDate() );

            Calendar dtNow = Calendar.getInstance();

            if ( dtNow.get( Calendar.YEAR ) ==
dtReceived.get( Calendar.YEAR ) &&
                dtNow.get( Calendar.MONTH ) ==
dtReceived.get( Calendar.MONTH ) &&
                dtNow.get( Calendar.DAY_OF_MONTH ) ==
dtReceived.get( Calendar.DAY_OF_MONTH ) )
            {
                bt.setText( ( new
SimpleDateFormat( "hh:mm
a" ) ).format( oMsg.getReceivedDate() ) );
            }
            else {
                bt.setText( ( new SimpleDateFormat( "MM/
dd/yy" ) ).format( oMsg.getReceivedDate() ) );
            }
        }

        // Update appearance unread messages
        if ( tf != null && tt != null && bt != null )
        {
            if ( !oMsg.isMsgRead() )
            {

```

```

// Setup view for unread message
v.setBackgroundResource( R.drawable.unread_selector );

        tf.setTextColor( Color.WHITE );
        tf.setTypeface( null, Typeface.BOLD );
    }
    else
    {
        // Setup view for read message
        v.setBackgroundResource( 0 );

        TypedValue tv = new TypedValue();

getTheme().resolveAttribute( android.R.attr.textColorSecondary,
tv, true );

tf.setTextColor( getResources().getColor( tv.resourceId ) );
        tf.setTypeface( null, Typeface.NORMAL );
    }
}
}
else
{
    String sString = ( String ) oObject;
    LayoutInflater vi = ( LayoutInflater )
getSystemService( Context.LAYOUT_INFLATER_SERVICE );
    v = vi.inflate( R.layout.category, null );
    if( sString != null )
    {
        TextView tt = (TextView)
v.findViewById( R.id.category );
        if ( tt != null )
        {
            tt.setText( sString );
        }
    }
}
return v;
}

public void sort()
{
    // Sorts applications by name
    this.sort( new Comparator<Object>()
    {
        @Override
        public int compare( Object oObject1, Object
oObject2 )
        {
            if( oObject1 instanceof String && oObject2
instanceof String)
            {
                String sString1 = ( String ) oObject1;

```

```

        String sString2 = ( String ) oObject2;
        for( int index = 0; index <
m_asCategories.length; index++ )
        {

if( sString1.equals( m_asCategories[index] ) )
        {
            return -1;
        }

if( sString2.equals( m_asCategories[index] ) )
        {
            return 1;
        }
        }
        else if( oObject1 instanceof Message && oObject2
instanceof Message )
        {
            Message oMessage1 = ( Message ) oObject1;
            Message oMessage2 = ( Message ) oObject2;

            int iCategoryIndex1 =
getCategoryIndex( oMessage1 );
            int iCategoryIndex2 =
getCategoryIndex( oMessage2 );

            if( iCategoryIndex1 == iCategoryIndex2 )
            {
                return
oMessage1.getReceivedDate().compareTo( oMessage2.getReceivedDate(
) );
            }
            else
            {
                return iCategoryIndex1 - iCategoryIndex2;
            }
        }
        else
        { //we have one String (category heading) and one
HybridApp
            Message oMessage = null;
            String sString = null;
            int iSwitch = 1;
            if( oObject1 instanceof Message)
            {
                oMessage = ( Message ) oObject1;
                sString = ( String ) oObject2;
            }
            else
            {
                oMessage = ( Message ) oObject2;
                sString = ( String ) oObject1;
                iSwitch = -1;
            }
        }
    }

```



```

        int iMessageCategoryId =
getCategoryId( oMessage );
        int iCategoryId = getCategoryIndex( sString );
        if( iCategoryId <= iMessageCategoryId )
        {
            return 1*iSwitch;
        }
        else
        {
            return -1*iSwitch;
        }
    }

    return 0;
}

private int getCategoryIndex( String sString )
{
for( int index = 0; index < m_asCategories.length;
index++ )
    {
if( m_asCategories[index].equalsIgnoreCase( sString ) )
        {
            return index;
        }
    }
    return m_asCategories.length - 1;
}

private int getCategoryIndex( Message oMessage )
{
    MessageDb oMessageDb = (MessageDb) oMessage;
    if( oMessageDb != null )
    {
        HybridApp oHybridApp =
HybridAppDb.getHybridApp(oMessage.getModuleId(),
oMessage.getModuleVersion());
        String sModuleName =
oHybridApp.getDisplayName();
        if( sModuleName != null )
        {
            for( int index = 0; index <
m_asCategories.length; index++ )
            {
if( sModuleName.toLowerCase().indexOf( m_asCategories[index].toLo
werCase() ) >= 0 )
                {
                    return index;
                }
            }
        }
    }
}
}

```

```

        return m_asCategories.length - 1;
    }
    });
}
}

```

11. In the **onResume** method, make these changes (changes are shown in **bold**):

```

try
{
    // ANDROID_CUSTOMIZATION_POINT_FILTERING
    ArrayList<Message> alMessages = MessageDb.getMessages();
    ArrayList<Object> alMessagesObjects = new
ArrayList( alMessages );
    this.m_adapter = new MessageAdapter( this,
R.layout.workflowmessages, alMessagesObjects,
UiHybridAppScreen.m_asHybridAppCategories );

    this.m_adapter.sort();
}

```

12. In the **onListItemClick** method, make these modifications (changes are shown in **bold**):

```

public void onListItemClick(ListView oParent, View v, int iPos,
long id )
{
    try
    {
Object oObject = m_adapter.getItem( iPos );
if( oObject instanceof Message )
    {
        Message oMsg = ( Message ) oObject;

        // Check if Hybrid App is available
        HybridApp oHybridApp =
HybridAppDb.getHybridApp( oMsg.getModuleId(),
oMsg.getModuleVersion());

        // CR668069 -Check if we can handle transform data -
lmb limit by sqllite database
        try
        {
            oMsg.getTransformData();
        }
        catch ( Exception ex )
        {
            MocaLog.getAmpHostLog().logMessage( "Failed to
read transform data", MocaLog.eMocaLogLevel.Normal );

            new AlertDialog.Builder( this )
                .setTitle( android.R.string.dialog_alert_title )
                .setMessage( R.string.IDS_MSG_ERR_MESSAGE_TOO_L
ARGE )

                .setIcon( android.R.drawable.ic_dialog_alert )
                .setPositiveButton( android.R.string.ok,
                    new DialogInterface.OnClickListener()

```

```

        {
        public void onClick( DialogInterface dialog, int
whichButton)
        {
            dialog.dismiss();
        }
    } )
    .show();

    return;
}

// Update read flag
if ( !oMsg.isMsgRead() )
{
    m_adapter.notifyDataSetChanged();
}

// Open Hybrid App
Intent oIntentHybridAppContainer = new Intent( this,
UiHybridAppContainer.class );

oIntentHybridAppContainer.putExtra( Consts.INTENT_PARAM_HYBRIDAPP
_START_MODE, Consts.START_MODE_MESSAGE );

oIntentHybridAppContainer.putExtra( Consts.INTENT_PARAM_HYBRIDAPP
_MSG_ID, oMsg.getMessageId() );

oIntentHybridAppContainer.putExtra( Consts.INTENT_PARAM_HYBRIDAPP
_MODULE_ID, oMsg.getModuleId() );

oIntentHybridAppContainer.putExtra( Consts.INTENT_PARAM_HYBRIDAPP
_MODULE_VERSION, oMsg.getModuleVersion() );

oIntentHybridAppContainer.putExtra( Consts.INTENT_PARAM_HYBRIDAPP
_PROGRESS_TEXT, oMsg.getSubject() );
startActivityForResult( oIntentHybridAppContainer,
Consts.INTENT_ID_HYBRIDAPP_CONTAINER );
    }
}
catch( Exception ex )
{
    MocaLog.getAmpHostLog().logMessage( "Failed to open
message. Caught exception - " + ex.getMessage(),
MocaLog.eMocaLogLevel.Normal );
}
}
}

```

- Open the CustomizationHelper.java file, which is located in the ... \HybridWebContainer\src\com\sybase\hwc folder and edit the getHybridAppScreenClass() method, to change the class returned to your new class, which you created in step 2.

That class must extend **Activity**.

14. (Android only) Update the `manifest.xml` file to include the new activity you create.

Making the List of Hybrid App Packages Searchable

Make the list of Hybrid App packages searchable.

The comment tag associated with making the list of Hybrid App packages searchable is `ANDROID_CUSTOMIZATION_POINT_HYBRIDAPPSEARCH`.

1. Add an XML layout called `emptyview.xml`, and do not add anything to the resulting autogenerated XML file.
2. Open the `hybridapps_list.xml` file for editing and add the following tag above the `ListView` tag:

```
<EditText
    android:hint="@string/SEARCH_HINT"
    android:id="@+id/EditTextSearchHybridAppList"
    android:layout_width="match_parent"
    android:layout_height="47dp" />
```

3. Open `... \Values \Strings.xml` and, between the `<resource>` and `</resource>` tags, add:

```
<string name="SEARCH_HINT">search</string>
```

4. Copy the `UiHybridAppScreen.java` file to your own class name, for example, `SearchableAppScreen.java` and open it for editing.

- a) Add these import statements:

```
import android.widget.EditText;
import android.text.Editable;
import android.text.TextWatcher;
```

- b) Add the following code to the end of the `onCreate` method:

```
final EditText edittext = (EditText)
findViewById(R.id.EditTextSearchHybridAppList);
edittext.addTextChangedListener( new TextWatcher ()
{
    public void afterTextChanged( Editable s )
    {
        String sSearchFor = s.toString();
        m_adapter.setSearch( sSearchFor );
        m_adapter.notifyDataSetChanged();
    }

    // stubs; have to implement the abstract methods
    public void beforeTextChanged( CharSequence s, int start, int
count, int after ) {}
    public void onTextChanged( CharSequence s, int start, int
before, int count) {}
});
```

- c) Add this member variable to the `HybridAppAdapter` class:

```
String m_sToSearchFor;
```

- d) Add this line of code to the end of the HybridAppAdapter constructor method:

```
m_sToSearchFor = "";
```

- e) Replace the code inside the getView method with:

```
public View getView(int position, View convertView, ViewGroup
parent)
{
    LayoutInflater vi =
(LayoutInflater) getSystemService (Context.LAYOUT_INFLATER_SERVI
CE);
    View v = vi.inflate(R.layout.hybridapps, null);

    HybridApp oHybridApp = getItem( position );
    if( oHybridApp != null )
    {
        if( m_abDisplayThisApp == null || position >=
m_abDisplayThisApp.length || m_abDisplayThisApp[position] )
        {
            ImageView ic = (ImageView)
v.findViewById( R.id.hybridApp_icon );

            ic.setImageResource( UiIconIndexLookup.getNormalIconIdForIndex
( oHybridApp.getIconIndex() ));
            TextView tt = (TextView)
v.findViewById(R.id.hybridApp_title);
            if (tt != null)
            {
                tt.setText( oHybridApp.getDisplayName());
            }
        }
        else
        {
            v = vi.inflate(R.layout.emptyview, null);
        }
    }
    return v;
}
```

- f) Add a search method to the HybridAppAdapter class:

```
public void search()
{
    m_abDisplayThisApp = new boolean[m_adapter.getCount()];

    for(int index = 0; index < m_adapter.getCount(); index++)
    {
        int iIndexOfResult =
m_adapter.getItem( index ).getDisplayName().indexOf( m_sToSear
chFor );
        if( iIndexOfResult >= 0 )
        {
            m_abDisplayThisApp[index] = true;
        }
    }
}
```

```
    }  
}
```

- g) Add these methods to the `HybridAppAdapter` class:

```
public void notifyDataSetChanged()  
{  
    search();  
    super.notifyDataSetChanged();  
}  
public void setSearch( String sSearchFor )  
{  
    m_sToSearchFor = sSearchFor;  
}
```

- h) Add this member variable to the `UiHybridAppScreen` class:

```
private boolean[] m_abDisplayThisApp;
```

5. Open the `CustomizationHelper.java` file, which is located in the `... \HybridWebContainer\src\com\sybase\hwc` folder and edit the `getHybridAppScreenClass()` method, to change the class returned to your new class.
(Android only) That class must extend **Activity**.
6. (Android only) Update the `manifest.xml` file to include the new activity you create.

Customizing the Push Notification Handler in the Android Hybrid Web Container

The comment tag associated with this customization is
`ANDROID_CUSTOMIZATION_POINT_PUSH_NOTIFICATION`.

By default, when a push notification is received by the Hybrid Web Container push listener, it returns the `PushNotificationListener.NOTIFICATION_CONTINUE` method, which allows the next push listener to handle the notification.

The comments in the `onPushNotification` method in the `CustomizationHelper.java` file include sample code that demonstrates how to open the default client-initiated Hybrid App if no Hybrid App is currently opened and also, optionally, calls a JavaScript method to initialize the Hybrid App once it is opened.

1. Open the `CustomizationHelper.java` file for editing.
2. Find the `onPushNotification` method and make your changes.
For example, if `PushNotificationListener.CANCEL` is returned, then the push listener manager will not invoke the next push notification listener.
3. Save the file.
4. Rebuild the project.

Testing Android Hybrid Web Containers

After making any customizations to the provided Hybrid Web Container source code, you should test the changes before using the application.

Note: The steps or interface may be different depending on which Android SDK version you are using.

This procedure assumes that you are using Eclipse.

1. Create a new Android virtual device.
 - a) a. Open the Android SDK Manager. If you are using Eclipse choose **Window > AVD Manager**.
 - b) b. Select **Tools > Manage AVDs**.
 - c) Click **New**.
 - d) Enter a name for the device and select **Android 2.2** as the target.
 - e) Click **Create AVD**.
2. Create a debug configuration for Android applications.
 - a) In Eclipse, in WorkSpace Navigator, right-click the Hybrid Web Container project and select **Debug as > Debug Configurations**.
 - b) Right-click **Android Application**.
 - c) Click **Target**.
 - d) In Deployment Target Selection Mode, select **Manual** and click **Debug**.
In the future you will only need to right-click the project and choose **Debug As > Android Application**.
 - e) In the Android Device Chooser, select **Launch a New Android Virtual Device (AVD)** and select the AVD you created in step 1.
 - f) Click **Start**.
 - g) Click **Launch**.

The Hybrid Web Container automatically launches when the emulator is fully started.

Upgrading the PhoneGap Library Used by the Android Hybrid Web Container

SAP Mobile Platform includes the Cordova (PhoneGap) 2.0 libraries. Follow these steps if you want to upgrade the Android Hybrid Web Container to a more recent version of the Cordova library.

This procedure describes upgrading the Cordova library from version 2.0.0 to version 2.9.0. The steps to upgrade to other versions differ slightly. Since the Hybrid Web Container template project does not include the source code for building `HWCLib.jar`, the ability to upgrade Cordova to newer versions is limited, and certain new Cordova features may not work properly in Hybrid Web Container project.

Note: Upgrading the Hybrid Web Container container to use Cordova 3.0.0 is not supported because the Hybrid Web Container project does not work with Cordova 3.0.0 CLI.

1. Download phonegap 2.9.0 from *phonegap.com*, and unzip it to a local folder.
2. Open Eclipse and import the HWC template project.
3. Expand the HWC template project, and delete the `cordova-2.0.0.jar` file from the `libs` folder. Copy the `cordova-2.9.0.jar` file from the unzipped phonegap2.9.0 \lib\android\ folder, and copy it to the `libs` folder.
4. Right click the Hybrid Web Container project and click the **Properties** menu. Select **Java Build Path > Libraries**.
5. Select the **cordova-2.0.0.jar** file, then select **Remove**, to remove the old jar file.
6. Select **Add JARs...** and expand the **HybridWebContainer\libs** node. Select the new **cordova-2.9.0.jar** file, and click **OK** to confirm the selection.
7. Select **OK** to close the “Properties” dialog.
8. Update the private `void initWebView()` method.

`private void initWebView()` calls the `super.loadUrlWithData(sBaseURL, abData)` Cordova method, which no longer exists in cordova-2.9.0. Change this method to call the `super.loadUrl(sBaseURL)` method instead.

Open the `UiHybridAppContainer.java` class and navigate to the private `void initWebView()` method and make this change:

```
// PhoneGap Change: We must call through PhoneGap to load the URL
if ( USE_PHONEGAP )
{
    // PhoneGap may timeout loading the web page
    super.setIntegerProperty( "loadUrlTimeoutValue", 300000 );

    // PhoneGap will load the URL
    super.loadUrl( sBaseURL );
}
else
{
    m_oWebView.loadDataWithBaseURL( sBaseURL, new String( abData ),
    null, "utf-8", null );
}
```

9. Clean the HWC project and have Eclipse build the HWC project.
10. If the `cordova.js` file is used in your `HybridApp.js` app, you must also update `cordova.js` to the one provided with the new cordova library.

BlackBerry Hybrid Web Container Customization

Customize the look and feel and default behavior of the BlackBerry Hybrid Web Container.

Before getting started:

- Install the BlackBerry Java Plug-in for Eclipse. For information about the BlackBerry Java Plug-in for Eclipse, see <https://developer.blackberry.com/java/download/eclipse/>.

Note: If you are also developing for Android, SAP recommends that you do not install the BlackBerry Java Plug-in for Eclipse and the ADT plug-in in the same Eclipse environment.

- Build the Hybrid Web Container project as described in *Building the BlackBerry Hybrid Web Container Using the Provided Source Code*. The `HybridWebContainer` directory contains directories such as `libs`, as well as `images` and other files.

BlackBerry Customization Touch Points

All code areas associated with BlackBerry Hybrid Web Container customizations are annotated with `BLACKBERRY_CUSTOMIZATION_POINT_<customization>` comment tags, or touch points.

Touch Point	Description
<code>BLACKBERRY_CUSTOMIZATION_POINT_COLORS</code>	Use custom colors for the Hybrid Web Container.
<code>BLACKBERRY_CUSTOMIZATION_POINT_FONTS</code>	Use custom fonts in the Hybrid Web Container.
<code>BLACKBERRY_CUSTOMIZATION_POINT_BRAND</code>	Change application name, copyright, and developer information.
<code>BLACKBERRY_CUSTOMIZATION_POINT_SPLASHSCREEN</code>	Add a splash screen to the Hybrid Web Container.
<code>BLACKBERRY_CUSTOMIZATION_POINT_DEFAULTSETTINGS</code>	Set the defaults for the Settings screen.
<code>BLACKBERRY_CUSTOMIZATION_POINT_PRESETSETTINGS</code>	Hard-code Settings screen options so they do not show up on the device, preventing the user from changing the settings.
<code>BLACKBERRY_CUSTOMIZATION_POINT_PIN</code>	Use for PIN screen customizations, or to remove the PIN screen.
<code>BLACKBERRY_CUSTOMIZATION_POINT_SORTING</code>	Sort application messages based on a variety of criteria.
<code>BLACKBERRY_CUSTOMIZATION_POINT_FILTERING</code>	Filter the message list so only messages meeting certain criteria are shown.

Touch Point	Description
BLACKBERRY_CUSTOMIZATION_POINT_HYBRIDAPPSORT	Customize the criteria for sorting the Hybrid App list.
BLACKBERRY_CUSTOMIZATION_POINT_HYBRIDAPPSEARCH	Make the list of Hybrid App packages searchable.
BLACKBERRY_CUSTOMIZATION_POINT_HYBRIDAPPLIST	Customize the Hybrid App package list appearance.
BLACKBERRY_CUSTOMIZATION_POINT_CATEGORIZED-VIEWS	Create categorized views of the Hybrid App packages.
BLACKBERRY_CUSTOMIZATION_POINT_HTTPHEADERS	Set HTTP headers for the BlackBerry Hybrid Web Container to include authentication tokens.
BLACKBERRY_CUSTOMIZATION_POINT_MULTIHWC	Install more than one Hybrid Web Container on one device.
BLACKBERRY_CUSTOMIZATION_POINT_PREPACKAGE_APP	Run the Hybrid Web Container as a single Hybrid App.
BLACKBERRY_CUSTOMIZATION_POINT_PUSH_NOTIFICATION	Customize the way the Hybrid Web Container handles push notifications.
BLACKBERRY_CUSTOMIZATION_POINT_ANONYMOUS_USER	<p>Returns whether or not anonymous user login is supported. Change to YES to allow clients to register anonymously.</p> <hr/> <p>Note: For this to work, the HWC application connection template must be configured to use the anonymous security configuration. See <i>Application Connection Templates</i> in <i>SAP Control Center for SAP Mobile Platform</i>.</p>

Look and Feel Customization of the BlackBerry Hybrid Web Container

Customizations you can make to the look and feel include changing the splash screen, changing the Hybrid App icons and name, changing the Hybrid App package icons, changing labels and text, adding support for new languages, and so on.

Changing the BlackBerry Hybrid Web Container Icon

Replace the BlackBerry Hybrid Web Container icon image file.

1. Navigate to the `HybridWebContainer\res\images` folder.
2. Replace the `icon.png` file with another `.png` image of your choosing.
The new image must use the same name, resolution, and extension as the original file.
3. Rebuild the project.

Rebranding the BlackBerry Hybrid Web Container

Modify the strings used in the `Brand` class for the BlackBerry Hybrid Web Container.

Almost all company and product specific strings used in the Hybrid Web Container are accessed through the `Brand` class.

1. Open the `HybridWebContainer.java` file for editing.
2. Make your modifications at the beginning of the main method (if you do not want to modify a default value, simply omit the line that changes it):

```
Brand.OEM_COMPANY_NAME = "Your Company Name";
Brand.OEM_FORMAL_COMPANY_NAME = "Your Formal
Company Name";
Brand.OEM_ROBIE_PRODUCT = "Your Name of the
Product";
Brand.OEM_COPYRIGHT = "Your Copyright String";
Brand.OEM_CORPDIR_OB_NAME = "HybridAppList Title";
```

3. Save the file.
4. To change the title, which uses the string `HybridWebContainer`, that appears on the Hybrid Web Container settings Screen:
 - a) In the Package Explorer view, right-click the BlackBerry application project and click **Properties**.
 - b) In the Properties for pane, click **BlackBerry Project**.
 - c) Click **Application Descriptor**.
 - d) Click the **Application** tab and change the Title.
 - e) In Package Explorer, right-click the `BlackBerry_App_Descriptor.xml` file and choose **Open With > Text Editor**.
 - f) Find the tag named `Packaging` and change the value of the `OutputFileName` to the name you used in step 4d.

Note: Remove any spaces or dashes, since these are illegal characters for output files.

- g) Open the `HybridWebContainer.java` file for editing.
- h) Add this line at the beginning of the `postEvent` method:

```
Brand.OEM_ENGINE_EXE_NAME = "HybridWebContainer";
```

Replace `HybridWebContainer` with the name you used in step 4d.

Note: If you modify `Brand.OEM_HYBRIDAPP_APPID`, you must have a matching **Application ID** in SAP Control Center.

Adding a Splash Screen

Add a splash screen to the BlackBerry Hybrid Web Container.

The splash screen is the first screen you see in the Hybrid Web Container. The related comment tag is `BLACKBERRY_CUSTOMIZATION_POINT_SPLASHSCREEN`.

1. Open the `CustomizationHelper.java` file for editing.
2. Find the `getSplashScreenClass` method.
3. Write your own splash screen class.
4. Have `getSplashScreenClass` return the class that you wrote for your splash screen, for example:

```
return SplashScreen.class;
```

Your class must extend `MainScreen`, call `pushScreen` on itself so that it appears, then `popScreen` on itself when it is finished.

```
package com.sybase.hwc;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import java.util.*;
/**
 * A simple splash screen.
 */
public class SplashScreen extends MainScreen
{
    private Timer timer = new Timer();

    public SplashScreen()
    {
        setTitle("Splash Screen");
        add( new LabelField( "Splash" ) );
        addKeyListener( new SplashScreenListener( this ) );

        // Dismiss the splash screen after 5 seconds.
        timer.schedule( new Countdown(), 5000 );

        UiApplication.getUiApplication().pushScreen( this );

        UiApplication.getUiApplication().requestForeground();
    }

    public void dismiss()
    {
        timer.cancel();
        UiApplication.getUiApplication().popScreen( this );
    }

    private class Countdown extends TimerTask
```

```

    {
        public
            void run()
            {

                UiApplication.getUiApplication().invokeLater( new
DismissThread() );
            }
        }

private class DismissThread implements Runnable
{
    public void run() {
        dismiss();
    }
}

protected boolean navigationClick( int status, int time )
{
    dismiss();
    return true;
}

protected boolean navigationUnclick( int status, int time )
{
    return false;
}

protected boolean navigationMovement( int dx, int dy, int
status, int time )
{
    return false;
}

private static class SplashScreenListener implements
KeyListener
{
    private SplashScreen screen;

    public SplashScreenListener( SplashScreen splash )
    {
        screen = splash;
    }

    public boolean keyChar( char key, int status,
int time )
    {
        // Quit the splash screen if ESC or MENU
key pressed.
        switch ( key )
        {
            case
Characters.CONTROL_MENU:
            case Characters.ESCAPE:
                screen.dismiss();
                return true;
        }
    }
}

```

```
        }
        return false;
    }

    public boolean keyDown( int keycode, int time )
    {
        return false;
    }

    public boolean keyRepeat( int keycode, int time )
    {
        return false;
    }

    public boolean keyStatus( int keycode, int time )
    {
        return false;
    }

    public boolean keyUp( int keycode, int time )
    {
        return false;
    }
}
}
```

5. Save the file and rebuild the project.

Changing Labels and Text in the BlackBerry Hybrid Web Container

You can customize most of the text found in labels, dialogs, and error messages used by the Hybrid Web Container.

All of the text that is not branding related and that appears as part of the Hybrid Web Container is contained in the `HybridWebContainer.rrc` file.

1. Open the `HybridWebContainer\res\com\sybase\hwc\HybridWebContainer_<language>.rrc` file, where *<language>* is the language code.

This file contains the text for error messages, screen titles, screen labels, validation messages, and so on.

2. Make your changes and save the file.

Keep in mind that you must also make the same changes for each language you want to translate into.

Adding a New Language

Add support for a new language to the BlackBerry Hybrid Web Container.

The default language for the Hybrid Web Container is English, and the English strings are located in `HybridWebContainer\res\com\sybase\hwc\HybridWebContainer.rrc`. The strings for different languages are located in the

resources folder. In general, strings of a language are located in a file named `HybridWebContainer_<language_code>.rrc`. For example, the German resource file is named `HybridWebContainer_de.rrc`.

1. Right-click the **resources** folder and choose **Create new file in resources**.
2. Name the file `HybridWebContainer_<language_code>.rrc`, where `<language_code>` is the language code of the language you want to add.
3. Double-click the new file to open it.
4. Set all the values to be in the new language.
5. Save the file and rebuild the project.

When the Hybrid Web Container is built with the resource file you added, it automatically uses the values it contains when the language on the BlackBerry device is set to the matching language.

Customizing the About Screen for the BlackBerry Hybrid Web Container

The related comment tag for customizing the About screen is

```
BLACKBERRY_CUSTOMIZATION_POINT_BRAND.
```

1. Open the `CustomizationHelper.java` file for editing.
2. Find the `customAbout` method, which contains commented-out code in the `customAbout` method, and Replace the text with whatever values you require.
3. Save the file and rebuild the project.

Using Custom Colors

The comment tag for customizing colors is

```
BLACKBERRY_CUSTOMIZATION_POINT_COLORS.
```

There are a few places where you can change colors.

These steps provide an example of how to change the colors of different Hybrid Web Container components.

1. To change the highlight color of the selected Hybrid App in the Hybrid App list:
 - a) Open the `AppScreen.java` file for editing.
 - b) Make these modifications to the `drawListRow` method, found in the `ListFieldCallback` (the changes are in **bold**).

The changes in this example make the highlighted color orange and the unhighlighted color black (by default, they are blue and white, respectively).

