



**Developer Guide: Mobile Workflow
Packages**

**Sybase Unwired Platform 2.1
ESD #2**

DOCUMENT ID: DC01218-01-0212-01

LAST REVISED: January 2012

Copyright © 2012 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 Mobile Workflow	
Packages	1
Documentation Roadmap for Unwired Platform	1
Introduction to Developing Mobile Workflow	
Applications With Sybase Unwired Platform	3
Hybrid Web Container Architecture	3
Hybrid Web Container Development Task Flow	6
Identify a Business Process for Workflow	
Development	6
Hybrid Web Container Patterns	8
Online Lookup	9
Server Notification	14
Cached Data	19
Mobile Workflow Application Configuration for Data	
Change Notification	28
Extending Data Change Notification to Mobile	
Workflow Clients	28
Non HTTP Authentication Workflow DCN	
Request	30
Sending Workflow DCN to Users Regardless of	
Individual Security Configurations	30
Mobile Workflow DCN Request Response	31
Workflow DCN Design Approach and Sample	
Code	31
Mobile Workflow Development	37
Develop a Mobile Workflow Using the Mobile Workflow	
Forms Editor	37
The Mobile Workflow Forms Editor	37
Deploy the Mobile Workflow Package to Unwired	
Server	74

Generating the Files for a Mobile Workflow	
Package	74
Deployment Modes	76
Install and Configure the Mobile Workflow Container	
On the Device	77
Preparing Android Devices for the Mobile	
Workflow Package	77
Preparing iOS Devices for the Mobile Workflow	
Package	80
Installing the Mobile Workflow Container on	
Windows Mobile Devices	90
Preparing BlackBerry Devices for the Mobile	
Workflow Package	92
Configure Connection Settings on the Device	93
Install and Test Certificates on Simulators and	
Devices	97
Manage a Mobile Workflow Package	103
Registering and Reregistering Mobile Workflow	
Application Connections	103
Enabling and Configuring the Notification	
Mailbox	104
Assigning and Unassigning Mobile Workflows .	105
Activating the Workflow	106
Configuring Context Variables for Mobile	
Workflow Packages	106
Security	108
Credentials	108
Configuring the Workflow Application to Use	
Credentials	112
Content Security on Devices	118
Localization and Internationalization	123
Localization Limitations	124
Localizing a Mobile Workflow Package	125
Mobile Workflow Package Internationalization .	131
Internationalization on the Device	132

Test Mobile Workflow Packages	133
Testing Server Initiated Mobile Workflow Packages	134
Launching a Server-initiated Mobile Workflow on the Device	137
Debugging Custom Code	138
Mobile Workflow Package Customization	139
Adding Custom Code	140
Generated Mobile Workflow Files	140
Reference	146
Using Third-Party JavaScript Files	193
Repackaging Mobile Workflow Package Files ...	193
Common Customizations	194
Create a Mobile Workflow Package Manually	198
Mobile Workflow URL Parameters	198
Calling the Hybrid Web Container	199
Mobile Workflow Package Files	201
Using Third-party Files	222
Troubleshoot	223
HTTP Error Codes	223
Recovering from EIS Errors	224
Mapping of EIS Codes to Logical HTTP Error Codes ..	225
Credentials Are Lost after User Successfully Passes Activation Screen	226
Mobile Workflow Exception Handling	226
Unable to Deploy Workflow	227
Index	229

Introduction to Developer Guide for Mobile Workflow Packages

This developer guide provides information about using Sybase® Unwired Platform features to create Mobile Workflow packages. The audience is Mobile Workflow developers.

This guide describes requirements for developing a Mobile Workflow package, how to generate Mobile Workflow package code, and how to deploy the Mobile Workflow package to the device or simulator.

Companion guides include:

- *Sybase Unwired WorkSpace – Mobile Business Object Development*
- *Sybase Unwired WorkSpace – Mobile Workflow Package Development*
- *Tutorial: Mobile Workflow Package Development*
- *Troubleshooting for Sybase Unwired Platform*

Documentation Roadmap for Unwired Platform

Sybase® Unwired 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. *Fundamentals* is available on the Sybase Product Documentation Web site.