```
public void drawListRow(ListField listField, Graphics graphics,
int index, int y, int width) {
// y parameter is already offset to center text
int iOffset = (listField.getRowHeight() -
getFont().getHeight()) >> 1;

HybridApp oApp = ( HybridApp ) m_oApps.elementAt( index );
```

```

if( listField.getSelectedIndex() == index )
{
graphics.setColor( Color.ORANGE );
}
else
{
graphics.setColor( Color.BLACK );
}

graphics.fillRect( 0, y - iOffset, width,
listField.getRowHeight() + y - iOffset );

final int iMargin = 2;

// Draw image

EncodedImage oImage
= EncodedImage.getEncodedImageResource( "ampicon" +
oApp.getIconIndex() + ".png");
Bitmap oBitmap = oImage.getBitmap();

graphics.drawBitmap( iMargin, y - iOffset +
( listField.getRowHeight() -oBitmap.getHeight() ) / 2,
oBitmap.getWidth(), oBitmap.getHeight(), oBitmap, 0, 0);

// Draw text
graphics.drawText( oApp.getDisplayName(), 2 * iMargin +
oBitmap.getWidth(), y );
}

```

2. To change the text color of the Hybrid App names in the Hybrid App list:

- a) In the AppScreen.java file, go to the drawListRow method, which is in the ListFieldCallback.

The color of the text is set by the code below. The first color (white, by default) is used when the field is in focus. The second color is used when the field is not in focus. This example coordinates these colors with the colors used in step 1. The changed code is in **bold**.

- b) Modify the code. For example:

```

// Draw text
if( listField.getSelectedIndex() == index )
{
graphics.setColor( Color.BLACK );
}
else
{
graphics.setColor( Color.WHITE );
}

```



```
graphics.drawText( oApp.getDisplayName(), 2 * iMargin +
oBitmap.getWidth(), y );
```

3. To change the background color of the Hybrid Web Container:

- a) Add these import statements to the `AppScreen.java` file:

```
import net.rim.device.api.ui.decor.Background;
import net.rim.device.api.ui.decor.BackgroundFactory;
```

- b) In the `AppScreen.java` file, go to the constructor method and add these lines after the `setTitle(...);` line:

```
Background bg =
BackgroundFactory.createSolidBackground( Color.BLACK );
this.getMainManager().setBackground( bg );
```

4. Change the background color and text color of label and edit fields.

To change the background and text colors of a label or edit field, you must override its paint method. This is done when you create the label. Below is an example of how to set the background color to black and the text color to white for a label. You can also do this, similarly, for edit fields.

- a) Open the `HWCSettingsScreen.java` file for editing.
- b) Make the following modifications (changes in bold). These changes make the background of the label black, and the text white. To use the same background color as the rest of the screen, you can leave out the first two lines in the paint method below:

```
// Connection Header
m_oConnection = null;
m_oConnection = new
LabelField( m_res.getString( HybridWebContainerResource.IDS_CO
NNECTION ),

Field.FIELD_HCENTER )
{
    public void paint(Graphics g){
        g.setColor( Color.BLACK );
        g.fillRect( 0, 0, getWidth(), getHeight() );
        g.setColor( Color.WHITE );
        super.paint( g );
    }
};
```

5. Save the file and rebuild the project.

Using Custom Fonts

The customization tag for customizing fonts is `BLACKBERRY_CUSTOMIZATION_POINT_FONTS`.

Use custom `.ttf` font files, which have a maximum size of 60KB, to install and use a custom font. You can set the default font for the Hybrid Web Container (described in step 1), or change the fonts for individual labels (described in step 2). Fonts for the list of Hybrid Apps are a special case (described in step 3).

1. Set the default font for the Hybrid Web Container:

- a) Add the .ttf font file to the resources folder of the HybridWebContainer project.
- b) Open the HWCSettingsScreen.java file and navigate to the constructor method, and add the following code to the beginning of that method.

The value `FELIXTI.TTF` in the second line is used. This is the name of the font file, and you should replace this value with the name of the font file you added in step 1a.

```
String sCustomFontName = "MyCustomFont";
int iFontLoadCode =
FontManager.getInstance().load( "FELIXTI.TTF",
sCustomFontName,

FontManager.APPLICATION_FONT);
if( iFontLoadCode == FontManager.SUCCESS)
{
    try
    {
        FontFamily oFamily =
FontFamily.forName( sCustomFontName );
        Font oFont = oFamily.getFont( Font.PLAIN, 23 );
        FontManager.getInstance().setApplicationFont( oFont );
    }
    catch (ClassNotFoundException e)
    {
        // the font was not found, so it cannot be set
    }
}
else
{
    // error loading font
}
```

The default font is applied to menu items, but not to the menu item that has focus. The following steps correct this.

- c) Open the AppScreen.java file and add:

```
import net.rim.device.api.ui.Font;
import net.rim.device.api.ui.FontFamily;
```

- d) Add this code to the end of the makeMenu method:

```
try
{
    FontFamily oFamily =
FontFamily.forName( "MyCustomFont" );
    Font oFont = oFamily.getFont( Font.PLAIN, 23 );
    menu.setFont( oFont );
}
catch ( ClassNotFoundException e )
{
    // problem finding the custom font
    String errorMsg = e.getMessage();
}
```

- e) Open the `LogScreen.java` file and add:

```
import net.rim.device.api.ui.FontFamily;
import net.rim.device.api.ui.component.Menu;
```

- f) Add the following method to both the `LogScreen` class (in `LogScreen.java`) and to the `HWCSettingsScreen` class (in `HWCSettingsScreen.java`):

```
protected void makeMenu( Menu menu, int context )
{
    try
    {
        FontFamily oFamily =
FontFamily.forName( "MyCustomFont" );
        Font oFont = oFamily.getFont( Font.PLAIN, 23 );
        menu.setFont( oFont );
    }
    catch ( ClassNotFoundException e )
    {
        String errorMsg = e.getMessage();
        System.out.println( errorMsg );
    }
    super.makeMenu( menu, context );
}
```

- g) In the `HWCSettingsScreen.java` file, add:

```
import net.rim.device.api.ui.FontFamily;
import net.rim.device.api.ui.Font;
import net.rim.device.api.ui.component.Menu;
```

2. Set the font for an individual label:

This example shows how to change the font for the screen title. Changing the font for any label is similar.

- a) Add the font file (a `.ttf` file) to the `resources` folder of the `HybridWebContainer` project.
- b) To the `AppScreen.java` file, add:

```
import net.rim.device.api.ui.Font;
import net.rim.device.api.ui.FontFamily;
```

- c) If you are going to set the font on more than one label, have a helper method. Add the following method to the `AppScreen` class:

```
public void setCustomFont( LabelField oLabel, String
sCustomFontName, int iSize )
{
    try
    {
        FontFamily oFamily =
FontFamily.forName( sCustomFontName );
        Font oFont = oFamily.getFont( Font.PLAIN, iSize );
        oLabel.setFont( oFont );
    }
    catch ( ClassNotFoundException e )
    {
        // the font was not found, so it cannot be set
        System.out.println( "Exception: font not found!" );
    }
}
```

```
    }  
}
```

- d) In the `AppScreen` constructor, replace the `setTitle(...)` line with the code below.

"`SHOWG.TTF`" is the name of the font file. Replace this with the name of the font file you added in step 2a.

```
        LabelField oTitleLabel = new LabelField( Consts.APP_TITLE,  
        DrawStyle.ELLIPSIS );  
        FontManager.getInstance().load( "SHOWG.TTF",  
        "CustomTitleFont", FontManager.APPLICATION_FONT);  
        setCustomFont( oTitleLabel, "CustomTitleFont", 23 );  
        this.setTitle( oTitleLabel );
```

3. To change the font for the names of the Hybrid Apps in the list of Hybrid Apps:

- Add the font file (a `.ttf` file) to the `resources` folder of the `HybridWebContainer` project.
- Open the `AppScreen.java` file for editing.
- Navigate to the `drawListRow` in `ListFieldCallback` and make the changes below, shown in bold.

"`HARLOWSI.TTF`" is the name of the font file. Replace this with the name of the font file you added in step 3a.

```
// Draw text  
FontManager.getInstance().load( "HARLOWSI.TTF",  
"CustomHybridAppFont", FontManager.APPLICATION_FONT);  
try  
{  
    FontFamily oFamily =  
FontFamily.forName( "CustomHybridAppFont" );  
    Font oFont = oFamily.getFont( Font.PLAIN,  
23 );  
    graphics.setFont( oFont );  
    graphics.drawText( oApp.getDisplayNa  
me(), 2 * iMargin + iBitmap.getWidth(), y );  
}  
catch ( ClassNotFoundException e )  
{  
    //can't load the font  
}
```

Default Behavior Customization for the BlackBerry Hybrid Web Container

Remove a PIN screen, configure default values for the Settings screen, customize the About screen, sort Hybrid App messages, and so on.

Removing Fields from the Settings Screen

Hard-code the Settings screen so options do not appear on the Settings screen on the BlackBerry device.

The comment tag associated with the fields on the Settings screen is `BLACKBERRY_CUSTOMIZATION_POINT_DEFAULTSETTINGS`.

1. Open the `CustomizationHelper.java` file, which is located in the ...
`\HybridWebContainer\src\com\sybase\hwc` folder.
2. Search for the method named with the pattern `isConnection***Visible`, where `***` is the name of the connection setting field.

By default, each method returns true. To remove a field from the screen, change the appropriate method to return false.

3. Save the file.
4. Rebuild the project.

Configuring Default Values for the Settings Screen

All customization functionality for the Settings screen is grouped together in the `CustomizationHelper.java` file. The associated comment tag is `BLACKBERRY_CUSTOMIZATION_POINT_DEFAULTSETTINGS`.

1. Open the `CustomizationHelper.java` file for editing.
2. Search for the methods named with this pattern:
 - `getDefaultConnection***`
 - `isDefaultConnect***`
 where `***` is the name of the setting.
3. Edit the methods to return the value you specify.
4. Save the file.
5. Rebuild the project.

Using Multiple Hybrid Web Containers on the Same BlackBerry Device

Configure the Hybrid Web Container so that two or more Hybrid Web Containers can coexist on the same BlackBerry device.

Use a different COD module name, and make other changes to your new Hybrid Web Container, such as for the icon .png image, to differentiate between the Hybrid Web Containers on the device.

1. Double-click on the file `BlackBerry_App_Descriptor.xml` to open it.
2. In the **Application** tab, change the title of the Hybrid Web Container.

3. In the **Build** tab, change the output file name to the name you used in step 2, but remove any spaces or dashes, since these are illegal characters for output files.
4. Open the `CustomizationHelper.java` file for editing.
5. Find the method named `getAppId()` and replace `Brand.OEM_HYBRIDAPP_APPID` with a unique name for your application.

The user must be registered in SAP Control Center with a device ID that matches the value you use in this step. You may need to create the device ID in SAP Control Center.

6. Open the `CustomizationHelper.java` file for editing.
7. Change the return value of `getApplicationIndicatorIconName` to the new indicator icon name, for example:

```
public class CustomizationHelper
{
    ....
    public final String getApplicationIndicatorIconName()
    { //return HWCMessagesScreen.INDICATOR_PNG; return "icon.png"; }
}
```

Sorting the List of Hybrid Apps

By default, Hybrid Apps are sorted alphabetically, ignoring case. The customization tag associated with sorting the list of Hybrid Apps is

`BLACKBERRY_CUSTOMIZATION_POINT_HYBRIDAPPSORT`.

1. Open the `CustomizationHelper.java` file for editing.
2. Search for the method named `getHybridAppComparator()` and modify the code to suit your sorting requirements.

This example shows the Hybrid App being sorted by display name in reverse alphabetical order:

```
public Comparator getHybridAppComparator() {
return new Comparator() {
public int compare(Object oApp1, Object oApp2) {
String sDisplayName1 = ((HybridApp) oApp1).getDisplayName()
.toLowerCase();
String sDisplayName2 = ((HybridApp) oApp2).getDisplayName()
.toLowerCase();
return (-1)*sDisplayName1.compareTo(sDisplayName2);
}
};
}
```

3. Save the file.
4. Rebuild the project.

Sorting Hybrid AppMessages

The default sorting behavior for Hybrid App messages is to list messages in the order they are received, newest first. The customization tag for sorting messages is `BLACKBERRY_CUSTOMIZATION_POINT_SORTING`.

1. Open the `CustomizationHelper.java` file for editing.
2. Search for the method named `getMessageComparator()` and modify the code to your sorting requirements.
3. Save the file.
4. Rebuild the project.

Filtering Hybrid App Messages

Filter the list of Hybrid App messages so only messages that meet specified criteria are shown. The default behavior is to return all messages. The comment tag associated with filtering Hybrid App messages is `BLACKBERRY_CUSTOMIZATION_POINT_FILTERING`.

1. Open the `CustomizationHelper.java` file for editing.
2. Find the method named `getFilteredMessages()` and modify it to meet your criteria.
`getFilteredMessages()` includes commented-out sample code that demonstrates how to filter out low-importance messages.
3. Save the file.
4. Rebuild the project.

Setting HTTP Headers

Set HTTP headers for the BlackBerry Hybrid Web Container to include authentication tokens.

These sample methods show how to do this in the BlackBerry Hybrid Web Container template source code.

- `setHttpHeaders()` – use this method to set the authentication tokens. The tokens you set are used until `setHttpHeaders` is called again.
- `setWorkflowTokenErrorListener()` – use this method to call `setHttpHeaders()` to put the authentication tokens back in a good state, if, for example, they have expired.
- `setHttpErrorListener()` – use this method to handle HTTP errors.

The comment tag associated with setting HTTP headers is `BLACKBERRY_CUSTOMIZATION_POINT_HTTPHEADERS`.

1. Open the `CustomizationHelper.java` file and make your changes.

2. Save the file.
3. Rebuild the project.

Modifying the Hybrid App List Appearance

The comment tag associated with customizing the Hybrid App list appearance is `BLACKBERRY_CUSTOMIZATION_POINT_HYBRIDAPPLIST`.

To show the list of Hybrid Apps, the Hybrid Web Container calls the `getHybridAppScreenClass()` method in the `CustomizationHelper.java` file. `getHybridAppScreenClass()` returns the default class `AppScreen` that displays the list.

1. To make small changes edit `AppScreen`, or create your own class that extends `UiHybridAppScreen`.
2. If you write your own class to extend `UiHybridAppScreen`, update `getHybridAppScreenClass` to return the name of your new class.
3. Save the file.
4. Rebuild the project.

Creating a Tree View

Modify the BlackBerry Hybrid Web Container so that Hybrid Apps appear in a tree view.

1. In the BlackBerry HybridWebContainer template project, in the `src` folder, right-click the `com.sybase.hwc.amp` package and choose **New > File**.
2. Enter `TreeViewAppScreen.java` for the file name, and click **Finish**.
3. Open the `TreeViewAppScreen.java` file for editing, and paste this code into the file.

```
/*
 Copyright (c) SAP, Inc. 2012 All rights reserved.

 In addition to the license terms set out in the SAP License
 Agreement for
 the SAP Mobile Platform ("Program"), the following additional or
 different
 rights and accompanying obligations and restrictions shall apply
 to the source
 code in this file ("Code"). SAP grants you a limited, non-
 exclusive,
 non-transferable, revocable license to use, reproduce, and modify
 the Code
 solely for purposes of (i) maintaining the Code as reference
 material to better
 understand the operation of the Program, and (ii) development and
 testing of
 applications created in connection with your licensed use of the
 Program.
 The Code may not be transferred, sold, assigned, sublicensed or
 otherwise
```



```

conveyed (whether by operation of law or otherwise) to another
party without
SAP's prior written consent. The following provisions shall apply
to any
modifications you make to the Code: (i) SAP will not provide any
maintenance
or support for modified Code or problems that result from use of
modified Code;
(ii) SAP expressly disclaims any warranties and conditions,
express or
implied, relating to modified Code or any problems that result
from use of the
modified Code; (iii) SAP SHALL NOT BE LIABLE FOR ANY LOSS OR
DAMAGE RELATING
TO MODIFICATIONS MADE TO THE CODE OR FOR ANY DAMAGES RESULTING
FROM USE OF THE
MODIFIED CODE, INCLUDING, WITHOUT LIMITATION, ANY INACCURACY OF
DATA, LOSS OF
PROFITS OR DIRECT, INDIRECT, SPECIAL, INCIDENTAL OR CONSEQUENTIAL
DAMAGES, EVEN
IF SAP HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES; (iv)
you agree
to indemnify, hold harmless, and defend SAP from and against any
claims or
lawsuits, including attorney's fees, that arise from or are
related to the
modified Code or from use of the modified Code.
*/
package com.sybase.hwc.amp;

import com.sybase.mo.*;
import com.sybase.hybridApp.*;

import java.util.Enumeration;

import net.rim.device.api.i18n.ResourceBundle;
import net.rim.device.api.system.*;
import net.rim.device.api.ui.*;
import net.rim.device.api.ui.component.*;
import net.rim.device.api.ui.container.*;
import net.rim.device.api.util.SimpleSortingVector;
import com.sybase.hwc.*;

// BLACKBERRY_CUSTOMIZATION_POINT_AUTOSTART
// BLACKBERRY_CUSTOMIZATION_POINT_COLORS
// BLACKBERRY_CUSTOMIZATION_POINT_FONTS
// BLACKBERRY_CUSTOMIZATION_POINT_HYBRIDAPPLIST

/**
 * This class displays a list of user invokable widgets currently
 * present on the
 * device.
 */
public class TreeViewAppScreen extends MainScreen {

    // Create a ResourceBundle object to contain the localized

```

Hybrid Web Container Customization

```
resources.  
    // Here is a little bit of MAGIC. How do you know what there is  
    // a class HybridWebContainerResource? (hint: its not from the docs)  
    // It is auto generated by the JDE. Convention is  
    AppNameResource.BUNDLE_ID, AppNameResource.BUNDLE_NAME  
    // http://www.codeproject.com/KB/mobile/EndToEndBBApp5.aspx  
    public static final ResourceBundle RESOURCE =  
        ResourceBundle.getBundle(  
            HybridWebContainerResource.BUNDLE_ID,  
            HybridWebContainerResource.BUNDLE_NAME );  
  
    public TreeViewAppScreen() {  
        super(Manager.VERTICAL_SCROLL |  
            Manager.NO_HORIZONTAL_SCROLLBAR);  
  
        setTitle( RESOURCE.getString( HybridWebContainerResource.IDS_HYBR  
            IDAPPS ) );  
  
        // Sort apps by their display name  
        m_oApps = new SimpleSortingVector();  
  
        m_oApps.setSortComparator( CustomizationHelper.getInstance().getH  
            ybridAppComparator() );  
  
        m_oApps.setSort(false);  
  
        // Populate and sort list  
        BBHybridAppHelper.addAppStoreListener( m_oAppListener );  
        // Add list field to screen  
        m_oTreeField = new TreeField( m_oTreeFieldCallback,  
            TreeField.FOCUSABLE );  
  
        m_oTreeField.setEmptyString( BBHybridWebContainer.getMocaStringRe  
            source( MocaClientLibResource.LBL_NO_WIDGETS_FOUND ),  
            DrawStyle.HCENTER );  
        // set the size of the indentation  
        m_oTreeField.setIndentWidth( 30 );  
        populateList();  
        updateScreen();  
  
        // add the tree field to the screen  
        add( m_oTreeField );  
    }  
  
    /**  
     * Handle clicking on an application  
     */  
    protected boolean navigationClick(int status, int time)  
    {  
        Field oField = getFieldWithFocus();  
        // only handle if it was the tree field that was clicked  
        if ( oField instanceof TreeField )  
        {  

```

```

        Object obj = m_oTreeField.getCookie( ( ( TreeField )
oField ).getCurrentNode() );
        // only handle the click if it was a hybrid app (not a tree
label)
        if( obj instanceof HybridApp )
        {
            // launch the clicked hybrid app
            HybridApp oApp = ( HybridApp ) obj;
            XmlHybridApp.startHybridApp( oApp.getModuleId(),
oApp.getVersion(), false );
            return true;
        }
        return super.navigationClick(status, time);
    }

/**
 * Override the default Screen.close method
 */
public void close()
{
    BBHybridAppHelper.removeAppStoreListener( m_oAppListener );

    UiApplication oApp = UiApplication.getUiApplication();
    oApp.popScreen(this);

    if ( oApp.getScreenCount() == 0 )
    {
        oApp.requestBackground();
    }
}

protected void makeMenu( Menu menu, int instance )
{
    menu.deleteAll();

    if ( CustomizationHelper.getInstance().enableSettings() )
    {
        menu.add(m_mniSettings);
    }

    menu.add(MenuItem.getPrefab(MenuItem.CLOSE));
}

/**
 * Fills in list of apps
 */
private void populateList()
{
    m_oApps.removeAllElements();

    for ( Enumeration e =
BBHybridAppHelper.getClientHybridApps().elements();
e.hasMoreElements(); )
    {

```

```

        HybridApp oHybridApp = ( HybridApp )e.nextElement();
        m_oApps.addElement( oHybridApp );
    }
    m_oApps.reSort();
}

/**
 * Updates the screen
 */
private void updateScreen()
{
    // have to do stuff to the UI on a separate thread
    UiApplication.getUiApplication().invokeLater(
        new Runnable()
        {
            public void run()
            {
                m_oTreeField.deleteAll();
                // if there're no hybrid apps then we do not even
                want to add the tree labels
                // so that the empty string will be displayed
                if( m_oApps.size() > 0 )
                {
                    // In this example, there are 3 top level
                    categories of hybrid apps: Forms, Expense, and Miscellaneous.
                    // Forms has a sub-category of SpecialForms. In
                    practice you can have as many or as few categories
                    // and sub-categories as you like. Here the
                    category of a hybrid app is determined by whether
                    // keywords exist in the display name of that
                    hybrid app, but you could use anything else (for example
                    // you could determine the category of a hybrid
                    app by its icon).
                    int iMiscel = m_oTreeField.addChildNode( 0,
                    "Miscellaneous Hybrid Apps");
                    int iForms = m_oTreeField.addChildNode( 0, "Form
                    Hybrid Apps");
                    int iSpecialForms =
                    m_oTreeField.addChildNode( iForms, "Special Forms");
                    int iExpense = m_oTreeField.addChildNode( 0,
                    "Expense Hybrid Apps");
                    //have to iterate backwards through m_oApps
                    since addChildNode adds the new node
                    //to the first position (appears above the nodes
                    previously added).
                    for( int index = m_oApps.size()-1; index >= 0;
                    index-- )
                    {
                        HybridApp oHybridApp = (HybridApp)
                        m_oApps.elementAt( index );
                        int iParent = iMiscel;
                        if( oHybridApp.getDisplayName().indexOf("Expense") >= 0 )
                        {
                            iParent = iExpense;
                        }
                    }
                }
            }
        }
    );
}

```

```

        else
if( oHybridApp.getDisplayName().indexOf("Form") >= 0 )
    {
if( oHybridApp.getDisplayName().indexOf("Special") >= 0 )
    {
        iParent = iSpecialForms;
    }
    else
    {
        iParent = iForms;
    }
    }
    m_oTreeField.addChildNode( iParent,
m_oApps.elementAt( index ) );
    }
    }
    } );
}

// Settings menu item
private MenuItem m_mniSettings =
    new
MenuItem( m_res.getString(HybridWebContainerResource.IDS_SETTINGS
),
        100001,
        10)
{
    public void run()
    {
        XmlHybridApp.startHybridAppSettings(false);
    }
};

// Listener for app changes
private HybridAppsListener m_oAppListener =
    new HybridAppsListener()
{
    public void onRefreshRequired()
    {
        populateList();
        updateScreen();
    }

    public void onHybridAppAdded(HybridApp oHybridApp)
    {
        populateList();
        updateScreen();
    }

    public void onHybridAppRemoved(HybridApp oHybridApp)
    {
        populateList();
        updateScreen();
    }
}

```

```
    }

    public void onHybridAppUpdated(HybridApp oHybridApp)
    {
        populateList();
        updateScreen();
    }
};

private SimpleSortingVector m_oApps;

private TreeField m_oTreeField;

private static ResourceBundle m_res =
ResourceBundle.getBundle(
    HybridWebContainerResource.BUNDLE_ID,
    HybridWebContainerResource.BUNDLE_NAME );

private TreeFieldCallback m_oTreeFieldCallback = new
TreeFieldCallback()
{
    public void drawTreeItem( TreeField oTree, Graphics
oGraphics, int iNode, int iY, int iWidth, int iIndent )
    {
        Object obj = oTree.getCookie( iNode );
        if( obj instanceof String )
        {
            oGraphics.setColor( Color.BLACK );
            oGraphics.drawText( (String)obj, iIndent, iY);
        }
        else if( obj instanceof HybridApp )
        {
            // y parameter is already offset to center text
            int iOffset = (oTree.getRowHeight() -
getFont().getHeight()) >> 1;

            // Draw a background color for the hybrid apps to
distinguish them from the tree labels.
            // However, if this node has focus we don't want to draw
the grey rectangle because it
            // will cover up the blue color indicating the node is
selected.
            if( iNode != m_oTreeField.getCurrentNode() )
            {
                oGraphics.setColor( Color.LIGHTGRAY );
                oGraphics.fillRect( iIndent, iY - iOffset, iWidth,
m_oTreeField.getRowHeight() );
            }

            HybridApp oApp = ( HybridApp ) obj;

            final int iMargin = 2;

            // Draw image
            EncodedImage oImage =
```

```

EncodedImage.getEncodedImageResource( "ampicon" +
oApp.getIconIndex() + ".png" );
    int iBitmapWidth = 0;

    if ( oImage != null )
    {
        CustomIcon oIcon = oApp.getDefaultCustomIcon();

        if ( oIcon != null )
        {
            EncodedImage oImageTmp =
oApp.getCustomIconImage( oIcon );

            if ( oImageTmp != null )
            {
                if ( oImageTmp.getHeight() != oImage.getHeight()
|| oImageTmp.getWidth() != oImage.getWidth() )
                {
                    MocaLog.getAmpHostLog().logMessage(
                        "Icon image size doesn't match the
built-in icon size, the layout result could be different.",
                        MocaLog.eMocaLogLevel.Normal );
                }

                oImage = oImageTmp;
            }
        }

        Bitmap oBitmap = oImage.getBitmap();
        int iRowHeight = oTree.getRowHeight();

        int iSize = oImage.getHeight() > oImage.getWidth() ?
oImage.getHeight() : oImage.getWidth();

        if ( iSize >= iRowHeight )
        {
            oBitmap =
HWCMessagesListField.getScaledBitmapImage( oImage, iRowHeight -
iMargin, iSize );
        }

        oGraphics.drawBitmap(
            iMargin + iIndent,
            iY - iOffset + ( oTree.getRowHeight() -
oBitmap.getHeight() ) / 2,
            oBitmap.getWidth(), oBitmap.getHeight(),
oBitmap, 0, 0 );

        iBitmapWidth = oBitmap.getWidth();
    }
    else
    {
        MocaLog.getAmpHostLog().logMessage( "Can not find
application icon image of application " +
oApp.getDisplayName() + ".",

```

Hybrid Web Container Customization

```
MocaLog.eMocaLogLevel.Normal );
    }

    // Draw text
    oGraphics.setColor( Color.BLACK );
    oGraphics.drawText( oApp.getDisplayName(), 2 * iMargin
+ iBitmapWidth + iIndent, iY );
    }
}
};
}
```

This file is based on the `AppScreen.java` file. The main differences are in the constructor, `navigationClick`, `populateList`, and `updateScreen` functions. Also, the `TreeFieldCallback` class replaces the `ListFieldCallback` class from `AppScreen.java`.

4. Open the `CustomizationHelper.java` file for editing, find the `getHybridAppScreenClass` function, and replace the existing return statement with this line:

```
return com.sybase.hwc.amp.TreeViewAppScreen.class;
```
5. Save the `CustomizationHelper.java` file.
6. Rebuild the `HybridWebContainer` project.
When you run the Hybrid Web Container, the Hybrid Apps are shown in a tree field.

Creating Categorized Views

Create a set of categories for the list of Hybrid Apps. The comment tag associated with this customization is `BLACKBERRY_CUSTOMIZATION_POINT_CATEGORIZEDVIEWS`.

First, determine names for the categories. SAP recommends that you name the final category “Miscellaneous;” this adds all applications and messages that do not match a category to the Miscellaneous category. Also in this example, all applications that belong to a category must include the category name contained in their display name. For example, an application named “Financial Claim” belongs in the “Financial” category.

There are other ways to determine categories; if you know the names of the applications in advance, you can simply list all the application names that belong in each category.

1. Open the `AppScreen.java` file for editing and add:

```
import java.util.Vector;
import net.rim.device.api.util.Comparator;
```

2. Add a list of categories as a private final member variable to the `AppScreen` class, for example:

```
private final String[] m_asHybridAppCategories = { "Financial",
"Utilities", "Miscellaneous" };
```

3. In the constructor of `AppScreen`, replace the compare method in the `Comparator` with the following modified version:


```
// BLACKBERRY_CUSTOMIZATION_POINT_CATEGORIZEDVIEWS
m_oApps.setSortComparator(new Comparator()
{
public int compare(Object oApp1, Object oApp2)
{
return 0;
}
});
```

Although you can sort with categories, doing so becomes complicated since you must check whether an element is a category name or a Hybrid App, and you typically want to sort only Hybrid Apps within a common category.

4. Replace the populateList method with this modified version:

```
private void populateList()
{
m_oApps.removeAllElements();
Vector vHybridApps = BBHybridAppHelper.getClientHybridApps();
for (int i = 0; i < m_asHybridAppCategories.length; i++)
{
m_oApps.addElement(m_asHybridAppCategories[i]);
for (int j = 0; j < vHybridApps.size(); j++)
{HybridApp ha = (HybridApp) vHybridApps.elementAt(j);
if (ha.getDisplayName().indexOf(m_asHybridAppCategories[i]) >= 0
|| i + 1 == m_asHybridAppCategories.length)
{m_oApps.addElement(ha);vHybridApps.removeElementAt(j--);
}
}
}
}
```

5. Replace the drawListRow method in ListFieldCallback with this modified version:

```
public void drawListRow(ListField listField, Graphics graphics,
int index, int y, int width) {
//
BLACKBERRY_CUSTOMIZATION_POINT_CATEGORIZEDVIEWS
// y parameter is already offset to center
text
int iOffset = (listField.getRowHeight() -
getFont().getHeight()) >> 1;
//
BLACKBERRY_CUSTOMIZATION_POINT_HYBRIDAPPLIST
// HybridApp oApp = ( HybridApp )
m_oApps.elementAt( index );
// BLACKBERRY_CUSTOMIZATION_POINT_COLORS
final int iMargin = 2;
Object element =
m_oApps.elementAt( index );
```

```

        if( element instanceof HybridApp )
        {
            HybridApp oApp = ( HybridApp ) element;

            // Draw image

            EncodedImage oImage
= EncodedImage.getEncodedImageResource( "ampicon" +
oApp.getIconIndex() + ".png" );

            Bitmap oBitmap = oImage.getBitmap();

            graphics.drawBitmap( iMargin, y - iOffset +
( listField.getRowHeight() - oBitmap.getHeight() ) / 2,
oBitmap.getWidth(), oBitmap.getHeight(), oBitmap, 0, 0 );

            // Draw text
            graphics.drawText( oApp.getDisplayName(),
2 * iMargin + oBitmap.getWidth(), y );
        }
        else
        {
            // element must be a String
            String sCategoryName = (String) element;

            graphics.drawText( sCategoryName, iMargin, y );
        }
    }
}

```

6. Replace the navigationClick method in the AppScreen class with this modified version:

```

protected boolean navigationClick(int status, int time)
{
    Field oField = getFieldWithFocus();
    if ( oField instanceof ListField )
    {
        int iIndex = ( ( ListField )
oField ).getSelectedIndex();

        if ( iIndex != -1 && m_oApps.size() > 0 )
        {
            Object oElement = m_oApps.elementAt( iIndex );

            if( oElement instanceof HybridApp )
            {
                HybridApp oApp = ( HybridApp )
oElement;
            }
        }
    }
}

```

```

        XmlHybridApp.startHybridApp( oApp.getModuleId(),
oApp.getVersion(), false );
                                return true;
                                }
        }
        return super.navigationClick(status, time);
    }
}

```

7. Replace the `onHybridAppAdded` method in the `HybridAppsListener` with this modified version:

```

public void onHybridAppAdded(HybridApp oHybridApp) {
    onRefreshRequired();
}

```

8. Save the `AppScreen.java` file.
9. Open the `CustomizationHelper.java` file, which is located in the `... \HybridWebContainer\src\com\sybase\hwc` folder and edit the `getHybridAppScreenClass()` method, to change the class returned to your new class.

Making the List of Hybrid Apps Searchable

Add a search field to the top of the Hybrid App list.

Whenever the contents of the search field change, only Hybrid Apps with matching names are listed. The comment tag associated with this customization is

```
BLACKBERRY_CUSTOMIZATION_POINT_HYBRIDAPPSEARCH.
```

1. Open the `AppScreen.java` file for editing and add the following member variable to the `AppScreen` class:

```
private String m_sSearchFor;
```

2. Add the following code in the constructor of `AppScreen`, before the line that says `// Add list field to screen`:

```

//add in the search UI
LabelField searchLabel = new LabelField( "Search: " );
add( searchLabel );
EditField searchEdit = new EditField();
searchEdit.setChangeListener( new SearchFieldListener() );
add( searchEdit );
m_sSearchFor = "";

```

3. Add the following code to the end of the `populateList` method:

```

// BLACKBERRY_CUSTOMIZATION_POINT_HYBRIDAPPSEARCH
for (int i = 0; i < m_oApps.size(); i++) {
    HybridApp ha = (HybridApp) m_oApps.elementAt(i);
    if( m_sSearchFor == null || m_sSearchFor.equals("") ||
ha.getDisplayName().indexOf( m_sSearchFor ) >= 0 )

```

```
{
    // there is no search, or this Hybrid App matches the
    search.
    // do nothing since the Hybrid App is already in the list
}
else
{
    // there is a search and this Hybrid App does not match
    // remove this Hybrid App from the list
    m_oApps.removeElementAt(i);
    i--;
}
}
```

4. Add the following class to the AppScreen class:

```
final class SearchFieldListener implements FieldChangeListener
{
    public void fieldChanged( Field field, int context)
    {
        if( field instanceof EditField )
        {
            EditField oEditField = (EditField) field;
            m_sSearchFor = oEditField.getText();
            populateList();
            updateScreen();
        }
    }
}
```

5. Open the CustomizationHelper.java file, which is located in the . . . \HybridWebContainer\src\com\sybase\hwc folder and edit the getHybridappScreenClass() method, to change the class returned to your new class.

Customizing the Push Notification Handler in the BlackBerry Hybrid Web Container

The comment tag associated with this customization is

```
BLACKBERRY_CUSTOMIZATION_POINT_PUSH_NOTIFICATION.
```

By default, when a push notification is received by the Hybrid Web Container push listener, it returns the `PushNotificationListener.NOTIFICATION_CONTINUE` method, which allows the next push listener to handle the notification.

The comments in the `onPushNotification` method in the `CustomizationHelper.java` file include sample code that demonstrates how to open the default client-initiated Hybrid App if no Hybrid App is currently opened and also, optionally, calls a JavaScript method to initialize the Hybrid App once it is opened.

1. Open the `CustomizationHelper.java` file for editing.
2. Find the `onPushNotification` method and make your changes.

For example, if `PushNotificationListener.CANCEL` is returned, the push listener manager does not invoke the next push notification listener.

3. Save the file.
4. Rebuild the project.

Upgrading the PhoneGap Library Used by the BlackBerry Hybrid Web Container

SAP Mobile Platform includes the Cordova (PhoneGap) 2.0 libraries. Follow these steps if you want to upgrade the BlackBerry Hybrid Web Container to a more recent version of the Cordova library.

This procedure describes upgrading the Cordova library from version 2.0.0 to version 2.9.0. The steps to upgrade to other versions differ slightly. Since the Hybrid Web Container template project does not include the source code for building `HybridAppLib.jar`, the ability to upgrade Cordova to newer versions is limited, and certain new Cordova features may not work properly in Hybrid Web Container project.

Note: Upgrading the Hybrid Web Container container to use Cordova 3.0.0 is not supported because the Hybrid Web Container project does not work with Cordova 3.0.0 CLI.

1. Download `phonegap 2.9.0` from *phonegap.com*, and unzip it to a local folder.
2. After unzipping the `phonegap2.9.0` zip file, go into the `\blackberry\bbos\framework\ext` folder, and import the project to Eclipse blackberry plugin to compile/package the source code. The `Cordova.jar` can be found under the `deliverables` directory, in your project.
3. Open Eclipse and import the HWC template project.
4. Expand the HWC template project, and delete the `PhoneGapExtension.jar` file from the `libs` folder. Copy the `cordova.jar` file built above and copy it to the `libs` folder.
5. Right-click the Hybrid Web Container project and click the **Properties** menu. Select **Java Build Path > Libraries**. Select the `PhoneGapExtension.jar` file, then remove the old jar file.
6. Select **Add JARs...** and expand the **HybridWebContainer\libs** node. Select the new **cordova-2.9.0.jar** file, and click **OK** to confirm the selection.
7. Right-click the Hybrid Web Container project and click the **Properties** menu. Select **Java Build Path > Order and Export**. Select the checkbox for `Cordova.jar` and **OK** to close the “Properties” dialog.
8. Update the `com.sybase.hwc.amp.HWCBrowserFieldListener.java` class, since the constructor of class `CordovaExtension` was changed in 2.9.0.

Open the `com.sybase.hwc.amp.HWCBrowserFieldListener.java` class and make this change:

```
*****line 323*****
if( extension instanceof CordovaExtension )
```

```
{
    extension = new
CordovaExtension(getClass().getResourceAsStream("/xml/
plugins.xml"));
}
```

Change to:

```
if( extension instanceof CordovaExtension )
{
    extension = new CordovaExtension();
}
```

9. Update the `com.sybase.hwc.amp.HWCWidgetConfigImpl.java` class:

*****line 175*****

```
widgetExtensions.addElement( new
CordovaExtension(this.getClass().getResourceAsStream("/xml/
plugins.xml")) );
```

Change to:

```
widgetExtensions.addElement( new CordovaExtension() );
```

10. Clean the HWC project and have Eclipse build the HWC project.
11. If the `cordova.js` file is used in your `HybridApp.js` app, you must also update `cordova.js` to the one provided with the new cordova library.

iOS Hybrid Web Container Customization

The Hybrid Web Container project that comes with SAP Mobile Platform is accompanied by libraries and the source code necessary for you to build the Hybrid Web Container.

Before getting started, unzip the directory that contains the Hybrid Web Container project as outlined in *Building the Hybrid Web Container Using the Provided iOS Source Code*. The Hybrid Web Container project unzips to a directory called HWC. Any references to a directory path in these procedures are relative to that top-level HWC directory.

The HWC directory contains directories such as `Classes`, `libs`, and `includes`, as well as images and other files. It also contains the `HWC.xcodeproj`, which is the Xcode project that builds the Hybrid Web Container, and is the project that is referenced in the customization procedures.

Whenever a customization requires a source code modification, there is a reference to “touch points” in the code. These references are annotated with `IOS_CUSTOMIZATION_POINT` and a descriptor identifying the customization to which they belong.

For example, all code areas associated with removing the PIN screen are annotated with `IOS_CUSTOMIZATION_POINT_PIN`. The touch points are typically accompanied by

brief comments in the code explaining the necessary changes. Only source code files contain these touch points. The procedures describe where to modify plist files, strings files, and other non-source code files, but you must locate where to apply those changes.

The `CustomizationHelper.m` file included in the HWC project under the `Classes` group folder in the Xcode Project Navigator is used to encapsulate some of your customizations in a single place. In many cases, this file contains sample implementations of the customizations that you can follow.

Note: After performing any customizations, you must rebuild the project. SAP recommends that you always test your changes before using the resulting application.

iOS Customization Touch Points

All code areas associated with iOS Hybrid Web Container customizations are annotated with `IOS_CUSTOMIZATION_POINT_<customization>` comment tags, or touch points.

Touch Point	Description
<code>IOS_CUSTOMIZATION_POINT_PRESET-SETTINGS</code>	Provides alternative ways to get connection settings so they do not show up on the Settings screen. This prevents the user from changing them. There are variations on this customization.
<code>IOS_CUSTOMIZATION_POINT_DEFAULT-SETTINGS</code>	Set the defaults for the Settings screen.
<code>IOS_CUSTOMIZATION_POINT_PREPACKAGED_APP</code>	Include a prepackaged Hybrid App that launches automatically when the Hybrid Web Container starts.
<code>IOS_CUSTOMIZATION_POINT_PIN</code>	Use for PIN screen customizations, or to remove the PIN screen.
<code>IOS_CUSTOMIZATION_POINT_SORTING</code>	Sort Hybrid Apps or messages based on different criteria.
<code>IOS_CUSTOMIZATION_POINT_FILTERING</code>	Filter the list of Hybrid Apps or messages so only items meeting certain criteria are shown.
<code>IOS_CUSTOMIZATION_POINT_HTTPHEADERS</code>	Set HTTP headers for the iOS Hybrid Web Container to include authentication tokens.
<code>IOS_CUSTOMIZATION_POINT_FONTS</code>	Customize fonts in the Hybrid Web Container.
<code>IOS_CUSTOMIZATION_POINT_SPLASH-SCREEN</code>	Change the splash screen, or the length of time for which it is shown.

Touch Point	Description
IOS_CUSTOMIZATION_POINT_PUSH_NOTIFICATION	Customize how the Hybrid Web Container handles the push notification.
IOS_CUSTOMIZATION_POINT_ANONYMOUS_USER	Returns whether or not anonymous user support is being used. Change to YES to allow clients to register anonymously. Note: For this to work, the HWC application connection template must be configured to use the anonymous security configuration. See <i>Application Connection Templates</i> in <i>SAP Control Center for SAP Mobile Platform</i> .
IOS_CUSTOMIZATION_POINT_HTTPS_CLIENT_CERT_LISTENER	Customize how to handle client certificate authentication challenge.

Look and Feel Customization of the iOS Hybrid Web Container

Customizations you can make to the look and feel include changing the splash screen, changing the Hybrid App icons and name, changing the Hybrid App package icons, changing labels and text, and adding support for new languages.

Changing the Hybrid Web Container Application Icon

Modify the application icon shown on the home screen by replacing the image files in the HybridWebContainer directory.

1. Go to the HybridWebContainer directory, which is in the location where you unpacked the iOS_HWC_<version.>tar.gz file.
2. Open the HWC.xcodeproj project with XCode 5 or above.
3. In the left project panel, select **HWC > Resources > Images.xcassets**.
4. Select **AppIcon** in the opening main view.
5. Select new images, and drag and drop the app icons you want to replace.

Note: The new icon files must use the same name as those you replaced, including the file extension, and they must have the same resolution as the original images.

6. Rebuild the HWC.xcodeproj project.
7. From the Xcode menu, select **Product > Clean**.
8. Select **Product > Build**.
9. Click **Run**.

Changing the iOS Hybrid App Name

Edit a `plist` file to modify the application name.

1. In Xcode, use Project Navigator to find the file named `HWC-Info.plist`.
2. Open the file and change the **Bundle display name** to the new name.
3. Save the file.
4. Rebuild the `HWC.xcodeproj` project.
 - a) From the Xcode menu, select **Product > Clean**.
 - b) Select **Product > Build**.

Customizing the Splash Screen

The splash screen is the first screen that appears when you start the Hybrid Web Container.

You can change either the image that is shown, or you can change the length of time that it appears. The splash screen is stored on a per-language basis in the `HybridWebContainer/<language>.lproj` directories. In each of these directories, there are three files that contain the splash screens for iPhone (`Default.png`) and iPad (`Default-Landscape.png` and `Default-Portrait.png`).

You must replace the file in each language subdirectory, or your new splash screen does not appear when the language setting is changed. The splash screen does not include any localizable strings, so you must provide the correct screen for each language, if you plan to support multiple languages.

1. Add a custom splash screen by replacing the appropriate files in the `HybridWebContainer/<language>.lproj` directory.

Note: The new image files must use the same name as those you replaced, including the file extension, and they must have the same resolution as the original images.

2. Rebuild the `HWC.xcodeproj` project.
 - a) From the Xcode menu, select **Product > Clean**.
 - b) Select **Product > Build**.

Changing Labels and Text

You can customize most of the text found in labels, dialogs, or error messages used by the Hybrid Web Container.

Changes that you can make include:

- Buttons, labels, and error messages – these strings are in `Localizable.strings`, under the `Resources/<language>.lproj` group folders in the Xcode Project Navigator.

- Application branding – strings that identify the application, among other things. These strings are in `Branding.strings`, under the `Resources/<language>.lproj` group folder in the Xcode Project Navigator.
- About box – these strings are in `About.strings`, under the `Resources/Settings.bundle/en.lproj` folder. Expand the `Settings.bundle` under the `Resources` group folder in the Xcode Project Navigator. Here, you can change the company name or the version number that is shown in the About box in the Settings screen.

Keep in mind that for any change you make you must also make equivalent changes for each language if you want your changes to translate across other languages.

When modifying one of the `*.strings` files, you need only to change the second string value. For example, to change the `AppId` in `Branding.strings`, on this line: `AppId = HWC`, change only the "HWC."

Adding a New Language

Add support for new languages by dropping new `<language>.lproj` directories into the project.

By default, the **hybrid-container** is localized to several different languages. Localized resources are in `<language>.lproj` directories and group folders throughout the project, where *<language>* may be the full language name, or a two-digit country code. The simplest way to add a new language is to copy existing `lproj` directories for another language, translate the strings into the new language, and add the new `lproj` directories to the project.

This procedure uses English as a starting point.

1. Copy `HybridWebContainer/English.lproj` directory to `HybridWebContainer/<new_language>.lproj`.
This contains resources for the PIN screens and for the splash screen. You can localize or entirely redesign the PIN screen .
2. Add the newly created `HybridWebContainer/<new_language>.lproj` directory to the project, at the top level (not under any group folders).
3. In Finder, right-click `HybridWebContainer/Settings.bundle`, and select **Show Package Contents**.
The `Settings.bundle` directory opens.
4. Copy `en.lproj` to `<new_language>.lproj`.
5. Translate the strings in `Root.strings` (these are the strings that identify names of settings in the Settings screen) and `About.strings` (associated with the About box).
6. In Xcode, in the Project Navigator, find the newly created `<new_language>.lproj` directory under the `Resources/Settings.bundle`.

You do not need to explicitly add the new directory to the project, but you should verify it is there.

7. Copy `HybridWebContainer/strings/English.lproj` to `HybridWebContainer/strings/<new_language>.lproj`.
8. Translate the strings in `Branding.strings` and `Localizable.strings`.
9. In Project Navigator, add the newly created `HybridWebContainer/strings/<new_language>.lproj` directory to the project under the **Resources** group folder.

Default Behavior Customization for the iOS Hybrid Web Container

You can change the default behavior of the iOS Hybrid Web Container, including customizing or removing the PIN screen, changing the default behavior for the way the application launches, sorting and filtering the list of Hybrid App packages and messages, and so on.

Customizing PIN Screens on iOS

PIN screens prompt the user to either create or enter a password, respectively.

You can modify the PIN screens with custom text, or you can redesign them entirely. PIN screens include Create PIN and Enter PIN screens.

The PIN screens are stored in `.xib` files in the `HybridWebContainer/<language>.lproj` directories:

- `CreatePasswordViewController.xib` – constructs the Create Password screen
- `EnterPasswordViewController.xib` – constructs the Enter Password screen

Creating New PIN Screens

You can completely redesign the PIN screens by modifying the `.xib` files.

1. Using Interface Builder, open the `CreatePasswordViewController.xib` and `EnterPasswordViewController.xib` files located in `HybridWebContainer/<language>.lproj`.

2. Make your modifications.

You can change the look and feel of buttons, change the text, or change the background. You likely do not want to remove buttons or fields, as doing so interferes with the functioning of the application.

Note: You must make the equivalent changes to each language for your new PIN screen to show correctly in other languages.

3. Rebuild the `HWC.xcodeproj` project.
 - a) From the Xcode menu, select **Product > Clean**.
 - b) Select **Product > Build**.

Changing Localizable Strings in the PIN Screen

To modify the text, you must change strings files.

Each of the PIN screen .xib files has a corresponding strings file with the same name with .strings appended to the end, for example, HybridWebContainer/<French>.lproj\CreatePasswordViewController.xib.strings.

1. Open the CreatePasswordViewController.xib.strings and EnterPasswordViewController.xib.strings files, which are located in HybridWebContainer/<language>.lproj.
2. Modify and save the files.
3. Regenerate the .xib files:
 - a) Open a Terminal window.
 - b) Navigate to the HybridWebContainer directory, and execute:

```
ibtool --strings-file <language>.lproj/<strings file>  
<language>.lproj/<xib file> --write <language>.lproj/  
<xib file>
```

Note: <language> must be the same throughout, and the .strings file must correspond with the .xib file.

4. After rebuilding the .xib files, you can return to Xcode and view the new screens before rebuilding the Hybrid Web Container.

Removing the PIN Screen

You can disable and remove the PIN screen by making a minor code modification to the CustomizationHelper.m file.

Note: If you have previously used the Hybrid Web Container with a password on a particular device, you will no longer be able to access the encrypted database, or any data stored there, and the application may not work correctly if you remove the PIN screen. In this case, uninstall the Hybrid Web Container from the device before using the Hybrid Web Container without a PIN screen. For a simulator, click **Reset Content and Settings** first.

Note: Removing the PIN screen leaves data that is stored on the device less secure. You should remove the PIN screen only if you are not concerned about keeping your data secure.

All code areas associated with removing the PIN screen are annotated with IOS_CUSTOMIZATION_POINT_PIN.

1. In Xcode Project Navigator, open the CustomizationHelper.m file, which is located in HWC\Classes.
2. Find the usePIN function and change it to return NO instead of YES.

3. Save the file.
4. Rebuild the HWC.xcodeproj project.
 - a) From the Xcode menu, select **Product > Clean**.
 - b) Select **Product > Build**.

Using Default Connection Settings

You can customize the Hybrid Web Container so that it is pre-populated with connection settings, or to use default values if nothing is provided by the user, or to always use default values on startup.

These customizations involve changes to either `Root.plist` or `CustomizationHelper.m`.

All code areas associated with removing fields from the Settings screen are annotated with `IOS_CUSTOMIZATION_POINT_DEFAULTSETTINGS`. The customizations described here assume the Settings screen is used as the interface for providing input from the user. For alternatives to using the default Settings screen, see *Removing Fields from the Settings Screen*.

1. In the Xcode project, in the Project Navigator, expand **Resources > Settings.bundle** and open the `Root.plist` file.
2. Expand the item for the settings you want to preset, and fill in the **DefaultValue** attribute.

Most settings do not have default values, with the exception of the protocol and the registration method. Because these settings have a "Multi Value" **Type** in the `.plist` file (instead of Text Field), they always have a default value that is one of the accepted values listed in Values. You can open the **Values** tab to see the acceptable values for these settings.

This example sets a default value of **443** for the server port, and sets the default protocol to **HTTPS**. The **Values** item is expanded and shows the acceptable values.

Key	Type	Value
Strings Filename	String	Root
▼ Preference Items	Array	(19 items)
▶ Item 0 (Group -	Diction...	(2 items)
▼ Item 1 (Text Field -	Diction...	(7 items)
Type	String	Text Field
Title	String	ServerNameSetting
Identifier	String	servername_preference
Default Value	String	
Text Field Is Secure	Boolean	NO
Keyboard Type	String	URL
Autocorrection Style	String	No Autocorrection
▼ Item 2 (Text Field -	Diction...	(6 items)
Type	String	Text Field
Title	String	ServerPortSetting
Identifier	String	serverport_preference
Default Value	String	443
Text Field Is Secure	Boolean	NO
Keyboard Type	String	Number Pad
▼ Item 3 (Text Field -	Diction...	(7 items)
Type	String	Text Field
Title	String	CompanyIDSetting
Identifier	String	companyid_preference
Default Value	String	
Text Field Is Secure	Boolean	NO
Keyboard Type	String	Alphabet
Autocorrection Style	String	No Autocorrection
▼ Item 4 (Multi Value -	Diction...	(6 items)
Type	String	Multi Value
Title	String	ProtocolSetting
Identifier	String	protocol_preference
▼ Values	Array	(2 items)
Item 0	String	HTTP
Item 1	String	HTTPS
▶ Titles	Array	(2 items)
Default Value	String	HTTPS
▶ Item 5 (Text Field - URL Prefix)	Diction...	(7 items)
▶ Item 6 (Group -	Diction...	(2 items)
▼ Item 7 (Multi Value -)	Diction...	(6 items)
Type	String	Multi Value
Title	String	
Identifier	String	registrationmethod_preference
▶ Values	Array	(3 items)
▶ Titles	Array	(3 items)
Default Value	String	0

Note: Pre-populating a value only sets its initial value on a one-time basis; it does not prevent the user from later changing it, nor does it prevent a server change from overwriting it. This approach also cannot be combined with the *Removing Fields from the Settings Screen* customization because it relies on using the settings bundle.

3. Save the file.
4. Rebuild the `HWC.xcodeproj` project.
 - a) From the Xcode menu, select **Product > Clean**.
 - b) Select **Product > Build**.

Removing Fields from the Settings Screen

Customize the Settings screen to prevent certain settings from showing.

For example, you can preset the server port connection value, and then choose not to display that field in the Settings screen, bypassing the user’s ability to change or see that field. If you want this behavior, but you want the user to also see the property value, see *Using Default Connection Settings*.

All code areas associated with removing fields from the Settings screen are annotated with `IOS_CUSTOMIZATION_POINT_PRESETSETTINGS`.

Keep in mind that connection settings sometimes have more than one “internal” name because different developers may reference the same settings using different names, particularly in local variable names. For example:

- server name = server id
- company id = farm id
- activation code = validation code

1. In the Xcode project, in the Project Navigator, expand **Resources > Settings.bundle** and open the `Root.plist` file.

2. Delete the dictionary item that corresponds to the setting to remove from the Settings screen.

For example, to remove the server port setting, delete the Text Field item with the title `ServerPortSetting`.

3. Save the file.

4. Rebuild the `HWC.xcodeproj` project.

a) From the Xcode menu, select **Product > Clean**.

b) Select **Product > Build**.

5. For each property you remove from the Settings screen, you need to provide a way to configure that property.

See *Using Default Connection Settings*.

Using Multiple Hybrid Web Containers on the Same iOS Device

You can configure two or more Hybrid Web Containers to coexist on the same device.

This customization allows two or more independent users to use the same device, but with their own private version of the application. In summary, you need to change the application ID, the bundle identifier, and possibly the URL scheme.

The application ID is used by the server to identify the application, and because of this, you cannot run two applications on the same device with the same application ID. By default, the Hybrid Web Container uses “HWC” for its application ID. Changing the application ID involves a minor change to `CustomizationHelper.m`. Additionally, you must signify to iOS that this is a distinct application. This requires a minor change to update the application

bundle ID in the `plist` file. Finally, if your application needs to communicate with the Afaia client for provisioning your application or retrieving a certificate, you need to specify a unique URL scheme in the `plist` file. If your application does not need to communicate with the afaia client, then you should delete the “URL types” item from the same `plist` file.

1. Change the project name:
 - a) In the Xcode Project Navigator, click on the root Hybrid Web Container element.
 - b) With the Hybrid Web Container element highlighted click on the Hybrid Web Container text to rename.
 - c) Change the name of the Hybrid Web Container element to your new project name.
 - d) A window to rename project content items appears. Click **Rename**.
2. Change the application ID:
 - a) In Xcode Project Navigator, find and open the `CustomizationHelper.m` file, which is located in the `Classes` group folder,
 - b) Locate the customization point that accompanies the `getAppId` function, and change it so that it returns a unique name.
 - c) Save and close the file.
3. To differentiate this version of the Hybrid Web Container from another:
 - a) In Xcode Project Navigator, find and open the `HWC-Info.plist` file, which is located in the `Resources` group folder.
 - b) Change the bundle identifier value to something unique.
 - c) Save and close the file.

The container template project has a URL schema setting in the project `plist` file, which is used to communicate with Afaia client.
4. To avoid multiple container applications from interfering with each other when communicating with the Afaia client, the URL schema must be unique among all container applications that are installed on the device, otherwise, the application may be launched by the afaia client by mistake, or fail to launch altogether.
 - a) In Xcode Project Navigator, find and open the `HWC-Info.plist` file, which is located in the `Resources` group folder.
 - b) Expand the **URL types item > Item 0 > URL Schemes item**.
 - c) Select **Item 0**, and change its value to a unique value among all other applications.
 - d) Save and close the file.

Sorting and Filtering the List of Hybrid App Packages and Messages

By default, the Hybrid Web Container sorts the list of applications and messages in alphabetical order by package name.

There is no filtering by default.

You can sort and filter this list in any way you want. For example, you can filter Hybrid App packages from appearing according to whatever criteria you specify. You can filter out

particular Hybrid App packages by name, or you can sort Hybrid App messages by subject. Hybrid App messages are server-initiated messages associated with a Hybrid App package, and appear in a separate TableView.

The sorting and filtering is done using arrays of NSSortDescriptor and NSPredicate objects, respectively. These arrays can be initialized at application startup, and can also be changed dynamically, giving you the ability to change the sorting or filtering criteria while the application is running.

The `HybridAppViewController.h` file defines the interface for a Hybrid App object. You can sort and filter the properties of this object.

1. Locate the `HybridAppViewController.h` file.

You do not need to modify this file, but you can view the properties of a Hybrid App object on which you might want to filter or sort.

This file is included in the `HWC/includes` directory, but it is not explicitly included in the Xcode project. To get the file to appear in the Xcode editor:

- a) In Xcode, open the `HWC.xcodeproj`.
- b) Open the `WidgetFolderController.h` file.
- c) Locate this line: `#import "HybridAppViewController.h"`, right-click inside the quotes, then select **Jump to Definition**.

Xcode opens the file.

2. Customizations involving filtering and sorting for both Hybrid App packages and messages can be made in the `CustomizationHelper.m` file.

- a) In Xcode Project Navigator, open the `CustomizationHelper.m` file, which is located in `HWC\Classes`.
- b) If you are customizing sorting behavior, locate the `IOS_CUSTOMIZATION_POINT_SORTING` customization tags that accompany these functions:

- `initializeHybridAppSortingDescriptors`
- `initializeMessageSortingDescriptors`
- `addHybridAppSortDescriptor`
- `addMessageSortDescriptor`
- `clearHybridAppSortDescriptors`
- `clearMessageSortDescriptors`

Customize the initialize functions to add sort descriptors at application startup. If you want to dynamically change the sorting criteria, you can call the add functions to add a sort descriptor to the end of the array, or you can call the clear functions to start over and then add to a clean array. Typically, you do not need to modify the add or clear functions.

The sort descriptor array is processed in order, so descriptors that appear toward the end of the array are only used when descriptors earlier in the array result in a tie between two elements. This allows you to sort on multiple property keys.

- c) If you are customizing filtering behavior, locate the `IOS_CUSTOMIZATION_POINT_FILTERING` customization tags that accompany these functions:
- `initializeHybridAppFilterPredicates`
 - `initializeMessageFilterPredicates`
 - `addHybridAppFilterPredicate`
 - `addMessageFilterPredicate`
 - `clearHybridAppFilterPredicates`
 - `clearMessageFilterPredicates`

Customize the **initialize** functions to add filter predicates at application startup. If you want to dynamically change the filtering criteria, you can call the **add** functions to add a filter predicate to the end of the array, or you can call the **clear** functions to start over and then add to a clean array. Typically, you do not need to modify the **add** or **clear** functions.

3. Save the file.
4. Rebuild the `HWC.xcodeproj` project.
 - a) From the Xcode menu, select **Product > Clean**.
 - b) Select **Product > Build**.

Changing to a New UI Control

You can change the way the list of Hybrid App packages and messages appear.

Hybrid Web Container uses `UITableView` objects to display the list of Hybrid App packages and messages. To change this behavior, you must completely rewrite some files. This procedure shows an example of a fully functional Cover Flow style view. You can use any UI library.

This customization involves rewriting one or two classes, depending on whether you want to customize the appearance of the application list or the messages list, or both. The application list view is in the `HybridAppsFolderView (.m and .h)` files, while the messages list view is in the `MessagesFolderView (.m and .h)` files. You can change the appearance of one or the other independently of one another.

This customization is not too difficult if you use the existing classes as an example. For the most part, you can (and probably should) reuse a lot of the code in the original classes. You will likely see the biggest divergence when you replace the `UITableViewDelegate` and `UITableViewDataSource` functions, as well as the code that creates cells. This code is tailored to a `UITableView`, but you will probably find that the UI library you are trying to replace it with will have callback functions that accomplish similar things. In many cases, you will be able to copy and paste code from the original functions into your new class with very

few modifications needed. The sample code provides very rudimentary views, but you can experiment with different views.

This example uses an open source UI library called iCarousel, available under the zlib License. The source is at <http://cocoacontrols.com/platforms/ios/controls/icarousel>. This example replaces the UI for the applications folder, while leaving the messages folder unchanged.

1. Download the iCarousel source code.
2. Copy the `iCarousel.h` and `iCarousel.m` files to the `HWC/Classes` directory, then add these files to the `Classes` group folder in the Project Navigator in Xcode.
Do not drag and drop the files into the `Classes` group folder, or they will not be incorporated into the project build phase. Instead, right-click the `Classes` group folder, and select **Add Files to HWC...**
3. If you are viewing this guide online from the Product Documentation Web site, click [*iOS_HWC_Customization_Supplement.zip*](#) to access the ZIP file containing new copies of `HybridAppsFolderView.h` and `HybridAppsFolderView.m`.
4. Drop the unzipped `HybridAppsFolderView` files into the `HybridWebContainer/Classes` directory, overwriting the original files.
You can customize the code to suit your needs, for example, you may want to design your own `UIViews`, or change from a cover flow to any of the other supported view types within iCarousel, or to a different UI library altogether.

Setting HTTP Headers

You can set HTTP headers for the iOS Hybrid Web Container to include authentication tokens.

There are three sample methods showing how to do this in the iOS Hybrid Web Container template source code, which include:

- `setHttpHeaders` – use this method to set the authentication tokens. The tokens you set are used from then on until `setHttpHeaders` is called again.
- `onHybridAppTokenError` – use this method to call `setHttpHeaders` to put the authentication tokens back in a good state, if, for example, they have expired.
- `onHTTPError` – use this method to handle HTTP errors.

All code areas associated with HTTP header customization are annotated with `IOS_CUSTOMIZATION_POINT_HTTPHEADERS`.

1. Open the `CustomizationHelper.m` file, which is located in `HybridWebContainer\Classes`.
2. Locate the `setHttpHeaders` method, and uncomment its contents.

The stub code that is provided shows an example of how to add headers and cookies. You simply need to replace the header and cookie assignments with your own. The `setHttpHeaders` function is already called in the `startEngine` function just

before the client engine starts, so you need to provide the implementation of `setHttpHeaders`.

3. `CustomizationHelper.m` also includes stub implementations of `onHybridAppTokenError` and `onHTTPError` that you can implement. The `onHybridAppTokenError` method is called when Hybrid App token authentication failure occurs, so it is a good idea to use this callback as an opportunity to refresh the HTTP headers again. A common way to do this is to maintain member variables that contain the values for the headers you want to set. Implement the `setHttpHeaders` function to use the values in those member variables when it sets the headers, then, in `onHybridAppTokenError`, you can update the member variables with the new header values, and then call `setHttpHeaders` again, for example:

```
[[CustomizationHelper getInstance] setHttpHeaders];
```

4. If you have custom code to run when an HTTP error occurs, add it to the `onHTTPError` function.

This method is called any time there is an HTTP error. You can use this to inform the user of errors, or log errors, or perform other custom steps in response to particular error codes.

Customizing the Push Notification Handler in the iOS Hybrid Web Container

Customize the way the Hybrid Web Container handles push notifications.

By default, when a push notification is received by the Hybrid Web Container push listener, the `kNotificationContinue` method is returned, which allows the next push listener to handle the notification. The comments in the `onPushNotification` method in the `HWCAppDelegate.m` file includes some sample code that demonstrates how to open the default client-initiated Hybrid App if no Hybrid App is currently opened.

The comment tag associated with this customization is `IOS_CUSTOMIZATION_POINT_PUSH_NOTIFICATION`.

1. Open the `HWCAppDelegate.m` file for editing.
2. Find the `onPushNotification` method and make your changes.
For example, if `kNotificationCancel` is returned, the push listener manager does not invoke the next push notification listener.
3. Save the file.
4. Rebuild the `HWC.xcodeproj` project.
 - a) From the Xcode menu, select **Product > Clean**.
 - b) Select **Product > Build**.

Hiding the Listview on iPad

Hide the listview on the iPad when in landscape orientation so the Hybrid App opens in the full screen.

When the Hybrid Web Container runs on iPad, it uses a `UISplitViewController` to display its main views. The list of Hybrid Apps and messages occupies the left-hand view (the master

view), while the Hybrid App contents occupy the right-hand view (the details view). By default, the master view hides away while the device is in the portrait orientation, and can be accessed using a button on the navigation bar. The master view is presented side-by-side with the detail view while the device is in the landscape orientation. To hide the listview when using landscape orientation so the Hybrid App opens in full screen, use the customization tag `IOS_CUSTOMIZATION_POINT_IPAD_LIST_VIEW`.

Note: This customization is not supported on iOS 4.3. On iOS 5.1 and later, this customization disables the ability to present the master view with a swipe gesture, which is enabled by default.

1. In Xcode Project Navigator, find and open the `CustomizationHelper.m` file, which is located in the `Classes` group folder.
2. Locate the `shouldHideIpadListView` function and change it so it returns `YES`.
3. Rebuild the `HWC.xcodeproj` project.
 - a) From the Xcode menu, select **Product > Clean**.
 - b) Select **Product > Build**.

Handling Client Certificate Challenge

Customize the client certificate authentication challenge.

To customize the client certificate authentication challenge, uncomment the `onClientCertificateChallenge` method in the source `CustomizationHelper.m` file, then implement the logic to get a certificate identity as described by the comment within the method.

Note: It is not safe to display a model view within this method, as there may be another model view already displayed on the screen. Refer to the `[HWCAppDelegete onClientCertificateChallenge]` method in `HWCAppDelegate.m` for details of this model view issue.

Upgrading the PhoneGap Library Used by the iOS Hybrid Web Container

SAP Mobile Platform includes the Cordova 2.0 libraries. Follow these steps if you want to upgrade the iOS Hybrid Web Container to a more recent version of the Cordova library.

The Cordova library used by the Hybrid Web Container uses source code that has been modified from the original source, primarily because the original source does not support some Hybrid Web Container features. This procedure describes upgrading the Cordova library from version 2.0.0 to version 2.9.0. The steps to upgrade to other versions differ slightly. Since the Hybrid Web Container template project does not include the source code for building `HWCLib.a`, the ability to upgrade Cordova to newer versions is limited, and certain new Cordova features may not work properly in Hybrid Web Container project.

Note: Upgrading the Hybrid Web Container container to use Cordova 3.0.0 is not supported because the Hybrid Web Container project does not work with Cordova 3.0.0 CLI.

1. From a browser on your Mac, download phonegap 2.9.0 from *phonegap.com*, and unzip it to a local folder.
2. Open terminal app, go to the unzipped folder `phonegap-2.9.0/lib/ios/bin`, and create a Cordova prototype project using this command:

```
$ bash create </path>/hello hello.example.com hello
```

3. From Xcode, open the generated project `hello/hello.xcodeproj`.
The `cordova-lib` project will be used to update the Hybrid Web Container (HWC) project.
4. Open the HWC template project. In the HWC project's build setting, "Linking\Other Link Flag" section, delete all references to `cordova-lib.a`.

5. From Finder, delete the `Cordova-lib` folder under the HWC template project folder.
Delete all `cordova-lib.a` files from the HWC project's `lib` folder.
6. In the Xcode HWC project, add the `cordova-lib` project created in the `hello` project as a dependent project. Update the HWC project's setting to include the `cordova` project in the HWC's "Target Dependencies" and "Link Binary With Libraries" section.

7. In the sub project `cordova-lib` Navigator, find and open the `CDVViewController.h` file located in the `Classes/Cleaver` group folder (Make sure only the `cordova-lib` project opens in HWC Xcode, otherwise these files are invisible in Navigator).

8. Add a `UIViewController` property declaration:

```
@property (nonatomic, strong) UIViewController* viewController;
```

9. Add these function declarations in `CDVViewController`:

```
-(void)setTheWebView: (UIWebView*) theWebView;  
-(void)setTheViewController: (UIViewController*)  
theViewController;
```

10. In the Xcode Project Navigator, find and open the `CDVViewController.m` file, synthesizing the `viewController` property:

```
@synthesize viewController = _viewController;
```

11. Use `#if 0` to comment out these portions of the `viewDidLoad` function in the `CDVViewController.m` file:

```
#if 0  
NSURL* appURL = nil;  
NSString* loadErr = nil;  
  
if ([self.startPage rangeOfString:@"://"].location != NSNotFound)  
{ appURL = [NSURL URLWithString:self.startPage]; } else if  
([self.wwwFolderName rangeOfString:@"://"].location !=  
NSNotFound) { appURL = [NSURL URLWithString:[NSString  
stringWithFormat:@"%s/%s", self.wwwFolderName,  
self.startPage]]; } else {  
NSString* startFilePath = [self.commandDelegate  
pathForResource:self.startPage];  
}
```

```

if (startFilePath == nil) { loadErr = [NSString
stringWithFormat:@"ERROR: Start Page at '%@/%@"' was not found.",
self.wwwFolderName, self.startPage]; NSLog(@"%@", loadErr);
self.loadFromString = YES; appURL = nil; } else { appURL = [NSURL
fileURLWithPath:startFilePath]; }
}
#endif
...

#if 0
if (!loadErr) { NSURLRequest* appReq = [NSURLRequest
requestWithURL:appURL
cachePolicy:NSURLRequestUseProtocolCachePolicy timeoutInterval:
20.0]; [self.webView loadRequest:appReq]; } else { NSString* html
= [NSString stringWithFormat:@" %@", loadErr]; [self.webView
loadHTMLString:html baseURL:nil]; }
#endif

```

- 12.** Update the two `registerPlugin` functions in the `CDVViewController.m` file by replacing:

```
[plugin setViewController:self];
```

with:

```
[plugin setViewController: self.viewController];
```

- 13.** Add the implementations for the two new methods mentioned previously in the `CDVViewController.m` file:

```

-(void) setTheWebView: (UIWebView*) theWebView { self.webView =
theWebView; }

-(void) setTheViewController: (UIViewController
*)theViewController { self.viewController =
(CDVViewController*)theViewController; }

```

- 14.** Add this line to the `dealloc` function in the `CDVViewController.m` file:

```
self.whitelist = nil;
```

- 15.** Use `#if 0` to comment out the `CreateGapView` function implementation from the `CDVViewController.m` file.

```

-(void) createGapView
{
#if 0
CGRect webViewBounds = self.view.bounds;

webViewBounds.origin = self.view.bounds.origin;

if (!self.webView) { self.webView = [self
newCordovaViewWithFrame:webViewBounds];
self.webView.autoresizingMask = (UIViewAutoresizingFlexibleWidth
|
UIViewAutoresizingFlexibleHeight); [self.view
addSubview:self.webView];
[self.view sendSubviewToBack:self.webView]; _webViewDelegate =
[[CDVWebViewDelegate alloc] initWithDelegate:self];
self.webView.delegate = _webViewDelegate; #endif

```

```
// register this viewController with the NSURLProtocol, only after
the User-Agent is set
[CDVURLProtocol registerViewController:self]; #if 0 }
#endif
}
```

16. In the Xcode Project Navigator, find and open the `CDVPlugin.m` file, which is in the Classes/Commands group folder, and add this code at the very top of the `dealloc` function:

```
self.viewController = nil;
```

17. At the beginning of `initializeAppAfterKeyVaultUnlocked` method in the `HWCAppDelegate.m` file, call:

```
[super viewDidLoad];
```

18. Add `Jsonkit` back into the new Cordova library to avoid link errors. You can copy the `jsonkit.h` and `jsonkit.m` files from the PhoneGap 2.0.0 library project. Beginning with Cordova 2.4.0, the `JsonKit` is no longer included in cordova library, however, `HWCLib.a` uses some functions provided by `Jsonkit`.
19. Add `Jsonkit.h` and `Jsonkit.m` as source files in the Cordova library project.
20. Set the compile flag “-fno-objc-arc” for the `jsonkit.m` file in the “build phase”/“compile source” section. The new cordova library project is compiled with ARC enabled, but `jsonkit.m` does not support ARC.
21. Add `jsonkit.h` as a public header file in the Xcode cordovalib project’s “build phase”/“copy header”/“public file” section, so it can be found by the HWC project.
22. Delete `cordova.plist` from hwc project, and add `config.xml` created for the Hello project into the HWC project. Beginning with Cordova 2.3.0, `config.xml` replaces `cordova.plist`.
23. Delete `www` and `capture.bundle` in the HWC project to avoid build errors. Replace the file `VERSION` with the one created by the new Hello project.
24. Edit the `config.xml` file in the HWC project to enable `httpproxy` and `applog` plugin by adding these lines:

```
<feature name="AppLog">
  <param name="ios-package" value="AppLogPlugin"/>
</feature>
<feature name="HttpsProxy">
  <param name="ios-package" value="HttpsProxyPlugin"/>
</feature>
```

25. Replace this code in the `HWCAppDelegate.m` file in the HWC project:

```
#ifdef CORDOVA_FRAMEWORK
#import <Cordova/CDVViewController.h>
#import <Cordova/CDVContacts.h>
#else
#import "CDVViewController.h"
#import "CDVContacts.h"
#endif
```



```
#endif
```

with:

```
#import <Cordova/CDVViewController.h>
#import <Cordova/CDVContacts.h>
```

26. Add the `AssetsLibrary`, `CoreMotion` and `imageIO` frameworks to the “build phase”/“Link Binary With Libraries” section in the HWC project to avoid link errors.

27. Delete this line from the `initializeAppAfterKeyVaultUnlocked` function in the `HWCAppDelegate.m` file:

```
[CDVContacts setContactsAccessDelegate:self];
```

28. Delete these lines from the `HWCAppDelegate.m` file:

```
static BOOL s_bContactsChallengeInProgress = NO;

+ (BOOL) isContactsChallengeInProgress

{ return s_bContactsChallengeInProgress; }

(void) requestContactsAccess {
    #if __IPHONE_OS_VERSION_MAX_ALLOWED >= 60000
    NSArray* versionCompatibility = [[UIDevice
currentDevice].systemVersion componentsSeparatedByString:@"."];
    NSInteger iMajorVersion = [[versionCompatibility objectAtIndex:
0] intValue];
    if (iMajorVersion >= 6)
    {
        if (ABAddressBookGetAuthorizationStatus() ==
kABAuthorizationStatusNotDetermined)
        {
            s_bContactsChallengeInProgress = YES;
            ABAddressBookRef addressBookRef =
            ABAddressBookCreateWithOptions(NULL, NULL);
            ABAddressBookRequestAccessWithCompletion(addressBookRef, ^(bool
granted, CFErrorRef error)
            { s_bContactsChallengeInProgress = NO; });
            CFRelease(addressBookRef);
        }
    }

    while (s_bContactsChallengeInProgress)

    { NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
    [[NSRunLoop currentRunLoop] runUntilDate:[NSDate
dateWithTimeIntervalSinceNow:1]]; [pool release]; }
    }
#endif
}
```

29. Delete this line in the `applicationWillResignActive` method from the `HWCAppDelegate.m` file:

```
bPresentingContactsChallenge = [HWCAppDelegate
isContactsChallengeInProgress];
```

30. Delete this line from the `HWCAppDelegate.h` file:

```
(void) requestContactsAccess;
```

31. Delete the protocol “MissingFeaturesProvider” from the CDVViewController interface definition from the `HWCAppDeleage.h` file to avoid build errors.
32. Update `cordova.js` to the one provided with the new cordova library if `cordova.js` is used in your `HybridApp.js` app.

Windows Mobile Hybrid Web Container Customization

Customize the look and feel and default behavior of the Windows Mobile Hybrid Web Container.

Before getting started, build the Hybrid Web Container project in Visual Studio, as described in *Building the Windows Mobile Hybrid Web Container Using the Provided Source Code*. In Solution Explorer, the `HybridWebContainer` directory contains directories such as `libs`, as well as `images` and other files.

The `HybridWebContainer` solution includes a set of sample files that you can include in your project. After modifying the code in the sample files, rebuild your project: to preserve your changes in the generated code. Always test your changes before using the resulting application.

In the `HybridWebContainer` project, the `docs` directory includes JavaDoc documentation for applications in `com.sybase.hwc`, and the library in `com.sybase.hybridApp`.

Windows Mobile Customization Touch Points

Touch points for Hybrid Web Container customizations are indicated in code by comments of the form `WM_CUSTOMIZATION_POINT_ customization`.

Touch Point	Description
<code>WM_CUSTOMIZATION_POINT_BRAND</code>	Change application name, copyright, and developer information in the About form.
<code>WM_CUSTOMIZATION_POINT_HYBRID-APPSEARCH</code>	Make the list of Hybrid App packages searchable.
<code>WM_CUSTOMIZATION_POINT_HYBRID-APPLIST</code>	Change the appearance of the Hybrid App package list.
<code>WM_CUSTOMIZATION_POINT_CATEGORIZEDVIEWS</code>	Create categorized views of the Hybrid App packages.
<code>WM_CUSTOMIZATION_POINT_HYBRID-APPSORTING</code>	Customize the criteria for sorting the Hybrid App package list.

Touch Point	Description
WM_CUSTOMIZATION_POINT_MESSAGE-SORTING	Customize the criteria for sorting the message list.
WM_CUSTOMIZATION_POINT_MESSAGE-FILTERING	Change the filter used to sort the list of messages.
WM_CUSTOMIZATION_POINT_ANONYMOUS_USER	Indicates if the login mode is anonymous.
WM_CUSTOMIZATION_POINT_DEFAULT-SETTINGS	Change default server settings.
WM_CUSTOMIZATION_POINT_PRESET-SETTINGS	Hard-code settings for the Settings screen so they do not appear on the device. This prevents the user from changing the settings.
WM_CUSTOMIZATION_POINT_HTTPHEADERS	Set HTTPS headers for the Windows Mobile Hybrid Web Container to include authentication tokens.
WM_CUSTOMIZATION_POINT_HTTPERRORHANDLERS	Change the handling of HTTP errors.
WM_CUSTOMIZATION_POINT_TOKENERROR	Change how the client engine handles authentication token errors (for example, when a token expires).

Look and Feel Customization of the Windows Mobile Hybrid Web Container

Customizations you can make to the look and feel include changing the splash screen, changing the Hybrid App icons and name, changing the Hybrid App package icons, changing labels and text, adding support for new languages, and so on.

Changing the Hybrid Web Container Icon

Replace the icon shown on the home screen.

Changing the container icon also changes the image used on the About screen, and the image that sometimes shows up in the title bar.

1. In Solution Explorer, navigate to `HybridWebContainer\Resources\Images`.
2. Replace the `icon.ico` file with your version.

The new image must use the same name and extension as the original file, and the same resolution.

3. Rebuild and test the project.

Changing the Windows Mobile Hybrid App Package Icon

Modify the Hybrid App package application icon.

You cannot add new icons to the folder, but you can replace the existing icon images, using the same file name. The Hybrid App application icons are named `ampiconindex.png`, where *index* is a number between 30 and 116. The default Hybrid App icon is `ampicon48.png`. This is also the icon shown on the menu item that lists all the Hybrid Apps.

Each Hybrid App icon uses a pair of associated images:

- **`ampiconindexp.png`** – represents a processed message (indicated by the *p* suffix). Processed means the message has been submitted to the server.
- **`ampiconindex.png`** – is for unprocessed messages, which have not been submitted to the server.

1. Identify the image currently used by the Hybrid App package that you want to replace.

When you build the Hybrid Web Container with custom icons, the original icons still appear in SAP Control Center and in SAP Mobile WorkSpace.

2. In Solution Explorer, navigate to the `HybridWebContainer\Resources\Images` folder.
3. Replace the `ampiconindex.png` and `ampiconindexp.png` image files with the new images.

Note: For each icon file that you replace, use the same name, extension, and resolution as the original. To preserve the original image make a copy of it. To prevent the copy from interfering with resource indexing, place it in a different folder.

4. Rebuild and test the Hybrid Web Container.

Implementing a Custom HybridAppList Screen

Add a custom `HybridAppList` screen.

Use the `CustomCode` sample files as the starting point for your customization.

1. In Visual Studio Solution Explorer, click the **Show All** button.
2. Include all the files in the **CustomCode** folder.
3. Modify the code in your copy of the included files.

You can modify these files to customize the `HybridAppList` screen:

- **`MyHybridAppListScreen`** – class used to implement the `HybridAppList` screen.
- **`HybridAppComparer`** – comparer used by `MyHybridAppListScreen` to sort the Hybrid Application order.
- **`HybridAppFilter`** – filter used by `MyHybridAppListScreen` to filter the Hybrid App.
- **`CustomizationHelper`** – class that integrates the `HybridAppListScreen` into the Hybrid Web Container.

4. Rebuild and test your project.

Customizing the About Screen and Other Branding

Customize the About screen.

1. In Solution Explorer, click the **Show All** button.
2. Include all the files in the **CustomCode** folder.
3. Modify the code in your copy of the included files.

Code related to this customization is:

```
public override void ShowAboutForm()
{
    System.Text.StringBuilder _sb = new
System.Text.StringBuilder();
    _sb.Append("Copyright 2012 Esabys, Inc.");
    _sb.Append("\r\n");
    _sb.Append("Version: 1.0"); _
    sb.Append("\r\n");
    sb.Append("Build id:20120518-0123");
    MessageBox.Show(_sb.ToString(), Consts.APP_TITLE,
    MessageBoxButtons.OK,
    MessageBoxIcon.Asterisk, MessageBoxDefaultButton.Button1);
}
```

4. Rebuild and test your project.

Adding a Splash Screen

Add a splash screen to the Hybrid Web Container.

1. In Visual Studio Solution Explorer, click the **Show All** button.
2. Include all the files in the **CustomCode** folder.
3. Modify the code in your copy of the included files.
 - **SplashForm** – class used to implement the Splash screen. It starts a timer to show the splash image in about one second.
 - **SplashBitmap.png** – image shown in the splash screen.
 - **CustomRes.resx** – resource file that contains the image file.
 - **CustomizationHelper** – class that integrates the Splash Screen into the Hybrid Web Container. When the application starts, CustomizationHelper displays the splash screen.
4. Rebuild and test your project.

Changing Labels and Text

You can customize most of the text found in labels, dialogs, or error messages used by the Hybrid Web Container.

1. In your project, open `HybridWebContainer\strings.resx` for editing.

Hybrid Web Container Customization

This file contains the text for error messages, screen titles, screen labels, validation messages, and so on.

2. Make your changes to `strings.res` and save the file.

Note: Make the same changes for each language to which you translate your text. Edit the `Strings.xx.res` file, where `xx` is the ISO639 code for the language (for example, `it` for Italian).

Adding a New Language

Add support for a new language to the Hybrid Web Container.

1. In Solution Explorer, create a new subfolder under `HybridWebContainer` \Resources named `Strings.xx.res`, where `xx` is the ISO639 code for the language (for example, `it` for Italian).

2. Add a file called `Strings.xx.res` to the new folder.

You can copy the default `Strings.res` file from `HybridWebContainer` \Resources\Strings, and use the copy as a template for the new `Strings.xx.res` file.

3. In the language-specific `Strings.xx.res` file, add your translated text.

You need not include strings that do not require localization. Any strings that are omitted from localization are removed from the default `Strings.res` file.

Default Behavior Customization of the Windows Mobile Hybrid Web Container

You can add or remove screens from the Hybrid Web Container, and change the behavior, such as sorting and filtering of messages.

Customizing Settings Screen Fields

Hide fields in the Settings screen or change their default values.

1. In Visual Studio, open the `CustomizationHelper` class in the `CustomCode` folder.
2. Override the `DefaultServerSettings` method.
3. Initialize the default server settings and return them outside of the `DefaultServerSettings` method.
4. For each field you want to remove from the Settings screen, set its value to value to null. In this example, the server name field is visible but no default value is assigned; the server port is set to 5001 but the field is hidden:

```
public override ServerSettings DefaultServerSettings
{
    get
    {
        if (m_ServerSettings == null)
```

```

{
    m_ServerSettings = new ServerSettings();

    // Server name will be shown and initialized as empty.
    m_ServerSettings.ServerName.IsVisible = true;
    m_ServerSettings.ServerName.HasValue = false;

    // Server port will NOT be shown and initialized as 5001.
    m_ServerSettings.ServerPort.IsVisible = false;
    m_ServerSettings.ServerPort.HasValue = true;
    m_ServerSettings.ServerPort.Value = 5001;

    // Other fields will be shown.
}
return m_ServerSettings;
}
}
private ServerSettings m_ServerSettings;

```

Notes:

- By default, all fields are shown.
- To hide a field, set its `IsVisible` property to “false”.
- To change a field's initial value, set `HasValue` to “true”, and specify a value in the `Value` property.

Using Multiple Hybrid Web Containers on the Same Windows Mobile Device

You can configure two or more Hybrid Web Containers on a Windows Mobile device.

Each container can be installed separately on the same device, can connect to a different server, and can be used independently.

1. Create a Visual Studio project for each container.
2. For each container, edit the project's `config.properties` file and specify a unique `AppID` property for your container.
For example: `AppID="HWC1"`.

Note: Do not change the `AppID` property at runtime.

3. Rebuild the project, as described in *Building the Windows Mobile Hybrid Web Container Using the Provided Source Code*.
4. Configure the container's CAB build. In each project, edit the `OneBridge_ppc.inf` file and customize these properties:

AppName – provide a unique name for each container.

InstallDir – enter the path where the container is to be installed on the device. Each container must have a different path.

Shortcuts – declare a shortcut that launches the container application. Users can change shortcut names. Shortcut names do not have to be unique.

Here are sample customized lines in `OneBridge_ppc.inf`:

```
[CEStrings]
AppName = "HWC"
InstallDir=%CE1%\Sybase\%AppName%
...
[Shortcuts.All]
Hybrid Web Container,0,HWCA.exe,%CE11%
```

5. Build the CAB file for each container, as described in *Packaging a CAB File*.

Sorting the List of Hybrid App Packages

Change the default sorting of the list of Hybrid App packages.

By default, the Hybrid Web Container displays Hybrid App package names in alphabetical order. This example changes the list to sort case-sensitively

1. Add a `HybridWebAppComparer` class that uses the base class `IComparer<HybridWebAppInfo>`.

2. Override the `Compare` method using:

```
public int Compare(HybridWebAppInfo x, HybridWebAppInfo y)
{
    return string.Compare(x.DisplayName, y.DisplayName, false);
}
```

3. Open the `CustomizationHelper` class in the `CustomCode` folder.

4. Override the `HybridAppComparator` method using:

```
public override IComparer<HybridWebAppInfo> HybridAppComparator
{
    get { return new HybridWebAppComparer(); }
}
```

5. Save the file.

Sorting Hybrid App Messages

Sort Hybrid App messages based on different criteria.

1. Add a `MessageComparer` class that uses the base class `IComparer<Message>`.
2. Override the `Compare` method using this code:

```
public int Compare(Message x, Message y)
{
    int iModuleId1 = x.ModuleId;
    int iModuleId2 = y.ModuleId;

    int iCompareResult = 0;
    if (iModuleId1 < iModuleId2)
    {
        iCompareResult = -1;
    }
    if (iModuleId1 > iModuleId2)
    {
        iCompareResult = 1;
    }
    if (iCompareResult == 0)
```



```

    {
        iCompareResult = x.ReceiveDate.CompareTo(y.ReceiveDate);
    }
    return iCompareResult;
}

```

3. Open the CustomizationHelper class in the CustomCode folder.
4. Override the MessageComparator using:

```

public override IComparer<Message> MessageComparator
{
    get { return new MessageComparer(); }
}

```

5. Save the file.

Filtering Hybrid App Messages

Prevent the Hybrid App from displaying some messages.

1. Add a MessageFilter class that uses the base class IFilter<Message>.
2. Override the select method using code similar to:

```

public bool Select(Message subject)
{
    if (subject.Priority ==
MessageConsts.EMAIL_STATUS_IMPORTANCE_HIGH)
    {
        return false;
    }
    return true;
}

```

3. Open the CustomizationHelper class in the CustomCode folder.
4. Override the MessageFilter method using:

```

public override IFilter<Message> MessageFilter
{
    get
    {
        return new MessageFilter();
    }
}

```

5. Save the file.

Setting HTTP Headers

Set HTTP headers for the Hybrid Web Container to include authentication tokens.

These methods in the Hybrid Web Container template source code show how to set HTTP headers:

- **getHttpHeaders** – override this method to set the authentication tokens.
- **OnHTTPError** – listener called by the communication layer when an HTTP error occurs.

- **OnTokenError** – listener called by the client engine when Hybrid App token authentication failure occurs.
1. In Visual Studio, open the `CustomizationHelper` class in the `CustomCode` folder.
 2. Override the `getHttpHeaders` method and uncomment its contents.
The stub code shows how to add headers and cookies. Simply replace the header and cookie assignments with your own.
 3. Refresh the HTTP headers.
It is a good idea to refresh the HTTP headers in the `OnTokenError` method, which is called when a Hybrid App token authentication failure occurs.
Here is a common way to do this:
 - a. Maintain member variables that contain the values for the headers you want to set.
 - b. Override the `GetHttpHeaders` method to use the value in those member variables when it sets the headers.
 - c. In `OnTokenError`, update the member variables with the new header values.
 - d. Call `UpdateHttpHeaders` again.
 4. If you have custom code to run when an HTTP error occurs, add the code to override the `OnHTTPErrors` method.
Your method is called any time there is an HTTP error. You can use it to inform the user of errors, or to perform other custom steps in response to particular error codes.

Customizing OK Button Behavior

Control behavior when the OK button is clicked in Hybrid App forms.

To customize the OK button in the `MessageList`, `ApplicationList`, and `Application` forms, override the `OnClosing` methods for those forms:

```
internal virtual void OnClosingMessageListForm( MessageListForm
form )
{
}
```

```
internal virtual void
OnClosingApplicationListForm( HybridWebAppListForm form )
{
}
```

```
internal virtual void OnClosingHybridAppForm( HybridWebAppForm
form )
{
}
```

Packaging a CAB File

After rebuilding your customized Hybrid Web Container, package the generated files into a cab file that can be installed on a device.

Prerequisites

Install ActivePerl, available for download from <http://www.activestate.com/>. After installing ActivePerl, add it to the environment path. When you run Perl at the command prompt, the script is executed by the first Perl.exe it encounters in the list of paths in the PATH environment variable. To ensure the script is executed by the correct Perl interpreter, specify the complete path to the Perl.exe you want to use.

Task

When you build the template project, the binary release files are generated into the template output folder.

1. Open a Command Prompt.
2. In the Command Prompt, navigate to the `template\Tools` folder of your project.
3. Run the `buildcab` script, specifying the path to the location of the release files generated when you built the project.

For example:

```
perl buildcab.pl ..\bin\Release
```

The packaged CAB file is generated in `template\Tools`.

Prepackaged Hybrid Apps

You can use the Hybrid Web Container as the runtime shell for a single Hybrid App.

When you use the prepackaged Hybrid App, the application is launched immediately and there is no listview of Hybrid Apps. This allows for a single view of the Hybrid App. You can still assign other applications to the Hybrid Web Container, but while running in this new mode, only the Hybrid App designated as the default is active.

Note: Connection settings for the Hybrid Web Container must be configured before the prepackaged Hybrid Web Container can launch.

When the user closes the default Hybrid App, he or she can then view the messages associated with that application in the Hybrid Web Container.

Including a Prepackaged Hybrid App in the Android Hybrid Web Container

Run a prepackaged Hybrid App so that the Hybrid Web Container functions as a single-purpose application rather than a general purpose one.

1. Package the Hybrid App files.

You can use a Hybrid App that was generated with the Hybrid App Designer, or you can use the packaging tool to generate a new Hybrid App.

When packaging the Hybrid App, optimize the size by generating a version for each specific platform that includes only files for that platform.

See *Packaging Hybrid Apps Using the Packaging Tool*.

2. Copy the Generated Hybrid App folder under the package tool workspace, or copy the Generated Hybrid App folder under the SAP Mobile WorkSpace, to the `assets` directory of the Android Hybrid Web Container template.

3. Remove the ZIP file from the folder.

4. Refresh the Eclipse workspace.

5. Open the `CustomizationHelper.java` file, locate the `ANDROID_CUSTOMIZATION_POINT_PREPACKAGED_APP` customization point that accompanies the `getPrepackageAppPath` function, and change the contents of this function to return the name of the top-level directory you just added to the project.

If the prepackaged Hybrid App manages the server connection by itself and wants to exit the Hybrid Web Container after exiting the prepackaged Hybrid App, change return value of the method `exitHWConPrepackagedAppClose` to `true`.

6. To optionally enable the Hybrid Web Container to exit after closing the prepackaged Hybrid App, change the return value of the `exitHWConPrepackagedAppClose` method to `true`.

The default return value of the method is `false`.

Including a Prepackaged Hybrid App in the BlackBerry Hybrid Web Container

Run a prepackaged so that the Hybrid Web Container functions as a single-purpose application rather than a general purpose one.

Prerequisites

Install the BlackBerry Java Plug-in for Eclipse.

Task

1. Package the Hybrid App files.

You can use a Hybrid App that was generated with the Hybrid App Designer, or you can use the packaging tool to generate a new Hybrid App.

When packaging the Hybrid App, optimize the size by generating a version for each specific platform that includes only files for that platform.

See *Packaging Hybrid Apps Using the Packaging Tool*.

2. In Eclipse, import the BlackBerry Hybrid Web Container template as a legacy BlackBerry project:

- a) Select **File > Import**.

- b) Expand the **BlackBerry** folder.

- c) Select **Import Legacy BlackBerry Projects**.

- d) Click **Next**.

- e) Specify the JRE and, in the BlackBerry Workspace field, browse to the `HWCtemplate.jdw` file and select the project to import.

- f) Select **Copy BlackBerry projects into workspace** to create a copy of the imported project in the Eclipse workspace.

- g) Click **Finish**.

3. Copy the generated Hybrid App folder under the package tool workspace to the `res` directory of the imported Eclipse BlackBerry Hybrid Web Container project.

4. Remove the ZIP file from the folder, and refresh the Eclipse workspace.

5. Open the `CustomizationHelper.java` file for editing.

6. Find the `BLACKBERRY_CUSTOMIZATION_POINT_PREPACKAGE_APP` that accompanies the `getPrepackagedAppPath` function, and change the contents of the function to return the name of the top-level directory you just added to the project.

If the prepackaged Hybrid App manages the server connection by itself and wants to exit the Hybrid Web Container after exiting the prepackaged Hybrid App, change return value of the method `exitHWCOnPrepackagedAppClose` to `true`.

7. Save the `CustomizationHelper.java` file.

Including a Prepackaged Hybrid App in the iOS Hybrid Web Container

Run a prepackaged Hybrid App in the iOS Hybrid Web Container so that the Hybrid Web Container functions as a single-purpose application rather than a general purpose one.

1. Package the Hybrid App files.

You can use a Hybrid App that was generated with the Hybrid App Designer, or you can use the packaging tool to generate a new Hybrid App.

When packaging the Hybrid App, optimize the size by generating a version for each specific platform that includes only files for that platform.

See Packaging Hybrid Apps Using the Packaging Tool.

2. Copy the generated Hybrid App folder to a location that is accessible to your Xcode project.
3. In the Xcode Project Navigator, right-click the **Resources** group folder, and select **Add Files to HWC**.
4. Navigate to the directory you just created that contains the generated package, and select the top-level directory of the package.
Create folder references, not group references, when you add the files. The directories appear directly under `Resources`.
5. In the Project Navigator, find and open the `CustomizationHelper.m` file, which is located in the `Classes` group folder.
6. Locate the customization point, designated by the comment `IOS_CUSTOMIZATION_POINT_PREPACKAGED_APP`, that accompanies the `getPrepackagedAppPath` function, and change the contents of this function to return the name of the top-level directory you just added to the project.
7. Rebuild the `HWC.xcodeproj` project.
 - a) From the Xcode menu, select **Product > Clean**.
 - b) Select **Product > Build**.

Including a Prepackaged Hybrid App in the Windows Mobile Hybrid Web Container

Run a prepackaged so that the Hybrid Web Container functions as a single-purpose application rather than a general purpose one.

1. Package the Hybrid App files.

You can use a Hybrid App that was generated with the Hybrid App Designer, or you can use the packaging tool to generate a new Hybrid App.

When packaging the Hybrid App, optimize the size by generating a version for each specific platform that includes only files for that platform.

See Packaging Hybrid Apps Using the Packaging Tool.

2. Include the generated Hybrid App files in a Visual Studio project:
 - a) Copy the generated Hybrid App files to your Visual Studio project.
 - b) Open the `HybridWebContainer.csproj`, which is in the `WM_HWC<version>.zip` file.
 - c) In Visual Studio Solution Explorer, select **Show All Files**.
 - d) Right-click the Hybrid App folder and select **Include in Project**.
 - e) Set the **Copy to Output Directory** property to **Copy if newer** for all the files under this folder.

Note: You can select all the files using the SHIFT CTRL keys, and then set the property for all the selected files.

3. In the CustomCode folder, create a Partial class for CustomizationHelper.cs.
4. In the Partial class of the CustomizationHelper.cs file, create a method to override the property PrepackageAppPath to return the full installation path of the Hybrid App on the device.

```
public override string PrepackageAppPath
{
    get
    {
        return @"Program Files\sybase\hwc\iMOWebProto";
    }
}
```

5. Rebuild the project.
6. Include the prepackaged Hybrid App in a CAB file:

Most Windows Mobile applications are deployed as CAB files. You can find information about creating CAB files at <http://msdn.microsoft.com/en-us/library/aa448616.aspx> and information about the .inf file at <http://msdn.microsoft.com/en-us/library/aa448654.aspx>.

 - a) Open the onebridge_ppc.inf file, which is located in the Tools folder of the Hybrid Web Container template project.
 - b) Add the prepackaged Hybrid App folders in the [SourceDisksNames.ARM] section:

```
[SourceDisksNames.ARM]
1="PPC",,unsigned
3="zh-CN",,"unsigned\zh_CN"
4="zh-HK",,"unsigned\zh_HK"
5="de",,"unsigned\de"
6="fr",,"unsigned\fr"
7="fr-CA",,"unsigned\fr_CA"
8="ja",,"unsigned\ja"
9="es",,"unsigned\es"
10="prepackage",,"unsigned\prepackage"
11="prepackage.css",,"unsigned\prepackage\html\css"
12="prepackage.default",,"unsigned\prepackage\html\default"
13="prepackage.en",,"unsigned\prepackage\html\en"
14="prepackage.en_US",,"unsigned\prepackage\html\en_US"
15="prepackage.icon",,"unsigned\prepackage\html\icon"
16="prepackage.images",,"unsigned\prepackage\html\images"
17="prepackage.js",,"unsigned\prepackage\html\js"
18="prepackage.html",,"unsigned\prepackage\html"
```

- c) List all the required files in the [SourceDisksFiles.ARM] section:

```
[SourceDisksFiles.ARM]
CMessagingClient.2.2.0.dll=1
OBSetup.dll=1
HWCA.exe=1
HWCEngine.lnk=1
```

```
Plugins.xml=1
; other files
hybridapplib.dll=1
SQLite.Interop.DLL=1
System.Data.SQLite.dll=1
version.txt=1
config.properties=1
WorkflowClient.xml=10
index.xml=10
manifest.xml=10
"Stylesheet.css"=11
"hybridapp.html"=12
"hybridapp.html"=13
"hybridapp.html"=14
"API.js"=17
"Callbacks.js"=17
```

- d) Define the installation target in the [DestinationDirs] section:

```
[DestinationDirs]
Files.ARM = 0,%InstallDir%
Shortcuts.All = 0,%CE4%
System.ARM = 0,%CE2%
zh-CN = 0,%InstallDir%\zh-CN
zh-HK = 0,%InstallDir%\zh-HK
de = 0,%InstallDir%\de
fr = 0,%InstallDir%\fr
fr-CA = 0,%InstallDir%\fr-CA
ja = 0,%InstallDir%\ja
es = 0,%InstallDir%\es
prepackage.css = 0,"%InstallDir%\prepackage\html\css"
prepackage.default = 0,"%InstallDir%\prepackage\html\default"
prepackage.en = 0,"%InstallDir%\prepackage\html\en"
prepackage.en_US = 0,"%InstallDir%\prepackage\html\en_US"
prepackage.icon = 0,"%InstallDir%\prepackage\html\icon"
prepackage.images = 0,"%InstallDir%\prepackage\html\images"
prepackage.js = 0,"%InstallDir%\prepackage\html\js"
prepackage.html = 0,"%InstallDir%\prepackage"
prepackage = 0,"%InstallDir%\prepackage"
```

- e) Describe each file mapping in the File List section:

```
[prepackage.css]
Stylesheet.css,,0

[prepackage.default]
hybridapp.html,,0

[prepackage.en]
"hybridapp.html"

[prepackage.en_US]
"hybridapp.html"

[prepackage.icon]

[prepackage.images]
```



```

[prepackage.js]
"API.js"
"Callbacks.js"
"Camera.js"
"Certificate.js"
"Custom.js"
"dataajs-1.0.2.js"
"ExternalResource.js"
"json2.js"
"MAKit.js"
"Resources.js"
"SUP0.js"
"SUPStorage.js"
"Timezone.js"
"Utils.js"
"HybridApp.js"
"WorkflowMessage.js"

[prepackage]
"index.xml"
"manifest.xml"
"WorkflowClient.xml"

[prepackage.html]
hybridapp.html

[System.ARM]
manifest.xml,,0

```

- f) Include all the file lists in section [DefaultInstall.ARM]:

```

[DefaultInstall.ARM]
CopyFiles=Files.ARM, System.ARM, de, fr, fr-CA, es, zh-CN, zh-
HK, ja, prepackage,
prepackage.css, prepackage.default, prepackage.en, prepackage.en_
US,
prepackage.icon, prepackage.images, prepackage.js, prepackage.htm
l

```

- g) Run: buildcab.pl <Path to project output>.

7. Deploy and run the customized Hybrid Web Container on the device or emulator.
 - a) Compile the Hybrid Web Container.
 - b) Deploy the Hybrid Web Container to the device or emulator.
 - c) Run and test the prepackaged Hybrid App.

Adding Native Device Functionality to the Hybrid Web Container

PhoneGap (now known as Apache Cordova) is an open source framework that leverages Web technologies such as HTML, CSS, and JavaScript to access native (system and third-party) functionality across platforms.

SAP Mobile Platform comes with the Cordova libraries, which handle common tasks supported by most devices, linked in and ready to use. Integrating PhoneGap plug-ins with Hybrid Web Containers allows you to extend the set of APIs available within a Hybrid App. See www.phonegap.com for information about the supported PhoneGap APIs.

PhoneGap API calls are made from the Hybrid App JavaScript files.

Supported JavaScript PhoneGap APIs

The Hybrid Web Container comes with the PhoneGap library linked in and ready to use.

The PhoneGap library included with SAP Mobile Platform handles common native tasks supported by Android, BlackBerry, iOS and Windows Mobile devices, for example, accessing geolocation, accessing contacts, and invoking calls to make those common functions available to JavaScript.

Note: Keep in mind that PhoneGap APIs cannot be accessed successfully until initialization has taken place. If you make calls to the PhoneGap API from the `customAfterShowScreen` function, they should occur only after the PhoneGap subsystem is initialized and ready to execute these calls. For more information, see <http://wiki.phonegap.com/w/page/36868306/UI%20Development%20using%20jQueryMobile#HandlingPhoneGapsdevicereadyevent>.

You can make PhoneGap calls from the Hybrid Web Container JavaScript, such as `Custom.js`. For example, to save an entry to the contacts database, you can implement something similar to:

```
var contact = navigator.contacts.create();
    contact.nickname = "Plumber";
    var name = new ContactName();
    name.givenName = "Jane";
    name.familyName = "Doe";
    contact.name = name;
    // save
    contact.save(onSaveSuccess, onSaveError);
```

You can use both Hybrid Web Container JavaScript APIs and PhoneGap APIs in a single application. For information about PhoneGap APIs, see <http://docs.phonegap.com>.

Table 9. PhoneGap Supported Features

API	Object and Function	Platform
Accelerometer		
	accelerometer <ul style="list-style-type: none"> • <code>getCurrentAcceleration</code> <hr/> Note: On iOS, this function must be called after <code>watchAcceleration</code> . <ul style="list-style-type: none"> • <code>watchAcceleration</code> • <code>clearWatch</code> 	<ul style="list-style-type: none"> • Android • iOS • BlackBerry
	Acceleration <ul style="list-style-type: none"> • <code>x</code> • <code>y</code> • <code>z</code> • <code>timeStamp</code> 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS
Camera		
	Camera <ul style="list-style-type: none"> • <code>getPicture (Camera.PictureSourceType.CAMERA)</code> • <code>getPicture (Camera.PictureSourceType.PHOTOLIBRARY)</code> • <code>getPicture (Camera.PictureSourceType.SAVEDPHOTOALBUM)</code> 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile
	CameraOptions <ul style="list-style-type: none"> • <code>quality</code> • <code>dedestinationType.DATA_URL</code> • <code>dedestinationType.FILE_URI</code> FILE_URI is the default. • <code>allowEdit</code> • <code>encodingType</code> • <code>targetWidth</code> • <code>targetHeight</code> 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile
Capture		

API	Object and Function	Platform
	<p>Capture</p> <ul style="list-style-type: none"> captureAudio <hr/> <p>Note: On Android, whether this works depends on which application the device uses to record the audio. You can use <code>media.record</code> instead to work around this issue.</p> <hr/> <ul style="list-style-type: none"> captureImage captureVideo 	<ul style="list-style-type: none"> Android BlackBerry iOS
	<p>MediaFile</p> <ul style="list-style-type: none"> getFormatData 	<ul style="list-style-type: none"> Android iOS
Compass		
	<p>compass</p> <ul style="list-style-type: none"> getCurrentHeading watchHeading clearWatch watchHeadingFilter 	<ul style="list-style-type: none"> Android iOS
	<p>compass.Heading</p> <ul style="list-style-type: none"> magneticHeading trueHeading headingAccuracy timestamp 	<ul style="list-style-type: none"> Android iOS
Connection		
	<p>network.connection.type</p>	<ul style="list-style-type: none"> Android BlackBerry iOS
Contacts		
	<p>contacts.create</p>	<ul style="list-style-type: none"> Android BlackBerry iOS Windows Mobile

API	Object and Function	Platform
	contacts.find	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile
	contact.clone	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile
	Contacts.remove <hr/> Note: On Android, there is an issue with contacts not being fully removed. See https://issues.apache.org/jira/browse/CB-75 . <hr/>	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile
	Contacts.save	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile
Device		
	Device.name	<ul style="list-style-type: none"> • Android • BlackBerry • iOS
	Device.phonegap	<ul style="list-style-type: none"> • Android • BlackBerry • iOS
	Device.platform	<ul style="list-style-type: none"> • Android • BlackBerry • iOS
	Device.uuid	<ul style="list-style-type: none"> • Android • BlackBerry • iOS

API	Object and Function	Platform
	Device.version	<ul style="list-style-type: none"> • Android • BlackBerry • iOS
Events		
	Deviceready	<ul style="list-style-type: none"> • Android • iOS
	Pause	<ul style="list-style-type: none"> • Android
	Resume	<ul style="list-style-type: none"> • Android
	Online	<ul style="list-style-type: none"> • Android • iOS
	Offline	<ul style="list-style-type: none"> • Android • iOS
	Batterycritical	iOS
	Batterylow	iOS
	Batterystatus	iOS
	<p>Note: On Android, PhoneGap 1.4.1, this does not work due to a known issue. See https://issues.apache.org/jira/browse/CB-173.</p>	
	Menubutton	<ul style="list-style-type: none"> • Android
	Searchbutton	<ul style="list-style-type: none"> • Android
File		

API	Object and Function	Platform
	DirectoryEntry <ul style="list-style-type: none"> • copyTo • moveTo • toURI • remove • removeRecursively • getParent • createReader • getDirectory • getFile 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile
	FileEntry <ul style="list-style-type: none"> • copyTo • moveTo • toURI • remove • getParent • createWriter • file 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile
	FileReader <ul style="list-style-type: none"> • abort • readAsDataURL • readAsText 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile
	FileWriter <ul style="list-style-type: none"> • abort • seek • truncate • write 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile
	DirectoryReader <ul style="list-style-type: none"> • readEntries 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile

API	Object and Function	Platform
	LocalFileSystem <ul style="list-style-type: none"> • requestFileSystem • resolveLocalFileSystemURI 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile
	FileTransfer <ul style="list-style-type: none"> • upload • download 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile
Geolocation		
	geolocation <ul style="list-style-type: none"> • getCurrentPosition <hr/> <p>Note: This function does not work on the Android Galaxy Tab P1000 device.</p> <hr/> <ul style="list-style-type: none"> • watchPosition • clearWatch 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile
	Position <ul style="list-style-type: none"> • coords • timestamp 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile

API	Object and Function	Platform
	<p>Coordinates</p> <ul style="list-style-type: none"> latitude longitude altitude accuracy <hr/> <p>Note: On Android, the returned accuracy property is always null.</p> <hr/> <ul style="list-style-type: none"> altitudeAccuracy <hr/> <p>Note: On Android, the returned altitudeAccuracy property is always null.</p> <hr/> <ul style="list-style-type: none"> heading <hr/> <p>Note: Android only. The returned heading property is always null.</p> <hr/> <ul style="list-style-type: none"> speed <hr/> <p>Note: On Android, the returned speed property is always null.</p>	<ul style="list-style-type: none"> Android BlackBerry iOS Windows Mobile
Media		
	Media.play	<ul style="list-style-type: none"> Android iOS Windows Mobile
	Media.pause	<ul style="list-style-type: none"> Android iOS
	Media.stop	<ul style="list-style-type: none"> Android iOS Windows Mobile
	Media.release	<ul style="list-style-type: none"> Android iOS
	Media.record	<ul style="list-style-type: none"> Android iOS
	Media.startRecord	<ul style="list-style-type: none"> Android iOS

API	Object and Function	Platform
	Media.stopRecord	<ul style="list-style-type: none"> • Android • iOS • Windows Mobile
	Media.getCurrentPosition	<ul style="list-style-type: none"> • Android • iOS • Windows Mobile
	Media.seekTo	<ul style="list-style-type: none"> • Android • iOS • Windows Mobile
	Media.getDuration <u>Note: On Android, this function returns a value without an error but always returns -1, which indicates duration is not available.</u>	<ul style="list-style-type: none"> • Android • iOS • Windows Mobile
Notification		
	Notification.beep	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile
	Notification.confirm	<ul style="list-style-type: none"> • Android • BlackBerry • iOS
	Notification.alert	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile
	Notification.vibrate	<ul style="list-style-type: none"> • Android • BlackBerry • iOS • Windows Mobile
Storage		

API	Object and Function	Platform
	window <ul style="list-style-type: none"> • OpenDatabase 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS
	Database <ul style="list-style-type: none"> • transaction 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS
	SQLTransaction <ul style="list-style-type: none"> • executeSQL <hr/> <p>Note: On Android, queries on the first database created do not work. You can work around this by creating and opening two databases, the first of which can have the size of 0, and the second to use as you normally do. For example:</p> <pre>var db = window.openDatabase("aName1", "1.0", "aName1", 0); db = window.openDatabase("aName2", "1.0", "aName2", 200000);</pre> <hr/>	<ul style="list-style-type: none"> • Android • BlackBerry • iOS
	SQLResultSet <ul style="list-style-type: none"> • insertid • rowAffected <hr/> <p>Note: The returned SQLResultSet object does not contain a rowAffected property, as the Phone-Gap API states. Instead, use rowsAffected.</p> <hr/> <ul style="list-style-type: none"> • rows 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS
	SQLResultSetList <ul style="list-style-type: none"> • item • length 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS
	SQLError <ul style="list-style-type: none"> • code • message 	<ul style="list-style-type: none"> • Android • BlackBerry • iOS

API	Object and Function	Platform
	localStorage <ul style="list-style-type: none"> • key • getitem • setitem • removeitem • clear 	iOS

Implementing PhoneGap

The recommended methods of implementing PhoneGap are to use the AppFramework, or to load PhoneGap in the same way as the Apache Cordova installation does.

To use the same HTML for every platform, include the Cordova files as javascript files, then dynamically load that code based on the platform that is running. The Cordova files are packaged in the `SMP_HOME\UnwiredPlatform\MobileSDKversion\HybridApp\Containers\Platform` directories.