Check the Sybase Product Documentation Web site regularly for updates: access <http://sybooks.sybase.com/nav/summary.do?prod=1289>, then navigate to the most current version.

Introduction to Developing Mobile Workflow Applications With Sybase Unwired Platform

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

Within Sybase Unwired Platform, development tools enable both aspects of Mobile Workflow application development:

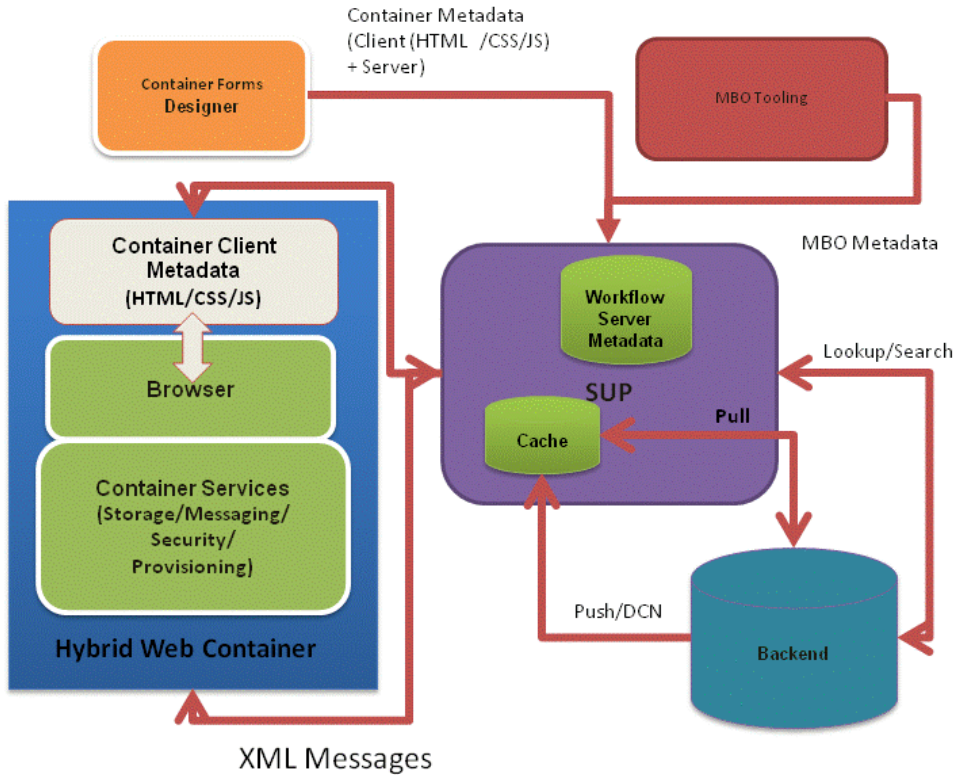
- The data aspects of the Mobile Workflow application are called mobile business objects (MBO), and “MBO development” refers to defining object data models with back-end enterprise information system (EIS) connections, attributes, operations, and relationships. Mobile Workflow applications can reference one or more MBOs and can include load parameters, personalization, and error handling.
- Once you have developed MBOs and deployed them to Unwired Server, develop device-resident presentation and logic for your Mobile Workflow application using the Mobile Workflow Forms Editor.

Note: See *Sybase Unwired WorkSpace – Mobile Business Object Development* for procedures and information about creating and deploying MBOs.

Hybrid Web Container Architecture

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

The Hybrid Web Container is a native application that 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. The Hybrid Web Container enables the rapid development of mobile workflows, in which you can extend existing enterprise business processes, to a mobile device so that business process decisions can be made on a mobile device.



Mobile Workflow Forms Editor

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

Mobile Workflow package files are generated using the Mobile Workflow Package generation wizard in the Mobile Workflow Forms editor. The generated Mobile Workflow 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 Mobile Workflow 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.

Mobile Workflow packages can be deployed to Unwired Server and assigned to users using the Mobile Workflow Forms Editor in Eclipse.

Customization

You can modify certain files in the generated Mobile Workflow package to customize application behavior.

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

- **HTML** – HTML files are generated in the Mobile Workflow Forms editor. The files that are generated 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 Mobile Workflow deployment tool. The Mobile Workflow Forms editor 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 `customBeforeWorkflowLoad()`. Additional custom JavaScript files can be added to the Mobile Workflow package in the Eclipse WorkSpace.
- **CSS** – the Hybrid Web Container uses a 3rd-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 Mobile Workflow Packages in Eclipse and manage them through the Sybase Control Center console. No device interaction is required from the administrator. Once a Mobile Workflow package is deployed into an existing installation, the administrator can configure the Mobile Workflow package and assign it to any active user in the system.

Offline Capabilities

Server-initiated notifications extract data from the backend and Unwired 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 Workflow 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.

Hybrid Web Container Development Task Flow

Developing a Hybrid Web Container 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 Sybase Control Center (SCC).

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

4. Use the Mobile Workflow Forms Editor to create a new Hybrid Web Container.

Note: Optionally, you can create a Hybrid Web Container manually, however, using the Mobile Workflow Forms Editor, automates many tasks and provides integration across different device platforms.

5. Use the Mobile Workflow Forms Editor 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. Use the Mobile Workflow Package Generation wizard to generate the Hybrid Web Container files.
8. (Optional) Customize the generated `Custom.js` file.
9. (Optional) If you customized the Hybrid Web Container files, re-generate the files using the Mobile Workflow Package Generation Wizard.
10. Deploy the Mobile Workflow package to Unwired Server.
11. Install and configure the Hybrid Web Container on the device or simulator.
12. In SCC, assign the Mobile Workflow application to the device user.
13. On the device or simulator, run, test and debug the Mobile Workflow application.

Note: See *Sybase Unwired WorkSpace – Mobile Business Object Development* for procedures and information about creating and deploying MBOs.

Identify a Business Process for Workflow Development

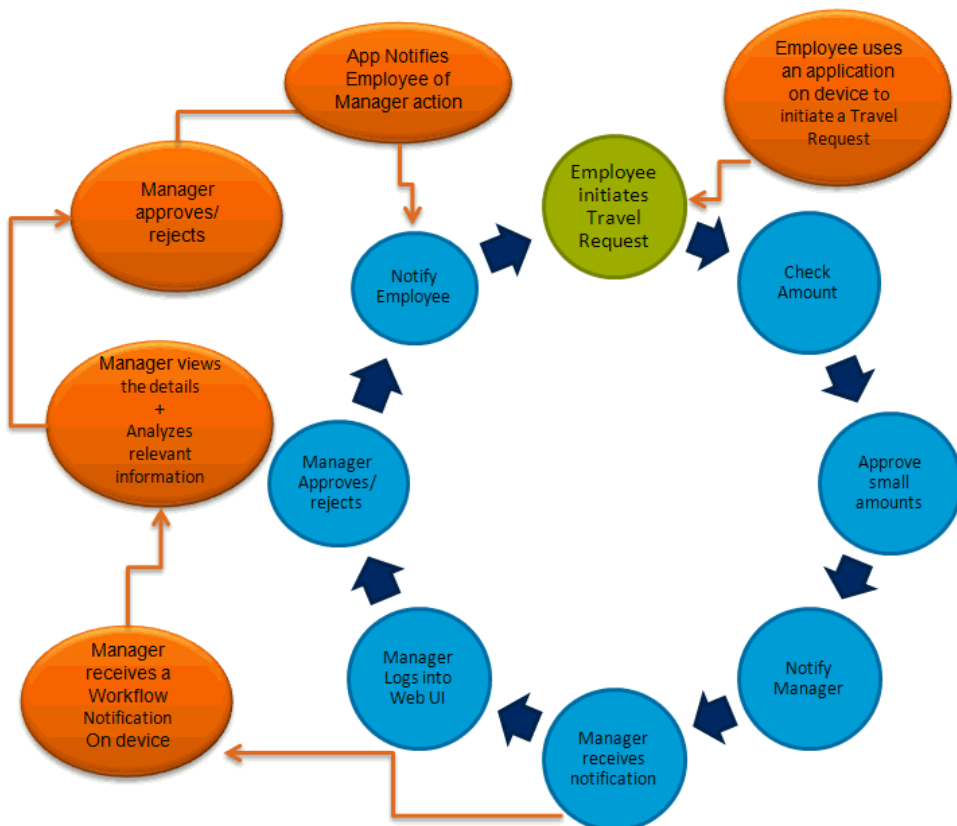
The first step is identifying whether a workflow package can implement a decision point in a particular business process.