```
function loadPhoneGap() {
var jsfile = null;
var pre = "";
var language = hwc.getURLParam("lang");
if (!(language === undefined) && (language.length > 0)){
pre = "../";
}
if (hwc.isAndroid()) {
jsfile = pre + "js/android/cordova-2.0.0.javascript";
}
else if (hwc.isIOS()) {
jsfile = pre + "js/ios/cordova-2.0.0.javascript";
}
else if (hwc.isBlackBerry()) {
jsfile = pre + "js/blackberry/cordova-2.0.0.javascript";
}
if (jsfile) {
var req = null;
if (window.XMLHttpRequest) {
req = new XMLHttpRequest();
}
else { // code for IE6, IE5
req = new ActiveXObject("Microsoft.XMLHTTP");
}
req.open("GET", jsfile, false);
req.send(null);
// Need to call eval with the global context
window[ "eval" ].call( window, req.responseText );
}
}
loadPhoneGap();
```

Initializing PhoneGap for Storage Methods

If your application calls a storage function (`hwc.SUPStorage` or `hwc.SharedStorage`), the PhoneGap must have been initialized first. If you generate your application in the Hybrid App Designer, the application detects the initialization automatically. However, if you do not generate your application using Designer, you must add code to recognize when PhoneGap is initialized.

For example, in `Custom.js`, add this code:

- This new function displays a notification:

```
function phoneGapIsready() {
    showAlertDialog("PhoneGap is ready");
}
```

- This customization detects when the PhoneGap initialization occurs and displays your notification:

```
function hwc.customAfterHybridAppLoad() {
    document.addEventListener("deviceready",
    phoneGapIsReady, false);
}
```

Alternatively, detect the initialization directly in your Hybrid App HTML file:

```
<body onload='document.addEventListener("deviceready",
phoneGapIsReady, false)'\>
```

PhoneGap Custom Plug-ins

You can write custom plug-ins for PhoneGap.

Custom PhoneGap plug-ins have a JavaScript component that exposes the custom native component and a native component. See the *PhoneGap* documentation for information about PhoneGap plug-ins.

Custom Plug-ins for the Android Hybrid Web Container

Integrate PhoneGap (Cordova) plug-ins with the Android Hybrid Web Container.

In general, adding a custom plug-in to Hybrid Web Container is identical to adding a plug-in to any PhoneGap application. The basic steps are as follows (see the *PhoneGap Wiki* for details).

1. Create an Android project.
2. Include Cordova dependencies.
3. Implement the plug-in class.
4. Implement the plug-in JavaScript.

Adding a Custom Plug-in to the Android Hybrid Web Container

Add a PhoneGap (now called Cordova) plug-in to the Android Hybrid Web Container.

Prerequisites

Download and install the Android Developer Tools (ADT), available from <http://developer.android.com/sdk/index.html>.

Task

1. In Eclipse, import the HybridWebContainer project:
 - a) Select **File > Import**.
 - b) Expand **Android**, choose **Existing Android Code into Workspace**, and click **Next**.
 - c) In **Import Projects**, click **Browse** and select the root directory of the Android project to import.
For example, if you have previously unpacked the Android HWC container to `SMP_HOME\MobileSDKversion\HybridApp\Containers\Android\Android_HWC_version`, select that folder and click **OK**.
 - d) Click **Finish**.

2. In the HybridWebContainer project, open `res/xml/config.xml`.

3. Add your custom plug-in.

For example:

```
<plugin name="DirectoryListPlugin"
value="com.sybase.hwc.DirectoryListPlugin" />
```

4. Add plug-in images to the HybridWebContainer project.

The plug-in used in this example does not include images, but they are allowed in plug-ins. Images for plug-ins are usually stored in `res\drawable`.

5. Add the Java source file that implements the custom plug-in, for example, `DirectoryListplugin.java`.

This example PhoneGap plug-in lists all files on the SDCard of the device.

```
/**
 * Example of Android PhoneGap Plugin
 */
package com.sybase.hwc;

import java.io.File;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;

import android.util.Log;

import org.apache.cordova.api.Plugin;
```

```

import org.apache.cordova.api.PluginResult;
import org.apache.cordova.api.PluginResult.Status;

/**
 * PhoneGap plugin which can be involved in following manner from
 javascript
 * <p>
 * result example - {"filename":"/
sdcard","isdir":true,"children":
[{"filename":"a.txt","isdir":false},{..}]
 * </p>
 * <pre>
 * {@code
 * successCallback = function(result){
 *     //result is a json
 * }
 * failureCallback = function(error){
 *     //error is error message
 * }
 * }
 * window.plugins.DirectoryListing.list("/sdcard",
 *                                     successCallback
 *                                     failureCallback);
 * }
 * </pre>
 * @author Rohit Ghatol
 */
public class DirectoryListPlugin extends Plugin {

    /** List Action */
    public static final String ACTION="list";

    /*
     * (non-Javadoc)
     * @see
org.apache.cordova.api.Plugin#execute(java.lang.String,
 * org.json.JSONArray, java.lang.String)
     */
    @Override
    public PluginResult execute(String action, JSONArray data,
String callbackId) {
        Log.d("DirectoryListPlugin", "Plugin Called");
        PluginResult result = null;
        if (ACTION.equals(action)) {
            try {

                String fileName = data.getString(0);
                JSONObject fileInfo = getDirectoryListing(new
File(fileName));
                Log
                    .d("DirectoryListPlugin", "Returning "
                        + fileInfo.toString());
            }
        }
    }
}

```

```
        result = new PluginResult(Status.OK, fileInfo);
    } catch (JSONException jsonEx) {
        Log.d("DirectoryListPlugin", "Got JSON Exception "
            + jsonEx.getMessage());
        result = new PluginResult(Status.JSON_EXCEPTION);
    }
    } else {
        result = new PluginResult(Status.INVALID_ACTION);
        Log.d("DirectoryListPlugin", "Invalid action : "+action
+" passed");
    }
    return result;
}

/**
 * Gets the Directory listing for file, in JSON format
 * @param file The file for which we want to do directory
listing
 * @return JSONObject representation of directory list. e.g
{"filename":"/sdcard","isdir":true,"children":
[{"filename":"a.txt","isdir":false},{..}]
 * @throws JSONException
 */
private JSONObject getDirectoryListing(File file)
    throws JSONException {
    JSONObject fileInfo = new JSONObject();
    fileInfo.put("filename", file.getName());
    fileInfo.put("isdir", file.isDirectory());

    if (file.isDirectory()) {
        JSONArray children = new JSONArray();
        fileInfo.put("children", children);
        if (null != file.listFiles()) {
            for (File child : file.listFiles()) {
                children.put(getDirectoryListing(child));
            }
        }
    }

    return fileInfo;
}
}
```

6. Save the file.

These are all the changes needed for the Hybrid Web Container; you can now build it and install it on the device. What the plug-in actually does is implemented in the Java file in the **execute** function.

Testing the Plug-in

Test the PhoneGap plug-in for the Android Hybrid Web Container.

1. Create a new Mobile Application project:

- a) Select **File > New > Mobile Application Project**.

- b) In Project name, enter PhonegapTest.
- c) Click **Finish**.
- 2. Right-click the PhonegapTest project folder and select **NewHybrid App Designer**.
- 3. Click **Next**.
- 4. Select **Can be started on demand from the client** and click **Finish**.
- 5. Click **Screen Design**.
- 6. Add a Menu Item control of type Custom to the Menu, and in the General properties, enter "c" for the menu item name.

This is the key name you will use for the `customAfterMenuItemClick ()` function in the `custom.js` file.

- 7. Run the Hybrid App Generation wizard to create the directory structure Generated Hybrid App\PhonegapTest\ html\js.
- 8. Open the `custom.js` file for editing and add this code before the line `(function(hwc, window, undefined) :`

```
var dirlist = {
    getlist: function(successCallback, errorCallback) {
        PhoneGap.exec(successCallback, errorCallback,
        'DirectoryListPlugin', 'list', ["/mnt/sdcard"]);
    }
};

function getDlist() {
    dirlist.getlist(function(r) {
        var theHtml = "";
        if(r.children)
        {
            var index = 0;
            for(index = 0; index <= r.children.length; index++)
            {
                if(r.children[index]){
                    theHtml += r.children[index].filename + " \n ";
                }
            }
        }
        else
        {
            alert("No r.children!!");
        }
        alert(theHtml);
    },
    function(error) {
        alert('Error:' + error);
    });
}
```

- 9. Add this code in the `customAfterMenuItemClick ()` function:

```
if(menuItem == "c"){
    getDlist();
}
```

10. Regenerate the Hybrid App.
11. Assign the Hybrid App to a device that has the Hybrid Web Container with the custom plug-in.
12. On the device, run the Hybrid App, click **Menu**, and click **c**.
The `custom_plug-in.java` file appears on the SD card in the list of files.

Note: The code returns a list of files only if an SD card is configured on the device (or, on an emulator, if an SD Card is configured in AVD). If no SD card is configured, the code returns no list.

Custom Plug-ins for the BlackBerry Hybrid Web Container

Integrate PhoneGap plug-ins with the BlackBerry Hybrid Web Container.

In general, adding a custom plug-in to Hybrid Web Container is identical to adding a plug-in to any PhoneGap application. See the *PhoneGap Wiki*. The basic steps are:

1. Set up a BlackBerry Eclipse development IDE. See <http://us.blackberry.com/developers/javaappdev/javaplugin.jsp>.
2. Create the plug-in source code.
3. Provide the JavaScript API.
4. Package the plug-in source code.
5. Include the PhoneGap dependencies.

Adding a Custom Plug-in to the BlackBerry Hybrid Web Container

Add a PhoneGap plug-in to the BlackBerry Hybrid Web Container

Prerequisites

Set up the BlackBerry Eclipse development IDE. See <http://us.blackberry.com/developers/javaappdev/javaplugin.jsp>

Task

This example procedure shows the steps to create and use a custom plug-in to get battery information for the device.

1. In Eclipse, import the HybridWebContainer project.
2. Open the `plugins.xml` file, which is located in `res/xml`, and add this tag:

```
< plugin name="Battery1" value="com.sybase.hwc.Battery1"/>
```
3. Add a new Java source file called `Battery1.java` to the `src` folder, and paste in this code:

```
package com.sybase.hwc;
import org.apache.cordova.api.Plugin;
```

```

import org.apache.cordova.api.PluginResult;
import org.apache.cordova.json4j.JSONArray;

public class Battery1 extends Plugin {
    public static final String GET_LEVEL = "getLevel";

    /**
     * Executes the requested action and returns a PluginResult.
     *
     * @param action      The action to execute.
     * @param callbackId The callback ID to be invoked upon action
completion.
     * @param args       JSONArray of arguments for the action.
     * @return           A PluginResult object with a status and
message.
     */
    public PluginResult execute(String action, JSONArray args,
String callbackId) {
        PluginResult result = null;
        if (GET_LEVEL.equals(action)) {
            // retrieve the device battery level
            int level =
net.rim.device.api.system.DeviceInfo.getBatteryLevel();
            result = new PluginResult(PluginResult.Status.OK,
level);
        }
        else {
            result = new
PluginResult(PluginResult.Status.INVALID_ACTION,
                "Battery: Invalid action: " + action);
        }
        return result;
    }

    /**
     * Called when Plugin is paused.
     */
    public void onPause() {
    }

    /**
     * Called when Plugin is resumed.
     */
    public void onResume() {
    }

    /**
     * Called when Plugin is destroyed.
     */
    public void onDestroy() {
    }
}

```

4. Save the file.

These are all the changes needed for the Hybrid Web Container; you can now build it and install it on the device. What the plug-in actually does is implemented in the Java file in the

execute function. The rest of this example explains how to test and use the PhoneGap plug-in.

5. Create a new Hybrid App.
 - a) Select **File > New > Mobile Application Project**.
 - b) In Project name, enter PhonegapTest.
 - c) Click **Finish**.
6. Right-click the **PhonegapTest** project folder and select **New > Hybrid App Designer**.
7. Click **Next**.
8. Select **Can be started, on demand, from the client** and click **Finish**.
9. Add an **HtmlView** control to the start screen of the Hybrid App.
10. Run the Hybrid App Package Generation wizard to create the Generated Hybrid App directory structure Generated Hybrid App\PhonegapTest\ html\js.
11. Open the Custom.js file and add this code:

```
var Battery1 = {
    level: function(successCallback, errorCallback) {
        PhoneGap.exec(successCallback, errorCallback,
        'Battery1', 'getLevel', []);
    }
};

function getBatteryLevel() {
    Battery1.level(function(level) {
        alert('Battery level is ' + level);
    },
    function(error) {
        alert('Error retrieving battery level:' + error);
    });
}
```

12. Find the customAfterHybridAppLoad() function, and add this code:

```
function customAfterHybridAppLoad() {
    document.addEventListener("deviceready", getBatteryLevel,
    false );
}
```

This is the code that makes use of the plug-in.

13. Generate the Hybrid App package again.
14. Assign the Hybrid App to a device that has the modified Hybrid Web Container installed.
15. On the device, run the Hybrid App.

You see the alert message with the battery level information.

Custom Plug-ins for the iOS Hybrid Web Container

Integrate PhoneGap plug-ins with the iOS Hybrid Web Container.

In general, adding a custom plug-in to Hybrid Web Container is identical to adding a plug-in to any PhoneGap application. The basic steps are as follows (see *the PhoneGap Wiki for details*).

1. Implement the plug-in class that extends PGPlugin in an .h and .m file.
2. Implement the PhoneGap plug-in JavaScript.
3. Edit the PhoneGap plist file with a new plug-in entry.
4. Use the plug-in from JavaScript.

Adding a Custom Plug-in to the iOS Hybrid Web Container

An example plug-in class that allows access to the iOS network activity monitor is available in HybridWebContainer/Classes/Plugins.

1. Copy the `networkActivityIndicator.h` and `networkActivityIndicator.m` files from HybridWebContainer/Classes/Plugins to the `HWC.xcodeproj` project.
2. Add the `networkActivityIndicator.js` to the Hybrid App `/html/js/` directory that corresponds with the Eclipse project that generated the Hybrid App files.
3. Modify your JavaScript for any event desired to call the new plug-in.

Here is an example that reacts to a menu item and uses a global variable to toggle the activity indicator on and off.:

```
var gActIndicator = true; // global variable

function customAfterMenuItemClick(screen, menuItem) {
  if (screen === "Start" && menuItem === "networkActivityIndicator")
  {
    window.networkActivityIndicator.set( gActIndicator, aiSuccess,
    aiFail );
    // Toggle the network activity indicator each time plugin is
    selected
    if ( gActIndicator )
      gActIndicator = false;
    else
      gActIndicator = true;
    return false;
  }
}

function aiSuccess() {
  alert("Successfully enabled activity indicator");
}

function aiFail() {
```

```
alert("Failed to enable activity indicator");
}
```

4. Add a plug-in entry to `Cordova.plist`:

```
Key: networkActivityIndicator
Type: String
Value: networkActivityIndicator
```

5. Generate the Hybrid App files and deploy the package to the server..
6. Test the event in the JavaScript file that is hooked into the new plug-in.

If the plug-in requires additional resources, such as images or other files, these should be added to the project under the `Resources` group folder. For example, the `ChildBrowser` plug-in available at github.com contains icons that are stored in a file called `ChildBrowser.bundle`. In this example, the `ChildBrowser.bundle` should be added to the `Resources` group folder in the project in Xcode.

Some plug-ins also require files to be in a `www/` directory. The `notification.beep` API is one example. If this is the case, add the resources to the `www` directory that is referenced by the project under the `Resources` group folder as described in Step 7 in *Upgrading the PhoneGap Library used by the iOS Hybrid Web Container*.

Custom Plug-ins for the Windows Mobile Hybrid Web Container

Integrate PhoneGap plug-ins with the Windows Mobile Hybrid Web Container.

In general, adding a custom plug-in to Hybrid Web Container is identical to adding a plug-in to any PhoneGap application. The basic steps include:

1. Implement the plug-in class that extends the class "PluginBase."
2. Implement the PhoneGap plug-in JavaScript.
3. Add the plug-in class to the plug-in configuration file.
4. Use the plug-in from JavaScript.

Adding a Custom Plug-in to the Windows Mobile Hybrid Web Container

This procedure shows an example of adding a plug-in class that allows access to the Windows Mobile calculator.

The plug-in class is available under the `TPTools\phoneGap\wm` directory. To include this plug-in in the Hybrid Web Container, follow these steps:

1. Add a new class called `Calculator` into the folder `CustomCode` and implement the code:

```
using WMGapClassLib.Cordova;
namespace Sybase.Hwc.CustomCode
{
    public class Calculator : PluginBase
    {
        public void sum(Session session,
            Sybase.HybridApp.Util.Json.JsonObject arguments)
        {
            try
```

```

        {
            double x = 0;
            double y = 0;
            x = double.Parse(arguments.GetString("x"));
            y = double.Parse(arguments.GetString("y"));
            this.DispatchCommandResult(session, new
PluginResult(PluginResult.Status.OK, x + y));
        }
        catch (System.Exception ex)
        {
            this.DispatchCommandResult(session, new
PluginResult(PluginResult.Status.ERROR, ex.Message));
        }
    }
}
}
}

```

2. Open the file `Plugins.xml`, which is located in the `HybridWebContainer` project, and add the custom plug-in:

```

<?xml version="1.0" encoding="utf-8" ?>
<plugins>
  <plugin id="showcertpicker"
        class="Sybase.Hwc.CertificationPickerPlugin"/>
  <plugin id="Calculator"
        class="Sybase.Hwc.CustomCode.Calculator"/>
</plugins>

```

3. Open the `Custom.js` file for editing and add this method:

```

function calculateSum(x, y, successCb, errorCallback){
cordova.require('cordova/exec') (
    successCb,
    errorCallback,
    "Calculator", "sum",
    { x: document.getElementById('x').value, y:
document.getElementById('y').value });
}

```

4. Call this JavaScript method somewhere else to get the result:

```

function doCalculateSum() {
    calculateSum(

        document.getElementById('x').value,

        document.getElementById('y').value,
        function (res){
            document.getElementById('res').innerHTML = res;
        },
        function (e) {

            console.log("Error occurred: " + e);

            document.getElementById('res').innerHTML = "Error
occurred: " + e;
        });
};

```

5. Generate and deploy the application and test the event in the `custom.js` file that is hooked into the new plug-in.

Initializing the PhoneGap Library for the Windows Mobile Hybrid Web Container

You must initialize the PhoneGap library before using it.

1. Open the HTML file for the Hybrid App for editing.
2. Add this code.

```
<Html>
<script>
Function onLoad() {
    try
    {

        cordova.require('cordova/
channel').onDOMContentLoaded.fire();

        cordova.require('cordova/
channel').onNativeReady.fire();
                                _nativeReady = true;
    }
    catch(e)
    {
        alert("Initialize
phonegap error:" + e.message);
    }
}
</script>
<body onload="onLoad();" >
</html>
```

3. Save the file.
4. Regenerate the Hybrid App package.

PhoneGap Library Downgrade

SAP Mobile Platform included PhoneGap 1.4.1 libraries embedded inside the iOS and Android Hybrid Web Containers.

SAP Mobile Platform 2.2 includes the Cordova 2.0 libraries for Android, BlackBerry, iOS, and Windows Mobile. When PhoneGap changed to the Cordova name in 1.5, interfaces to native PhoneGap plug-ins were renamed, thus, 2.1.3 Hybrid Apps that use the PhoneGap 1.4.1 will not work with 2.2 Hybrid Web Container. If you want to continue to use the PhoneGap 1.4.1 libraries with the 2.2 Android Hybrid Web Container, you can revert from the Cordova 2.0 libraries to the PhoneGap 1.4.1 libraries.

Downgrading the PhoneGap Library Used by the Android Hybrid Web Container

Change from the Cordova 2.0 library included with the Android Hybrid Web Container to the PhoneGap 1.4.1 library.

The files referenced in this procedure are located in the *Android_PhoneGap_Downgrade.zip* file.

1. Use a diff utility tool to compare the file `UiHybridAppContainer_before.java` and `UiHybridAppContainer_after.java` files.
2. Open the `UiHybridAppContainer.java` file, which is located in `..HybridWebContainer\src\com\sybase\hwc`, and apply the changes found with the diff utility tool.

Note: Keep in mind that this change could remove bug fixes, or cause unexpected behavior of the related new features.

3. Rebuild the Hybrid Web Container project to make sure there are no compilation errors.
4. Replace the `cordova-2.0.0.jar` located in `<SMP_HOME>\UnwiredPlatform\MobileSDK23\HybridApp\API\Container\android`, with the `phonegap-1.4.1.jar` file, which is in the *Android_PhoneGap_Downgrade.zip* file.
5. In the HybridWebContainer project, remove the `res/xml/config.xml` file and add the `plugins.xml` and `phonegap.xml` files.
6. Open the `UiHybridAppContainer.java` file for editing and change the import statement from `import org.apache.cordova.DroidGap` to `import com.phonegap.DroidGap`.
7. Find the method:

```
@Override
    public void onCreate( Bundle savedInstanceState ) {
        super.setBooleanProperty("showTitle", true );
        super.onCreate( savedInstanceState );
    }
```

Remove the line: `super.setBooleanProperty("showTitle", true);`.

8. Rebuild the HybridWebContainer project.
9. (Optional) Rename the `phonegap-1.4.1.js` file to `phonegap-1.4.1.javascript`.
10. (Optional) In the Container folder of generated applications, replace the `android/cordova-2.0.0.javascript` with `phonegap-1.4.1.javascript`.

11. (Optional) In the `API.js` file, remove the string `android/cordova-2.0.0.javascript` and replace it with `android/phonegap-1.4.1.javascript`.

Using the HTTPS Proxy Exposed by the PhoneGap Plugin

PhoneGap JavaScript application users that want to send an AJAX request to a HTTPS server, where the embedded browser does not support SSL, can use the HTTPS proxy exposed by the PhoneGap plugin.

The HTTPS proxy supports both HTTP and HTTPS connections. HTTPS connections are supported for both server side certificate validation and client side certificate, or either, or none. A keystore/certificate from either system, file, or Afaria is loaded when first required. Compared to the lifecycle of certificates, the lifecycle of certificates in memory is very short. The user cannot refresh newly loaded keystores/certificates. If a new certificate is added or updated, you must restart the application to include it.

PhoneGap HTTPS proxy connection properties

The JavaScript API definition of the HTTPS proxy is: Namespace : `HttpsConnetion`.

This table describes the platform dependent methods, error codes, and object definitions. Its implementation is platform dependent so every platform should provide its own version of the JavaScript code.

Table 10. PhoneGap HTTPS proxy connection properties

Method/definition/error code	Description
CertificateFromFile (path, password, certificate-Key)	<p>Create a certificate descriptor for the certificate from a file. Calling this method does not immediately load the certificate.</p> <ul style="list-style-type: none"> • path – path of the keystore file • password – password of the keystore • certificateKey – certificate key of the certificate in the keystore, which is the alias in the Java keytool
CertificateFromAfaria (cn, challengeCode)	<p>Create a certificate descriptor for the certificate from the Afaria server. Calling this method does not immediately load the certificate.</p> <ul style="list-style-type: none"> • cn – common name of the certificate • challengeCode – challenge code to the Afaria server

Method/definition/error code	Description
CertificateFromStore (certificateKey)	<p>Create a certificate descriptor for certificate from files. Calling this method does not immediately load the certificate.</p> <ul style="list-style-type: none">• certificateKey – certificate key of the certificate in the system keystore, which is the alias in the Java keytool
deleteCertificateFromStore(certificateKey)	<p>(iOS only) Delete a cached certificate from the keychain. The iOS client always tries the cached certificate first if available, before requesting the certificate from the Afaria server or loading the certificate from the file system.</p> <p>In cases where the cached certificate is no longer valid, use this method to delete it from the keychain.</p>

Method/definition/error code	Description
<p>get(url, header, successCB, errorCB, userId, password, timeout, certSource)</p>	<p>Send a HTTP request of the GET method.</p> <ul style="list-style-type: none"> • url – the full URL in format https://...[:port] • header – header of the request in JSON Object • successCB – callback method upon success. Its parameter is a string encoded JSON object with these fields: <pre data-bbox="759 477 1176 656"> {status: number; headers: JSON object with string fields; responseText: Optional if the requested data is text; responseBase64: Optional if the requested data is binary} </pre> <p>Callers must provide this method otherwise an exception is thrown.</p> • errorCB – callback method upon failure. Its parameter is an object with these fields: <pre data-bbox="759 819 1108 894"> {errorCode: number; description: string; nativeErrorCode: number} </pre> <p>Callers must provide this method otherwise an exception is thrown.</p> • userID – (Optional) for basic authentication • password – (Optional) for basic authentication • timeout – (Optional) in seconds • certSource – (Optional) The JavaScript certificate description object <p>Returns a JavaScript object that contains an abort() method to abort the current connection</p>

Method/definition/error code	Description
<p>sendRequest(method, url, header, requestBody, successCB, errorCallback, userId, password, timeout, certSource)</p>	<p>Send a generic HTTP request to the server.</p> <ul style="list-style-type: none"> • url – the full URL in format https://...[:port] • header – header of the request in JSON Object • requestBody – data as a string value to be sent to server with the request. • successCB – callback method upon success. Its parameter is a string encoded JSON object with these fields: <pre>{status: number; headers: JSON object with string fields; responseText: Optional if the requested data is text; responseBase64: Optional if the requested data is binary}</pre> <p>Callers must provide this method otherwise an exception is thrown.</p> • errorCB – callback method upon failure. Its parameter is an object with these fields: <pre>{errorCode: number; description: string; nativeErrorCode: number}</pre> <p>Callers must provide this method otherwise an exception is thrown.</p> • userID – (Optional) for basic authentication • password – (Optional) for basic authentication • timeout – (Optional) in seconds • certSource – (Optional) The JavaScript certificate description object <p>Returns a JavaScript object that contains an abort() method to abort the current connection</p>
ERR_UNKNOWN	The operation failed with an unknown error.
ERR_INVALID_PARAMETER_VALUE	The operation has an invalid parameter.
ERR_MISSING_PARAMETER	The operation failed because of a missing parameter.

Method/definition/error code	Description
ERR_NO_SUCH_ACTION	There is no such Cordova action for the current service.
ERR_FILE_CERTIFICATE_SOURCE_UNSUPPORTED	Certificate from file keystore is not supported on the current platform.
ERR_SYSTEM_CERTIFICATE_SOURCE_UNSUPPORTED	Certificate from system keystore is not supported on the current platform.
ERR_AFARIA_CERTIFICATE_SOURCE_UNSUPPORTED	Certificate from Afaria server is not supported on the current platform.
ERR_CERTIFICATE_ALIAS_NOT_FOUND	The certificate with given alias could not be found.
ERR_CERTIFICATE_FILE_NOT_EXIST	The certificate file could not be found.
ERR_CERTIFICATE_INVALID_FILE_FORMAT	Incorrect certificate file format.
ERR_GET_CERTIFICATE_FAILED	Failed in getting certificate.
ERR_CLIENT_CERTIFICATE_VALIDATION	The provided certificate failed server-side validation.
ERR_SERVER_CERTIFICATE_VALIDATION	The server certificate failed client-side validation.
ERR_SERVER_REQUEST_FAILED (-110)	Exception message reported by HttpURLConnection. The native code should contain the specific error information.
ERR_HTTP_TIMEOUT	Timeout error while connecting to the server.

Requirements and Limitations

Additional requirements and limitations for using the PhoneGap HTTPS proxy.

- Although the embedded browser in BlackBerry supports HTTPS requests, this implementation is required to support Afaria.
- System keystore for Android versions prior to 4.0 is not supported.
- Multiple instances of certificates/keystores are supported for certificate/keystore from files and Afaria.
- The plugin in a production environment denies any trust confirmation for server side certificates.
- The API format for how to call `cordova.exec(...)` is not defined, because it is transparent to users, and is platform dependent, so implementation is left to plugin developers.

Hybrid App Configuration for Data Change Notification

This section contains details about developing Hybrid Apps that take advantage of DCN updates.

Hybrid Apps require a server-initiated starting point and defined matching rules, which allows SAP Mobile Server to push changes to Hybrid App clients. See the topics *Starting Points* and *Adding Matching Rules* in *SAP Mobile WorkSpace - Hybrid App Package Development*.

Extending Data Change Notification to Hybrid Apps

Data change notification (WF-DCN) requests allow SAP Mobile Server to process the DCN request and send notification to the device of that data change.

Depending on the cache policy used by the affected MBO, once the application receives notification, it can retrieve data directly from the EIS or from the SAP Mobile Server cache, keeping the application synchronized. DCN messages targeted for MBOs used in applications (WF-DCN), uses similar syntax as general DCN, with these differences:

- The value of **cmd** is *wf* for WF-DCN requests, compared to *dcn* for regular DCN.
- The message contains the fields required for notification, such as the to address, from address, e-mail subject, and e-mail body.
- The WF-DCN message is captured and parsed by the server-initiated Hybrid App, which processes the WF-DCN message differently, depending on the message type: with payload or without payload.

WF-DCN format

The WF-DCN request is a JSON string consisting of these fields: engine converts MBO data and WF-DCN messages into email, and pushes it to device's inbox

1. Operation name(op) **:upsert** or **:delete**– same as regular DCN.
2. Message ID (id) of the Hybrid App – used for correlation (a **:delete** for a previously submitted request with **:upsert** is possible)

Note: Do not send DCNs with the same Message ID from different back-end systems. Make sure to use different Message IDs for different DCNs.

3. Username (to) – the SAP Mobile Platform user name. For the user to be recognized by WF-DCN, the device user should first have established communication using the activation mechanism in SAP Control Center.

Note: The "To" field must match the SAP Mobile Platform user name—for example, if using auto-registration, the user name used to register the device is the "device user name"

or the "application connection user name" (in either case this refers to the user name used to register the device). And the WF-DCN "To" field can use this name to push the message to the device. Additionally, there is another package user name which is established during activation of the Hybrid App. The package user name can also be used as the "To" field for pushing the message.

For manually registered devices, the WF-DCN is pushed based on the package user name established after Hybrid App activation.

-
4. Subject (subject) – subject of the Hybrid App message.
 5. Originator <from> – who the Hybrid App message is from.
 6. Body of the Hybrid App message <body> – it can embed customized information.
 7. <received> – received time of the Hybrid App message.
 8. <read> – whether the Hybrid App message is read.
 9. <priority> – whether the Hybrid App message has a high priority.
 10. List of dcn request <data> – JSON format string.

Example DCN request in JSON format:

```
{
  "op":":upsert",
  "id":":WID123",
  "to":":SUPAdmin",
  "subject":":Trip request approval required",
  "from":":user321",
  "body":":This is a message just used to do a test",
  "received":":2009-03-29T10:07:45+05:00",
  "read":false,
  "priority":true,
  "data":
  [
    {
      "id": "1",
      <general dcn request>
    }
    ...
    {
      "id": "4",
      < general dcn request>
    }
  ]
}
```

Hybrid App DCN request flow

WF-DCN with and without payload differ slightly, but the general flow is similar for each. When the WF-DCN request is received, SAP Mobile Server gets the **wf** cmd value from the request first, and:

1. SAP Mobile Server invokes `preProcessFilter` if the DCN filter is specified.
2. SAP Mobile Server receives a raw HTTP POST body to generate and return a WF-DCN request message object.
3. The JSON format string is parsed into a WF-DCN request object.

4. The DCN request in the Hybrid App message object is parsed and those within the scope of a single transaction per DCN request object in the array are executed. Results are recorded for a report after completing the WF-DCN request.
5. From the CDB, the server looks up all users assigned to the indicated Hybrid App package in the “to” attribute of the Hybrid App message, then matches them with the receiver list. For every receiver, SAP Mobile Server generates multiple Hybrid App messages (all Hybrid App messages are created within one transaction), one per device identified (one user might have multiple devices), and then sends them to the JMS queues.
The lookup of the logical id is performed by combining the username in the “to” list to the “securityProfile” specified in the HTTP POST REQUEST URL parameter list.
6. If any errors occur in step four, step five does not execute. If any errors occur in step five, step five is not committed. If any errors occur in either of those steps, an HTTP 500 error is returned.
7. SAP Mobile Server invokes the `postProcessFilter`, if specified.
8. If no errors occur, SAP Mobile Server returns success to the caller HTTP 200 with the body of the JSON string (or any opaque data returned from the `postProcessFilter`) of the WF-DCN Result. Otherwise, SAP Mobile Server returns an HTTP 500 error with the body of the JSON log records.

Device Registration

For Security reasons SAP Mobile Server pushes WF-DCN notifications only to auto-registered device users. For example, if there are two application connections with the same name and one is auto-registered and the other is manually registered, SAP Mobile Server pushes the WF-DCN notification only to the auto-registered device. For manually registered devices, the WF-DCN is pushed based on the package user, which is created when the Hybrid App is assigned to the device and the user activates the Hybrid App using EIS user name/password.

See the topic *Registering Applications, Devices, and Users* in the *Security Guide*.

Non HTTP Authentication Hybrid App DCN Request

You can send Hybrid App DCN requests that are not authenticated.

The URL is:

```
http://host:8000/dcn/DCNServlet?  
cmd=wf&security=admin&domain=default&username=supAdmin&password=sup  
Pwd&dcn_filter=aa.bb&dcn_request=<wfrequestdata>
```

where *supAdmin* represents the SAP Mobile Server Administrator, and *supPwd* represents the Administrator's password defined during SAP Mobile Platform installation.

Sending Hybrid App DCN to Users Regardless of Individual Security Configurations

You can send Hybrid App DCN requests to users in other security configurations if you belong to the default security configuration.

If the Hybrid App DCN sender is authenticated against the default admin security configuration, they are automatically authorized to push data to all users regardless of their individual security configuration. If not, the sender can only push to users within the same security configuration.

For example, in the case of a non HTTP authentication request, this request is authorized to push data to users in other security configurations since the sender *supAdmin*, belongs to the admin security configuration:

```
http://host:8000/dcn/DCNServlet?  
cmd=wf&security=othersecurity&domain=default  
&username=supAdmin@admin&password=supPwd&dcn_filter=aa.bb&dcn_reque  
st=<request>
```

And this request is denied because *supAdmin@mysecurityconfig* can only push data to users in the same security configuration:

```
http://host:8000/dcn/DCNServlet?  
cmd=wf&security=othersecurity&domain=default  
&username=supAdmin@mysecurityconfig&password=supPwd&dcn_filter=aa.b  
b&dcn_request=<request>
```

Hybrid App DCN Request Response

After processing of the Hybrid App DCN request, SAP Mobile Server sends the response to notify the caller whether the request was processed successfully.

The response includes two parts:

1. The result of processing the Hybrid App request.
2. The result of processing the general DCN requests.

The response is also in a JSON format string:

```
{  
<wf dcn result>  
"result":  
  [  
    {  
      <general dcn result>  
    },  
    {  
      <general dcn result>  
    }  
  ]  
}
```

```
}  
]  
}
```

An example response is:

```
{  
  "id": "1",  
  "success": false,  
  "statusMessage": "there is error in processing dcn",  
  "result":  
  [  
    {  
      "id": "1",  
      "success": true,  
      "statusMessage": ""  
    },  
    {  
      "id": "2",  
      "success": false,  
      "statusMessage": "bad msg2"  
    }  
  ]  
}
```

Hybrid App DCN Design Approach and Sample Code

Understand the design approach for both WF-DCN with and without payload, and view samples for each approach.

Note: Samples are for illustrative purposes only and should not be used as a guide for developing your DCN requests.

Comparing Hybrid App DCN With and Without Payload

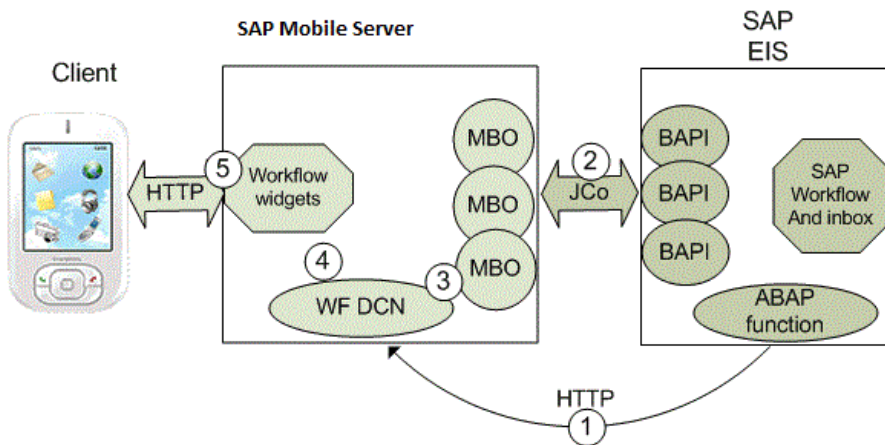
This section compares the two types of WF-DCN and includes examples of each.

Hybrid App DCN Without Payload

Understand how to construct a Hybrid App DCN without payload message.

This example illustrates data flow of a WF-DCN without payload using an SAP® EIS:

Hybrid App Configuration for Data Change Notification



1. The WF-DCN pushes new messages (workitems) to SAP Mobile Server, which are then delivered to the device, for example, a Hybrid App notification.
2. After the EIS sends a workitem id to SAP Mobile Server, SAP Mobile Server uses workitem MBO and workitem id to retrieve details of the workitem from the EIS.
3. After SAP Mobile Server receives the message, a matching Hybrid App server starting point parses the message and extracts data fields from the message, including data into the parameter of an MBO object query operation.
4. Since the MBO uses an online cache policy, the object query is mapped to a load operation, allowing the data to be passed into the load operation as a load argument to trigger an MBO data refresh.
5. The Hybrid App engine converts MBO data and the WF-DCN message into a notification, and pushes it to the device's mobile inbox.

MBO cache group policy

The cache group policy of MBOs used in the WF-DCN without payload must be online. The online MBO contains the findByParameter object query with the same parameters defined in the load operation. The query is triggered by the Hybrid App server-initiated starting point after extracting the parameter values from the WF-DCN message body.

Message format

The message format of the WF-DCN message without payload is:

```
{"id":"","op":"","subject":"","to":"","from":"","read":"","priority":"","body":"","data":{}}
```

For example:

Hybrid App Configuration for Data Change Notification

```
{"id":"","op":":upsert","subject":":test","to":":test","from":":test",  
"read"::,  
"priority":":","body":":MATCH:SUP_MWF,TaskID:TS97200149, WIID:  
1470577,  
USER:PERF0111*#END#*","data":{}}
```

SAP Mobile Server extracts information from the DCN message and retrieves details from the EIS.

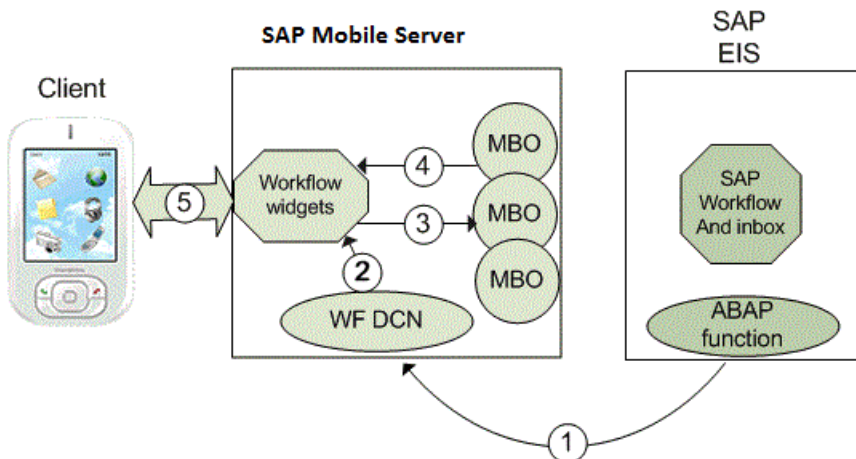
Processing the WF-DCN without payload message

After SAP Mobile Server receives the message, a matching Hybrid App server-initiated starting point parses the message and extracts data fields from the message. The server-initiated starting point sets extracted data into the parameter of an object query operation. Since the MBO used by the without payload message uses an online cache policy, the object query is mapped to a load operation. The data is passed into the load operation as a load argument to trigger MBO data refresh.

Hybrid App DCN With Payload

Understand how to construct a Hybrid App DCN with payload message.

This example illustrates data flow of a WF-DCN with payload using an SAP EIS:



1. When the EIS has new or modified data to push to SAP Mobile Server, it initiates an HTTP request to the WF-DCN URL. The WF-DCN message contains the new or changed data object.
2. When the WF-DCN message reaches SAP Mobile Server, the Hybrid App engine evaluates the matching rule against all registered Hybrid Apps. If a matching rule matches this message, the Hybrid App server starting point for that Hybrid App is triggered to process the message.

Hybrid App Configuration for Data Change Notification

3. The data object included in the WF-DCN message is applied to the MBO CDB table by inserting new records or updating existing records.
4. The Hybrid App server-initiated starting point extracts parameter values from the message body and triggers the MBO object query to retrieve the newly inserted or updated record.
5. The Hybrid App engine converts the MBO data and WF-DCN message into a Hybrid App notification, then pushes it to the device mobile inbox using SAP messaging (MOCA).

MBO cache group policy

The cache group policy of MBOs used in WF-DCN with payload must be DCN.

Message format

The message format of the WF-DCN message with payload is:

```
{ "id": "", "op": "", "subject": "", "to": "", "from": "", "read": "", "priority": "", "body": "", "data": [{"id": "", "pkg": "Package", "messages": [{"id": "2", "mbo": "MBO", "op": "upsert", "cols": {"attribute1": "value1", "attribute2": "value2", "attribute3": "value3"} } ] }
```

The message must contain e-mail information: subject, to, from, and so on, and include the MBO package name and version, MBO name, attribute name, and attribute value. The message can include multiple MBOs. For example:

```
{ "id": "1137", "op": "upsert", "subject": "PERF0111's Leave Request", "to": "PERF0111", "from": "Leave Work Flow", "read": "false", "priority": "true", "body": "MATCH: SUP_MWF, TaskID: TS97200149, WIID: 1470577, USER: PERF0111*##END#*", "data": [{"id": "dcbtst", "pkg": "sup_mwf:1.2", "messages": [{"id": "2", "mbo": "Workitem", "op": "upsert", "cols": {"WORKITEM": "1470577", "USERNAME": "perf0111", "DESCRIPTION": "cc", "DECISION": "test"}}, {"id": "6", "mbo": "Alternatives", "op": "upsert", "cols": {"WORKITEM": "1470577", "USERNAME": "perf0111", "PKEY": "01", "PVALUE": "Ap"} } ] } ] }
```

Sample Java Function for Generating Hybrid App DCN

This WF-DCN sample illustrates WF-DCN without payload.

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.net.Authenticator;
import java.net.HttpURLConnection;
```

```

import java.net.MalformedURLException;
import java.net.PasswordAuthentication;
import java.net.ProtocolException;
import java.net.URL;
import java.net.URLEncoder;

public class HttpAuth
{
    /**
     * @param args
     * @throws MalformedURLException
     */
    public static void main(String[] args) throws Exception
    {
        URL url = null;

        String wfdsn_request = "{\"id\":\"dcntest_69\",\"op\":
\":upsert\", \"
        + \"subject\":\"dept_id = 1300\", \"to\":\"perf0111\", \"
        + \"from\":\"SAP Leave WorkFlow\", \"read\":false,
\":priority\":true, \"
        + \"body\":\"\", TaskID:, WIID:000001468382,
USER:perf0111#END#\"}";

        url = new URL("HTTP", "10.42.39.149", 8000,
"/dcn/HttpAuthDCNServlet?
cmd=wf&security=admin&domain=default");

        HttpURLConnection con = null;

        con = (HttpURLConnection) url.openConnection();

        con.setDoOutput(true);
        con.setRequestMethod("POST");

        final String login = "supAdmin";
        final String pwd = "AdminPassword";
        Authenticator.setDefault(new Authenticator()
        {
            protected PasswordAuthentication
getPasswordAuthentication()
            {
                return new PasswordAuthentication(login,
pwd.toCharArray());
            }
        });

        StringBuffer sb = new StringBuffer();
        sb.append(wfdcn_request);
        OutputStream os = con.getOutputStream();
        os.write(sb.toString().getBytes());
        os.flush();
        os.close();

        StringBuffer xmlResponse = new StringBuffer();

```

Hybrid App Configuration for Data Change Notification

```
        int returnCode = con.getResponseCode();
        if (returnCode != 200)
        {
            String rspErrorMsg = "Error getting response from the
server (error code "
                + returnCode + ")" + con.getResponseMessage();
            System.out.println(rspErrorMsg);
        }
        else
        {
            BufferedReader in = new BufferedReader(new
InputStreamReader(con
                .getInputStream(), "UTF-8"));
            String line;
            while ((line = in.readLine()) != null)
            {
                xmlResponse.append(line).append("\n");
            }
            System.out.println("xmlResponse: " + xmlResponse);
        }
    }
}
```


Index

.p12 certificates 751

A

abort
 method 72
 activationRequired
 method 149, 249
 ActiveSync, installing and configuring 743
 addAppInstallationListener
 method 149
 addAppListener
 method 150
 addConnectionListener
 method 151
 addLogListener
 method 153
 addMenuItem
 method 212
 addMenuItemCollection
 method 154
 addMessageListener
 method 155
 addPushNotificationListener
 method 157
 Advanced Encryption Standard 696
 AES
 See also Advanced Encryption Standard
 AES-128 698
 AES-256 694
 Afaria® Security Manager 699
 Alert Message property 757
 alertDialogCallbackFunction
 method 73
 Alerts property 757
 Android emulator
 configuring 735
 Android Hybrid Web Container
 installing 735
 Android Hybrid Web Container customization
 setting HTTP headers 775
 ANDROID_CUSTOMIZATION_POINT_CATEG
 ORIZEDVIEWS 780
 ANDROID_CUSTOMIZATION_POINT_HYBRI
 DAPPSEARCH 794

anonymous
 namespace 68
 API functions
 credential functions 661
 general utility functions 661
 Hybrid App native device functions 661
 Hybrid App UI functions 661
 Hybrid App utility functions 661
 Hybrid App validation functions 661
 message data functions 661
 API.js 661
 APNS 754
 APNS Device Token property 757
 APP_ADDED
 member 126
 APP_REFRESH
 member 126
 APP_REMOVED
 member 127
 APP_UPDATED
 member 127
 AppInstallationListener
 method 73, 74
 Apple push notification properties 757
 Apple push notification, configuring 756
 application 719
 application ID
 guidelines 720
 ApplicationListener
 method 75
 AppLog
 namespace 84
 AppLogErrorCallbackParameter
 class 70
 arbitrary metadata 608
 ATTACHMENT_NOT_DOWNLOADED
 member 122
 AttachmentViewer control
 image limitations 604

B

BAD_OPTIONS
 member 142
 Badges property 757

Index

- BB7_MAX_STRING_STORAGE_LENGTH
 - member 117
 - BlackBerry 739
 - configuring the simulator 739, 740
 - BlackBerry Desktop Manager 738
 - BlackBerry Hybrid Web Container 739
 - adding a new language 804
 - adding a splash screen 802
 - BLACKBERRY_CUSTOMIZATION_POIN
T_SPLASHSCREEN 802
 - default behavior customization 810
 - setting HTTP headers 813
 - using custom colors 805
- ### C
- CACHE_FIRST
 - member 133
 - cached data lookup pattern
 - data flow diagram 644
 - overview 644
 - Callbacks.js 666
 - source file 253
 - CallbackSet 666
 - Camera.js
 - source file 257
 - categorized views 780
 - certificate picker 686
 - CERTIFICATE_NOT_SELECTED
 - member 122
 - Certificate.js 669
 - source file 268
 - CertificateFromAfaria
 - class 98
 - CertificateFromFile
 - class 99
 - CertificateFromStore
 - class 99
 - certificateLabels
 - method 158
 - certificates
 - for context variables 723
 - CertificateStore
 - method 158
 - class
 - AppLogErrorCallbackParameter 70
 - CertificateFromAfaria 98
 - CertificateFromFile 99
 - CertificateFromStore 99
 - sendRequestErrorCBParameter 70
 - sendRequestSuccessCBParameter 71
 - SUPStorage 116
 - SUPStorageException 120
 - clear
 - method 117
 - clearCache
 - method 164, 249
 - clearCacheItem
 - method 164, 249
 - ClientIconIndex 608
 - ClientVariables
 - method 165
 - ClientVariablesException
 - method 168
 - close
 - method 168
 - closeWorkflow
 - method 250
 - complete
 - method 76
 - conditional navigation 680
 - conditional start 682
 - CONNECTED
 - member 127
 - connection settings
 - configuring 746
 - default 835
 - device 746
 - Hybrid Web Container 746
 - CONNECTION_CONNECTED
 - member 128
 - CONNECTION_DISCONNECTED
 - member 128
 - CONNECTION_ERROR
 - member 128
 - CONNECTION_OTHER
 - member 129
 - CONNECTION_RETRIEVED_ITEMS
 - member 129
 - ConnectionSettings
 - method 168
 - ConnectionStateListener
 - method 77
 - connectToServer
 - method 170
 - content security 694
 - Android 694
 - BlackBerry 696
 - iOS 698

- Windows Mobile 699
- content type preference, changing 702
- context variables 608, 724
 - configuring 723
- convertLocalTimeToUtc
 - method 171
- convertUtcToLocalTime
 - method 171
- credential functions 661
- credentials, static and dynamic 684
- CredentialsCache 608
- Custom.js 653
- custom.js file
 - methods 671
- customAfterNavigateForward 671
- customAfterReportErrorFromNative 674
- customAfterShowScreen 671
- customAfterSubmit 671
- customAfterWorkflowLoad 671
- customBeforeMenuItemActivate 671
- customBeforeNavigateBackward 671
- customBeforeNavigateForward 671
- customBeforeReportErrorFromNative 674
- customBeforeShowScreen 671
- customBeforeSubmit 671
- customBeforeWorkflowLoad 671
- CustomIcon
 - method 172
- customization touch points
 - ANDROID_CUSTOMIZATION_POINT_DE
FAULTSETTINGS 772
- customValidateScreen 671

D

- data change notification 895, 896
 - GET 893
 - JSON format 893
 - POST 893
 - request response 896
- Datajs 43
- datajs library 64
- Datajs library 43
- DCN 896
- debugging 712
- default locale, creating 703
- DEFAULT_CUSTOM_ICON_INDEX
 - member 129
- defining an MBO
 - for cached data lookup 645

- for real-time data lookup 635
- deleteCertificateFromStore
 - method 100
- DeleteProcessedMessages 608
- Delivery Threshold property 757
- deploy 631
- device platforms 631
- device users
 - assigning Hybrid App packages 722
- DEVICE_NOT_CONNECTED
 - member 123
- devices
 - Apple push notification properties 757
- DISCONNECTED
 - member 130
- disconnectFromServer
 - method 175
- documentation roadmap 1
- Dynamic authentication 686

E

- e2eTrace
 - member 130
- editing
 - locale properties file 706
- Enable property 757
- encoding type, changing 702
- encryption key length 696
- encryption policy 677
- ERR_UNKNOWN
 - member 86
- errorCallbackFunction
 - method 77
- expireCredentials
 - method 175, 250
- ExternalResource.js
 - source file 283

F

- FAIL_TO_SAVE_CERTIFICATE
 - member 123
- FAIL_TO_SAVE_CREDENTIAL
 - member 123
- file association 702
- FILENAME_NO_EXTENSION
 - member 123

Index

- files
 - source 253
- filtering 838
- findByParameter
 - binding to a menu item 635
- findByParameter object query 639
- functions
 - resource 675
 - workflow UI 662

G

- general application properties 719
- general utility functions 661
- generated files 654, 655
- generateODataHttpClient
 - method 101
- genericCallbackFunction
 - method 78
- get
 - method 102
- getAllMessages
 - method 175
- getAppByID
 - method 177
- getAppIconUrl
 - method 177
- getApplicationConnectionID
 - method 178
- getBuiltInIconUrl
 - method 179
- getCallbackFromNativeError
 - method 180
- getClientVariables
 - method 180
- getCodeFromNativeError
 - method 181
- getCurrentApp
 - method 182
- getCurrentLocale
 - method 182
- getCurrentMessageValueCollection() 664
- getCustomIconUrl
 - method 183
- getDataMessage() 664
- getDefault
 - method 159
- getDstOffsetAtGivenTimeInMinutes
 - method 183
- getExternalResource
 - method 184