Workflow packages 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 Workflow package allows creating a mobile business object (MBO) and sending it to the Unwired Server, or retrieving an MBO from the Unwired Server and displaying that information in a decision step. A more complex Workflow package 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 mobile workflow 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.

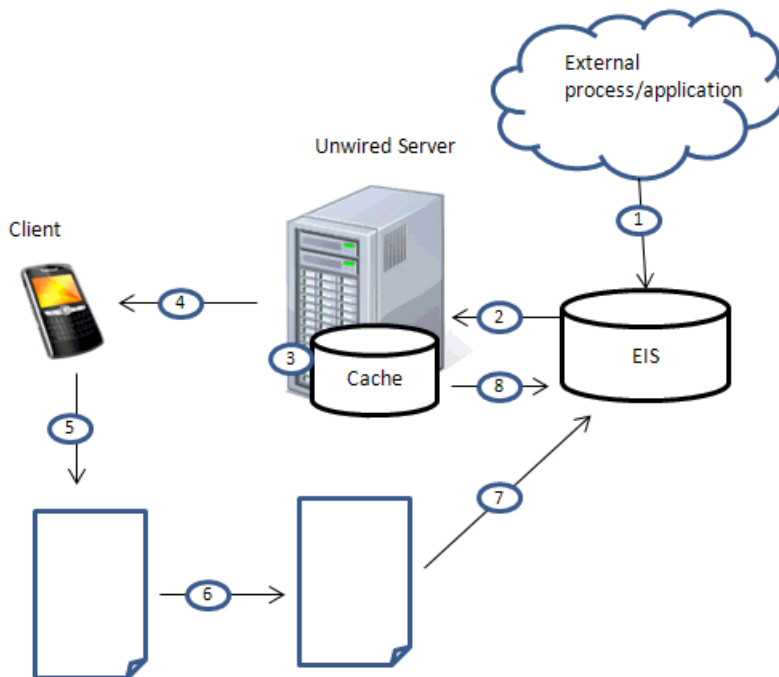


Hybrid Web Container Patterns

The Hybrid Web container allows you to create lightweight applications that implement various business solutions. These are some of the primary Hybrid Web container and the Unwired Platform patterns (models):

- Server notification – the enterprise information system (EIS) notifies SUP of data changes and SUP sends notifications to subscribed devices based on the rules.
- Online lookup – the client retrieves data directly from the EIS. This pattern typically uses a client-initiated starting point.
- Cached data – the client retrieves data from the Unwired 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:

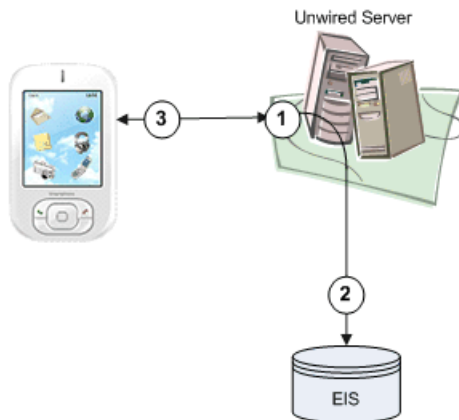


1. An external process or application updates EIS data.

2. The changed data triggers a data change notification (DCN), which is sent to Unwired Server, or a message from another workflow client updates mobile business object (MBO) data contained on Unwired Server.
3. The DCN could be programmed to update MBO data.
4. Unwired 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 Unwired Server.
8. Unwired Server updates the EIS.

Online Lookup

This pattern provides direct interaction between the data requester (workflow client) and the enterprise information system (EIS), supplying real-time EIS data rather than 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 workflow 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, Unwired 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 workflow client is triggered when the user selects the **Submit** menu item that calls the `findByParameter` object query.

Implementing Online Lookup for Workflow Clients

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 workflow application that calls the findByParameter object query to return real-time results from the EIS.

Defining Mobile Workflow 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 Unwired Workspace, create an MBO that has at least one load argument. For example, you could define an Emp MBO as:

```
SELECT    id,
          empName,
          empDeptId FROM sampledb.dba.employee
WHERE empDeptId = :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 Mobile Workflow 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 Unwired 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.

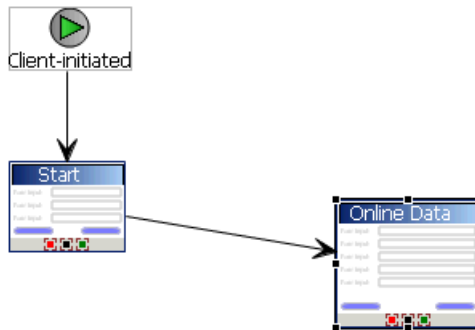
Prerequisites

You must have propagate-to attributes mapped to MBO load parameters, and the deployed MBO must use an Online cache group policy. Unwired Platform services must be running.

Task

1. From Unwired Workspace, launch the Mobile Workflow Forms Editor.
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.



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`.

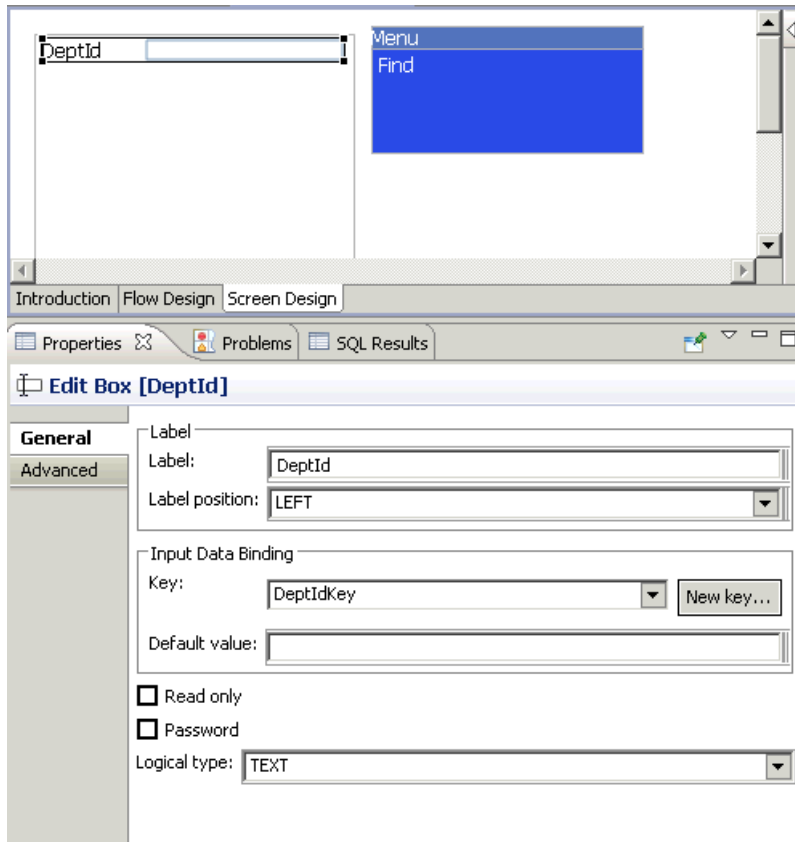