- getInstalledApps
 - method 185
- getItem
 - method 118
- getLocalizedDate
 - method 186
- getLocalizedDateTime
 - method 186
- getLocalizedTime
 - method 187
- getLogEntries
 - method 91, 187
- getLogEntriesErrorCallback
 - method 78
- getLogEntriesSuccessCallback
 - method 78
- getLoggingAlertDialog
 - method 188
- getLoggingCurrentLevel
 - method 188
- getMessageByID
 - method 189
- getMsgIconUrl
 - method 189
- getNativeMessageFromNativeError
 - method 190
- getOffsetFromUTC
 - method 191
- getOnErrorMessageFromNativeError
 - method 191
- getPicture 666
 - method 192
- getPublicCertificate
 - method 159
- getQueryVariable
 - method 193
- getServerInitiatedApps
 - method 193
- getSharedStorageKey
 - method 194
- getSignedCertificate
 - method 160
- getSignedCertificateFromAfaria
 - method 161
- getSignedCertificateFromFile
 - method 161
- getSignedCertificateFromServer
 - method 162

getTimezoneId
 method 194
 getTransformData
 method 195
 getUrl
 method 134
 getURLParamFromNativeError
 method 195
 getUsesDST
 method 195
 getXMLHttpRequest
 method 196, 250
 guid
 method 196, 250

H

hard coded credentials 724
 hideProgressDialog
 method 197
 HTML format 607
 HttpsConnection
 namespace 97
 hwc
 namespace 106
 hwc-api.js
 source file 294
 hwc-comms.js
 source file 432
 hwc-utils.js
 source file 490
 HWC.xcodeproj 828
 Hybrid App
 prepackaged, BlackBerry 858
 Hybrid App client
 using credentials 689
 Hybrid App clients
 and static SSO2 tokens 692
 and static X.509 certificates 690
 using credentials in 688
 using SSO2 tokens in 691
 Hybrid App native device functions 661
 Hybrid App package
 generated files 654
 Hybrid App Package Generation Wizard 631
 Hybrid App packages
 assigning device users 722
 configuring notification mailbox 721
 deploying 629
 Hybrid App UI functions 661

Hybrid App utility functions 661
 Hybrid App validation functions 661
 Hybrid Web Container
 Android 772
 ANDROID_CUSTOMIZATION_POINT_DE
 FAULTSETTINGS 772
 architecture 3
 building using source code 732
 customization 3, 772
 default values for settings screen 772
 installing from App Store 741
 management 3
 offline capabilities 3
 removing 758
 settings screen 772
 settings screen, default values 772
 HybridApp
 method 197
 hybridapp_Custom.html 655
 hybridapp_Custom.xml 628
 hybridapp_jQM.html 655
 hybridapp_JQM.xml 628
 hybridapp.html 656
 hybridapp.html generated file 655
 HybridApp.js 24
 HybridWebContainer.cod 739, 740

I

image
 limitations in Hybrid App messages 604
 IN_PROGRESS
 member 143
 INSTALLATION_BEGIN
 member 132
 INSTALLATION_END
 member 132
 installing 741
 internationalization
 Hybrid App Designer 707
 on the device 709
 INVALID_COMMON_NAME
 member 124
 InvokeOnClient 608
 iOS 740
 iOS Hybrid Web Container
 client certificate challenge 843
 customizations 830
 settings screen 837

Index

- iOS Hybrid Web Container customization 833
 - filtering 838
 - setting HTTP headers 841
 - sorting 838
- iOS push notification properties 757
- IOS_CUSTOMIZATION_POINT 828
- iPad
 - hiding the listview 842
- isAndroid
 - method 201
- isAndroid3
 - method 201
- isBlackBerry
 - method 201
- isBlackBerry5
 - method 202
- isBlackBerry5WithTouchScreen
 - method 202
- isBlackBerry6NonTouchScreen
 - method 202
- isBlackBerry7
 - method 203
- isClosed
 - method 203
- isDstActiveAtGivenTime
 - method 204
- isEnabled
 - method 141
- isIOS
 - method 204
- isIOS4
 - method 205
- isIOS5
 - method 205
- isIOS6
 - method 205
- isIOS7
 - method 206
- isIPad
 - method 206
- ISO-8859-1 encoding 702
- isSharedStorageEnabled
 - method 206
- isTraceEnabled
 - method 130
- isWindows
 - method 207
- isWindowsMobile
 - method 207

- ITEM_NOT_FOUND
 - member 165
- iTunes 742

J

- jquery.mobile-1.1.0.css 656

K

- key
 - method 118

L

- length
 - method 119
- listAvailableCertificatesFromFileSystem
 - method 163
- load arguments 635
- loadSettings
 - method 207
- locale
 - editing 706
 - properties file 703, 706
- localization 653, 700
 - creating a new locale 703
 - Hybrid App package 701
 - limitations 701
 - task flow 701
 - updating the current locale 706
- log
 - method 208
- LogEntry
 - method 92, 209
- logListener
 - method 79
- LogListener
 - method 79
- logToWorkflow
 - method 250
- look and feel 628
- look and feel files 656

M

- manage 717
- manifest.xml 608

- markAsActivated
 - method 211, 251
- markAsProcessed
 - method 211, 251
- MarkProcessedMessages 608
- master.css 656
- matching rules
 - specifying 641
- MediaCache
 - member 133
- member
 - APP_ADDED 126
 - APP_REFRESH 126
 - APP_REMOVED 127
 - APP_UPDATED 127
 - ATTACHMENT_NOT_DOWNLOADED 122
 - BAD_OPTIONS 142
 - BB7_MAX_STRING_STORAGE_LENGTH 117
 - CACHE_FIRST 133
 - CERTIFICATE_NOT_SELECTED 122
 - CONNECTED 127
 - CONNECTION_CONNECTED 128
 - CONNECTION_DISCONNECTED 128
 - CONNECTION_ERROR 128
 - CONNECTION_OTHER 129
 - CONNECTION_RETRIEVED_ITEMS 129
 - DEFAULT_CUSTOM_ICON_INDEX 129
 - DEVICE_NOT_CONNECTED 123
 - DISCONNECTED 130
 - e2eTrace 130
 - ERR_UNKNOWN 86
 - FAIL_TO_SAVE_CERTIFICATE 123
 - FAIL_TO_SAVE_CREDENTIAL 123
 - FILENAME_NO_EXTENSION 123
 - IN_PROGRESS 143
 - INSTALLATION_BEGIN 132
 - INSTALLATION_END 132
 - INVALID_COMMON_NAME 124
 - ITEM_NOT_FOUND 165
 - MediaCache 133
 - MSG_ADDED 135
 - MSG_PRIORITY_HIGH 135
 - MSG_PRIORITY_NORMAL 136
 - MSG_REFRESH 136
 - MSG_REMOVED 136
 - MSG_UPDATED 137
 - NAVIGATION_ERROR 124
 - NO_ERROR 143
 - NOT_SUPPORTED 143
 - NOTIFICATION_CANCEL 137
 - NOTIFICATION_CONTINUE 137
 - OPEN_APP_NOT_EXIST 138
 - OPEN_APP_OTHER 138
 - OPEN_APP_SUCCESS 138
 - OPEN_MSG_APP_NOT_EXIST 139
 - OPEN_MSG_NOT_EXIST 139
 - OPEN_MSG_OTHER 139
 - OPEN_MSG_SUCCESS 140
 - options 72
 - perf 140
 - PictureError 142
 - PictureOptions 72
 - Policy 133
 - REG_ERR_AUTO_REG_NOT_ENABLED 144
 - REG_ERR_AUTO_REG_TEMPLATE_NOT_FOUND 145
 - REG_ERR_AUTO_REG_USER_NAME_TOO_LONG 145
 - REG_ERR_AUTO_REG_WRONG_USER_FOR_DEVICE 145
 - REG_ERR_COULD_NOT_REACH_MMS_SERVER 146
 - REG_ERR_INVALID_USER_NAME 146
 - REG_ERR_MMS_AUTHENTICATION_FAILED 146
 - REGISTRATION_METHOD_AFARIA 147
 - REGISTRATION_METHOD_AUTOMATIC 147
 - REGISTRATION_METHOD_CERTIFICATE 147
 - REGISTRATION_METHOD_MANUAL 148
 - REGISTRATION_METHOD_NO_PREFERENCE 148
 - REQUIRED_PARAMETER_NOT_AVAILABLE 124
 - RESPONSE_TOO_LARGE 125
 - SERVER_FIRST 133
 - SETTING_SUCCESS 148
 - SSOCERT_EXCEPTION 125
 - STATUS 149
 - STATUS_EVENT_CONNECTED 86
 - STATUS_EVENT_DISCONNECTED 86
 - STATUS_EVENT_DISCONNECTED_LOW_STORAGE 87

Index

- STATUS_EVENT_DISCONNECTED_ROAMING 87
- STATUS_EVENT_FLIGHT_MODE 87
- STATUS_EVENT_NOTIFICATION_RECEIVED 88
- STATUS_EVENT_OUT_OF_NETWORK 88
- STATUS_EVENT_REGISTRATION_STARTED 88
- STATUS_EVENT_RESTART 89
- STATUS_EVENT_SET_DEFAULT_ITEM 89
- STATUS_EVENT_SHUTDOWN 89
- STATUS_EVENT_STARTUP 90
- STATUS_EVENT_UNKNOWN 90
- STATUS_EVENT_UNSET_DEFAULT_ITEM 90
- STATUS_EVENT_WAITING_TO_CONNECT 91
- TOO_LARGE 143
- UNKNOWN 144
- UNKNOWN_ERROR 125
- UNKNOWN_MIME_TYPE 125
- UNSUPPORTED_ATTACHMENT_TYPE 126
- USER_REJECT 144
- MenuItemCollection
 - method 212
- Message
 - method 215
- message data functions 661
- MessageFilter
 - method 221
- MessageListener
 - method 80
- method
 - abort 72
 - activationRequired 149, 249
 - addAppInstallationListener 149
 - addAppListener 150
 - addConnectionListener 151
 - addLogListener 153
 - addMenuItem 212
 - addMenuItemCollection 154
 - addMessageListener 155
 - addPushNotificationListener 157
 - alertDialogCallbackFunction 73
 - AppInstallationListener 73, 74
 - ApplicationListener 75
 - certificateLabels 158
 - CertificateStore 158
 - clear 117
 - clearCache 164, 249
 - clearCacheItem 164, 249
 - ClientVariables 165
 - ClientVariablesException 168
 - close 168
 - closeWorkflow 250
 - complete 76
 - ConnectionSettings 168
 - ConnectionStateListener 77
 - connectToServer 170
 - convertLocalTimeToUtc 171
 - convertUtcToLocalTime 171
 - CustomIcon 172
 - deleteCertificateFromStore 100
 - disconnectFromServer 175
 - errorCallbackFunction 77
 - expireCredentials 175, 250
 - generateODataHttpClient 101
 - genericCallbackFunction 78
 - get 102
 - getAllMessages 175
 - getAppByID 177
 - getAppIconUrl 177
 - getApplicationConnectionID 178
 - getBuiltInIconUrl 179
 - getCallbackFromNativeError 180
 - getClientVariables 180
 - getCodeFromNativeError 181
 - getCurrentApp 182
 - getCurrentLocale 182
 - getCustomIconUrl 183
 - getDefault 159
 - getDstOffsetAtGivenTimeInMinutes 183
 - getExternalResource 184
 - getInstalledApps 185
 - getItem 118
 - getLocalizedDate 186
 - getLocalizedDateTime 186
 - getLocalizedTime 187
 - getLogEntries 91, 187
 - getLogEntriesErrorCallback 78
 - getLogEntriesSuccessCallback 78
 - getLoggingAlertDialog 188
 - getLoggingCurrentLevel 188
 - getMessageByID 189
 - getMsgIconUrl 189
 - getNativeMessageFromNativeError 190

- getOffsetFromUTC 191
- getOnErrorMessageFromNativeError 191
- getPicture 192
- getPublicCertificate 159
- getQueryVariable 193
- getServerInitiatedApps 193
- getSharedStorageKey 194
- getSignedCertificate 160
- getSignedCertificateFromAfaria 161
- getSignedCertificateFromFile 161
- getSignedCertificateFromServer 162
- getTimezoneId 194
- getTransformData 195
- getUrl 134
- getURLParamFromNativeError 195
- getUsesDST 195
- getXMLHttpRequest 196, 250
- guid 196, 250
- hideProgressDialog 197
- HybridApp 197
- isAndroid 201
- isAndroid3 201
- isBlackBerry 201
- isBlackBerry5 202
- isBlackBerry5WithTouchScreen 202
- isBlackBerry6NonTouchScreen 202
- isBlackBerry7 203
- isClosed 203
- isDstActiveAtGivenTime 204
- isEnabled 141
- isIOS 204
- isIOS4 205
- isIOS5 205
- isIOS6 205
- isIOS7 206
- isIPad 206
- isSharedStorageEnabled 206
- isTraceEnabled 130
- isWindows 207
- isWindowsMobile 207
- key 118
- length 119
- listAvailableCertificatesFromFileSystem 163
- loadSettings 207
- log 208
- LogEntry 92, 209
- logListener 79
- LogListener 79
- logToWorkflow 250
- markAsActivated 211, 251
- markAsProcessed 211, 251
- MenuItemCollection 212
- Message 215
- MessageFilter 221
- MessageListener 80
- onGetPictureError 81
- onGetPictureSuccess 82
- openApp 221
- openMessage 222
- processDataMessage 251
- processWorkflowMessage 252
- removeAllMenuItems 223
- removeAppInstallationListener 223
- removeAppListener 224
- removeConnectionListener 226
- removeItem 119
- removeLogListener 227
- removeMessage 229
- removeMessageListener 229
- removePushNotificationListener 231
- sample_AppListener 232
- sample_ConnectionListener 233
- sample_InstallationAppListener 234
- sample_LogListener 234
- sample_MessageListener 235
- sample_PushNotificationListener 235
- saveLoginCertificate 236, 252
- saveLoginCredentials 236, 252
- saveSettings 237
- sendRequest 104
- sendRequestErrorCB 82
- sendRequestSuccessCB 83
- setItem 120
- setLoggingAlertDialog 238
- setLoggingCurrentLevel 239
- setOKAction 213
- setReportErrorFromNativeCallback 239
- setScreenTitle_CONT 240
- setSubMenuName 214
- setTraceLevel 131
- SharedStorage 240
- showAlertDialog 240
- showAttachmentContents_CONT 241
- showAttachmentFromCache_CONT 242
- showCertificatePicker 243, 253
- showConfirmDialog 243
- showLocalAttachment 244
- showProgressDialog 244

Index

- showUrlInBrowser 245, 253
- shutdown 246
- startClient 246
- startInteraction 141
- startInterval 141
- startLogListener 92
- startOrStopLogListenerErrorCallback 83
- startOrStopLogListenerSuccessCallback 84
- startTrace 131
- stopInteraction 142
- stopInterval 142
- stopLogListener 95
- stopTrace 131
- stringify 215
- this.containsName 165
- this.getAllVariableNames 166
- this.getClientVariables 198
- this.getCount 166
- this.getCustomIconList 198
- this.getDate 210
- this.getDefaultCustomIcon 199
- this.getDisplayName 199
- this.getEventID 210
- this.getHeight 173
- this.setIconIndex 200, 216
- this.setIconUrl 247
- this.getImagePath 173
- this.getMessage 211
- this.getMessageId 216
- this.getModuleId 217
- this.getModuleID 200
- this.getModuleVersion 217
- this.getName 173
- this.getPriority 217
- this.getProcessedImagePath 174
- this.getReceivedDate 218
- this.getSender 218
- this.getSubject 219
- this.getType 174
- this.getVariableValueByName 167
- this.getVersion 167, 200
- this.getWidth 174
- this.isProcessed 219
- this.isRead 219
- this.updateProcessed 220
- this.updateRead 220
- updateMessageProcessed 248
- updateMessageRead 248
- uploadTrace 132

- Microsoft ActiveSync, installing and configuring 743
- ModuleDesc 608
- ModuleDisplayName 608
- ModuleName 608
- ModuleVersion 608
- MSG_ADDED
 - member 135
- MSG_PRIORITY_HIGH
 - member 135
- MSG_PRIORITY_NORMAL
 - member 136
- MSG_REFRESH
 - member 136
- MSG_REMOVED
 - member 136
- MSG_UPDATED
 - member 137

N

- namespace
 - anonymous 68
 - AppLog 84
 - HttpsConnection 97
 - hwc 106
 - NativeErrorCodes 121
- native device functions 663
- NativeErrorCodes
 - namespace 121
- NAVIGATION_ERROR
 - member 124
- network edge authentication 688
- NO_ERROR
 - member 143
- non HTTP authentication request 895
- non-ASCII encoding 702
- NOT_SUPPORTED
 - member 143
- notification mailbox 721
- NOTIFICATION_CANCEL
 - member 137
- NOTIFICATION_CONTINUE
 - member 137
- notifications
 - creating 39
- null value support 600

O

- object queries
 - binding to a menu item 646
- object query parameters
 - defining a control that passes 647
- OData 43
- offline capabilities 3
- onGetPictureError
 - method 81
- onGetPictureSuccess
 - method 82
- OPEN_APP_NOT_EXIST
 - member 138
- OPEN_APP_OTHER
 - member 138
- OPEN_APP_SUCCESS
 - member 138
- OPEN_MSG_APP_NOT_EXIST
 - member 139
- OPEN_MSG_NOT_EXIST
 - member 139
- OPEN_MSG_OTHER
 - member 139
- OPEN_MSG_SUCCESS
 - member 140
- openApp
 - method 221
- openMessage
 - method 222
- Optimize for appearance look and feel 655
- Optimize for performance look and feel 655, 659
- options
 - member 72
- OTA 739
- over the air 739

P

- perf
 - member 140
- performance agent 725, 726
- PersistAppDomain 608
- PersistentContent 696
- PersistentContentListener 696
- PersistentStore 696
- PhoneGap 864
 - initializing 886
 - supported APIs 864
- Phonegap HTTPS proxy connection properties 888

- PhoneGap plugin
 - testing 878
- PictureError
 - member 142
- PictureOptions
 - member 72
- PIN screens
 - CreatePasswordViewController.xib 833
 - customizing 833
 - EnterPasswordViewController.xib 833
 - iOS 833
- PlatformIdentification.js
 - source file 500
- Plugins/AppLog/applog.js
 - source file 513
- Plugins/HttpsProxy/dataajs-https-proxy.js
 - source file 532
- Plugins/HttpsProxy/https-proxy.js
 - source file 538
- Policy
 - member 133
- preferences
 - appearance 702
 - content types 702
 - general 702
- processDataMessage
 - method 251
- ProcessUpdates 608
- processWorkflowMessage
 - method 252
- properties
 - push notification for iOS 757
- propagate to attributes 635
- proxy connection 44
- PurchaseOrderSample 703
- push notification properties for iOS 757

Q

- query types
 - addallmenuitems 602
 - addMenuItem 602
 - alert 602
 - clearrequestscache 602
 - clearrequestscacheitem 602
 - close 602
 - downloadattachment 602
 - formredirect 602
 - loadtransformdata 602
 - logtoworkflow 602

Index

- removeallmenuitems 602
 - rmi 602
 - setscreentitle 602
 - showattachment 602
 - showcertpicker 602
 - showInBrowser 602
 - showlocalattachment 602
 - submit 602
- R**
- real-time lookup pattern
 - data flow diagram 634
 - overview 634
 - REG_ERR_AUTO_REG_NOT_ENABLED
 - member 144
 - REG_ERR_AUTO_REG_TEMPLATE_NOT_FO
UND
 - member 145
 - REG_ERR_AUTO_REG_USER_NAME_TOO_L
ONG
 - member 145
 - REG_ERR_AUTO_REG_WRONG_USER_FOR_
DEVICE
 - member 145
 - REG_ERR_COULD_NOT_REACH_MMS_SER
VER
 - member 146
 - REG_ERR_INVALID_USER_NAME
 - member 146
 - REG_ERR_MMS_AUTHENTICATION_FAILED
 - member 146
 - REGISTRATION_METHOD_AFARIA
 - member 147
 - REGISTRATION_METHOD_AUTOMATIC
 - member 147
 - REGISTRATION_METHOD_CERTIFICATE
 - member 147
 - REGISTRATION_METHOD_MANUAL
 - member 148
 - REGISTRATION_METHOD_NO_PREFERENC
E
 - member 148
 - removeAllMenuItems
 - method 223
 - removeAppInstallationListener
 - method 223
 - removeAppListener
 - method 224
 - removeConnectionListener
 - method 226
 - removeItem
 - method 119
 - removeLogListener
 - method 227
 - removeMessage
 - method 229
 - removeMessageListener
 - method 229
 - removePushNotificationListener
 - method 231
 - REQUIRED_PARAMETER_NOT_AVAILABLE
 - member 124
 - RequiresActivation 608
 - resource functions 675
 - RESPONSE_TOO_LARGE
 - member 125
 - rmi.xml 712
 - RSA algorithm 751
- S**
- sample_AppListener
 - method 232
 - sample_ConnectionListener
 - method 233
 - sample_InstallationAppListener
 - method 234
 - sample_LogListener
 - method 234
 - sample_MessageListener
 - method 235
 - sample_PushNotificationListener
 - method 235
 - SAP passport 725
 - saveLoginCertificate
 - method 236, 252
 - saveLoginCredentials
 - method 236, 252
 - saveSettings
 - method 237
 - send a notification 711
 - sending server notification to a device 643
 - sendRequest
 - method 104
 - sendRequestErrorCB
 - method 82
 - sendRequestErrorCBParameter
 - class 70
 - sendRequestSuccessCB
 - method 83

- sendRequestSuccessCBParameter
 - class 71
- server notification pattern 639
 - creating an MBO for 639
 - data flow diagram 639
 - overview 639
- SERVER_FIRST
 - member 133
- server-driven notification
 - creating 641
- server-initiated 39
- setItem
 - method 120
- setLoggingAlertDialog
 - method 238
- setLoggingCurrentLevel
 - method 239
- setOKAction
 - method 213
- setReportErrorFromNativeCallback
 - method 239
- setScreenTitle_CONT
 - method 240
- setSubMenuName
 - method 214
- SETTING_SUCCESS
 - member 148
- setTraceLevel
 - method 131
- shared storage 678
- SharedStorage 678
 - method 240
- showAlertDialog
 - method 240
- showAttachmentContents_CONT
 - method 241
- showAttachmentFromCache_CONT
 - method 242
- showCertificatePicker
 - method 243, 253
- showConfirmDialog
 - method 243
- showErrorFromNative 674
- showLocalAttachment
 - method 244
- showProgressDialog
 - method 244
- showUrlInBrowser
 - method 245, 253
- shutdown
 - method 246
- signing key 731
- single sign-on 688
 - using credentials 688
 - using SSO2 tokens 691
 - using static SSO2 tokens 692
 - using static X.509 certificates 690
- single sign-on task flow 893
- sorting 838
- Sounds property 757
- source
 - files 253
- source file
 - Callbacks.js 253
 - Camera.js 257
 - Certificate.js 268
 - ExternalResource.js 283
 - hwc-api.js 294
 - hwc-comms.js 432
 - hwc-utils.js 490
 - PlatformIdentification.js 500
 - Plugins/AppLog/applog.js 513
 - Plugins/HttpsProxy/dataajs-https-proxy.js 532
 - Plugins/HttpsProxy/https-proxy.js 538
 - SUPStorage.js 560
 - Timezone.js 573
- SQLite Encryption Extensions (AES-128) 698
- SSOCERT_EXCEPTION
 - member 125
- startClient
 - method 246
- startInteraction
 - method 141
- startInterval
 - method 141
- startLogListener
 - method 92
- startOrStopLogListenerErrorCallback
 - method 83
- startOrStopLogListenerSuccessCallback
 - method 84
- startTrace
 - method 131
- static authentication 685
- STATUS
 - member 149
- STATUS_EVENT_CONNECTED
 - member 86

Index

- STATUS_EVENT_DISCONNECTED
 - member 86
- STATUS_EVENT_DISCONNECTED_LOW_STORAGE
 - member 87
- STATUS_EVENT_DISCONNECTED_ROAMING
 - member 87
- STATUS_EVENT_FLIGHT_MODE
 - member 87
- STATUS_EVENT_NOTIFICATION_RECEIVED
 - member 88
- STATUS_EVENT_OUT_OF_NETWORK
 - member 88
- STATUS_EVENT_REGISTRATION_STARTED
 - member 88
- STATUS_EVENT_RESTART
 - member 89
- STATUS_EVENT_SET_DEFAULT_ITEM
 - member 89
- STATUS_EVENT_SHUTDOWN
 - member 89
- STATUS_EVENT_STARTUP
 - member 90
- STATUS_EVENT_UNKNOWN
 - member 90
- STATUS_EVENT_UNSET_DEFAULT_ITEM
 - member 90
- STATUS_EVENT_WAITING_TO_CONNECT
 - member 91
- stopInteraction
 - method 142
- stopInterval
 - method 142
- stopLogListener
 - method 95
- stopTrace
 - method 131
- stringify
 - method 215
- strings.xml 767
- stylesheet.css 656
- SupCertificateIssuer 725
- SupCertificateNotAfter 725
- SupCertificateNotBefore 725
- SupCertificateSubject 725
- SUPMessaging_Pro.cab 744
- SUPMobileHybridApp.replaceHybridAppCertificate() 693
- SupPassword 724
 - for context variables 723
- SUPStorage 677
 - class 116
- SUPStorage.js 676
 - source file 560
- SUPStorageException
 - class 120
- SupUser 724, 725
 - for context variables 723
 - synchronization software 743

T

- task flow 7
- testing
 - X.509 certificates 751
- this.containsName
 - method 165
- this.getAllVariableNames
 - method 166
- this.getClientVariables
 - method 198
- this.getCount
 - method 166
- this.getCustomIconList
 - method 198
- this.getDate
 - method 210
- this.getDefaultCustomIcon
 - method 199
- this.getDisplayName
 - method 199
- this.getEventID
 - method 210
- this.getHeight
 - method 173
- this.getIconIndex
 - method 200, 216
- this.getIconUrl
 - method 247
- this.getImagePath
 - method 173
- this.getMessage
 - method 211
- this.getMessageId
 - method 216
- this.getModuleId
 - method 217

this.getModuleID
 method 200
 this.getModuleVersion
 method 217
 this.getName
 method 173
 this.getPriority
 method 217
 this.getProcessedImagePath
 method 174
 this.getReceivedDate
 method 218
 this.getSender
 method 218
 this.getSubject
 method 219
 this.getType
 method 174
 this.getVariableValueByName
 method 167
 this.getVersion
 method 167, 200
 this.getWidth
 method 174
 this.isProcessed
 method 219
 this.isRead
 method 219
 this.updateProcessed
 method 220
 this.updateRead
 method 220
 Timezone.js
 source file 573
 TOO_LARGE
 member 143
 touch point 828
 trace 725
U
 UNKNOWN
 member 144

UNKNOWN_ERROR
 member 125
 UNKNOWN_MIME_TYPE
 member 125
 UNSUPPORTED_ATTACHMENT_TYPE
 member 126
 updateMessageProcessed
 method 248
 updateMessageRead
 method 248
 upgrading the PhoneGap library 797, 827
 uploadTrace
 method 132
 URL parameters 42
 USER_REJECT
 member 144
 UTF-8 encoding 702

V

variables, context
 configuring 723
 viewing Hybrid App messages
 Android 711
 BlackBerry 711
 iOS 711
 Windows Mobile 711

W

whitelisting 44
 WorkflowClient.xml 39, 617

X

X.509 certificate 754