Unmapped parameters can get their value from the default value, if specified, or from the personalization key value they are mapped to, if that is specified. If the key is unmapped, and the parameter has no default value and is not mapped to a personalization key value, the parameter value is empty (NULL 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 Unwired Server. Define a screen that displays the results returned from the EIS.

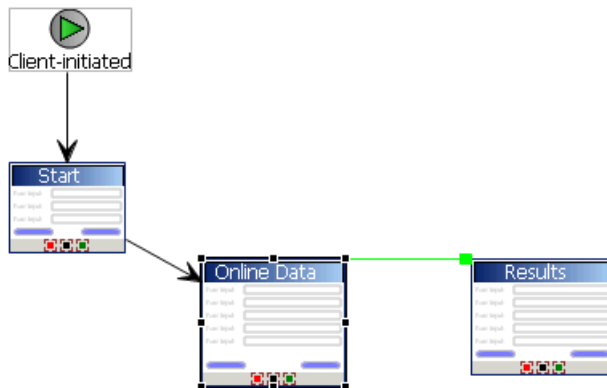
1. Define a control that passes the load argument to Unwired Server from the screen (named Online Data) that contains the menu item (named Find) that invokes the `findByParameter` object query:
 - a) Select an **EditBox** control and click in the control area.

- b) Name the EditText `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.
 - 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.

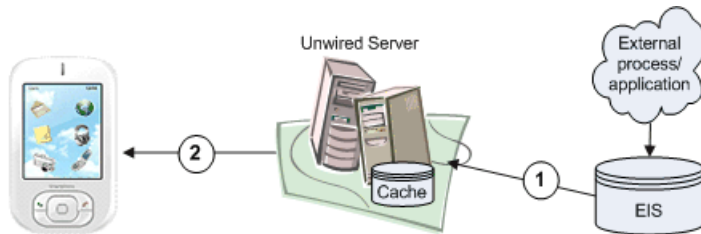


4. Configure the Results screen to display the results. In this example, the Emp MBO, contains three attributes: Id, empName, and empDeptId. Create a Listview with a cell for each attribute to display the results returned from the EIS as a list:
 - a) From the Flow Design window, double-click the **Results** 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 workflow 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 Unwired Server. Any data changes matching these rules trigger a notification from Unwired Server to the workflow 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/mobile workflow package, Unwired Server sends a notification to the client.

Implementing Server Notification for Workflow Clients

Set up Unwired Server to send notifications to workflow clients 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.

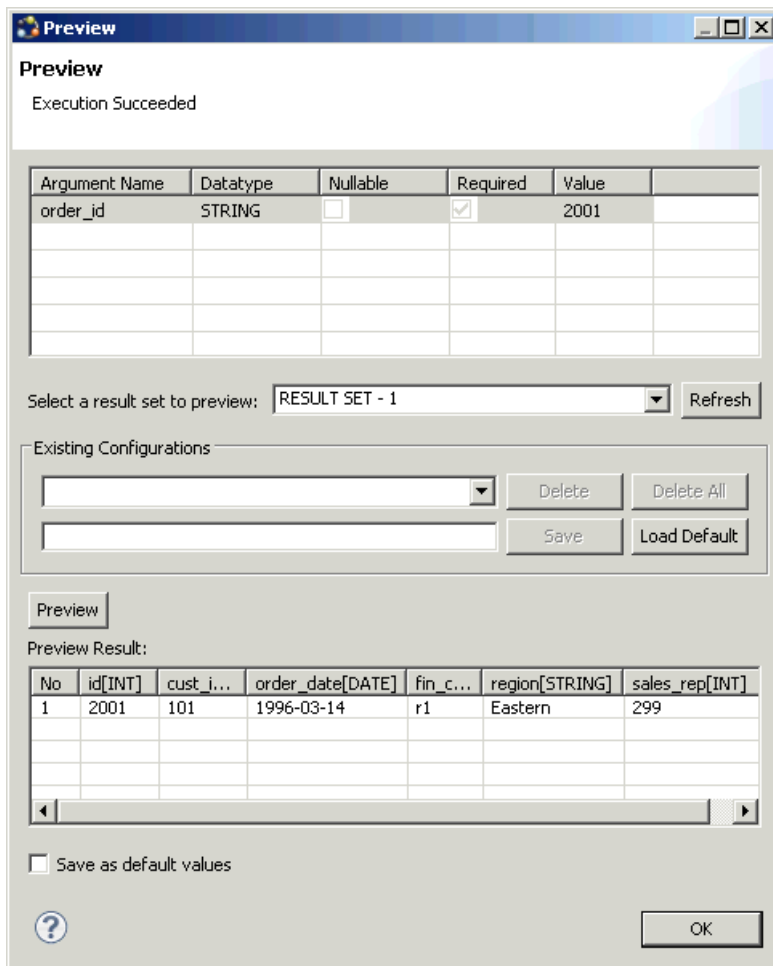
The MBO definition described here allows retrieval of online results by the workflow application to which the MBO belongs.

1. In Unwired WorkSpace, create an MBO from the sampled 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 sampled.db.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).



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 Unwired Server.

Creating the Server-Driven Notification Starting Point

Create a new workflow application with a server-initiated starting point.

1. From Unwired WorkSpace, select **File > New > Mobile Workflow Forms Editor**.
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 workflow the client receives is populated with values returned by the findByParameter object query.

New

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:

Start tag: ...

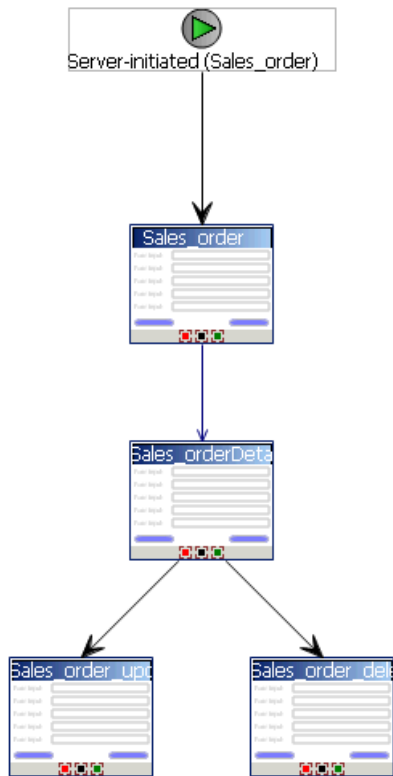
End tag: ...

Format:

Sample notification field contents:

Value extracted from sample notification field contents:

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 workflow package to Unwired Server.

Sending an Order Notification to the Device

Use the mobile workflow "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 a device user and assign it to the workflow package in Sybase Control Center (SCC).
2. Download and configure the Sybase messaging client on the device or emulator to match those performed in SCC.

See your Sybase documentation for details.

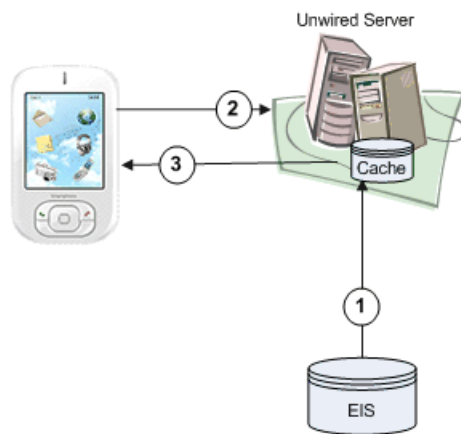
Task

1. In the Flow Design of the Mobile Workflow Forms Editor, 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 SCC and assigned to the workflow package.
3. In the Subject field, enter a sales order that meets the matching rules criteria defined for the Sales_order workflow application. 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 workflow client 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 workflow application 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 Unwired Server cache (also called the CDB) to the client.

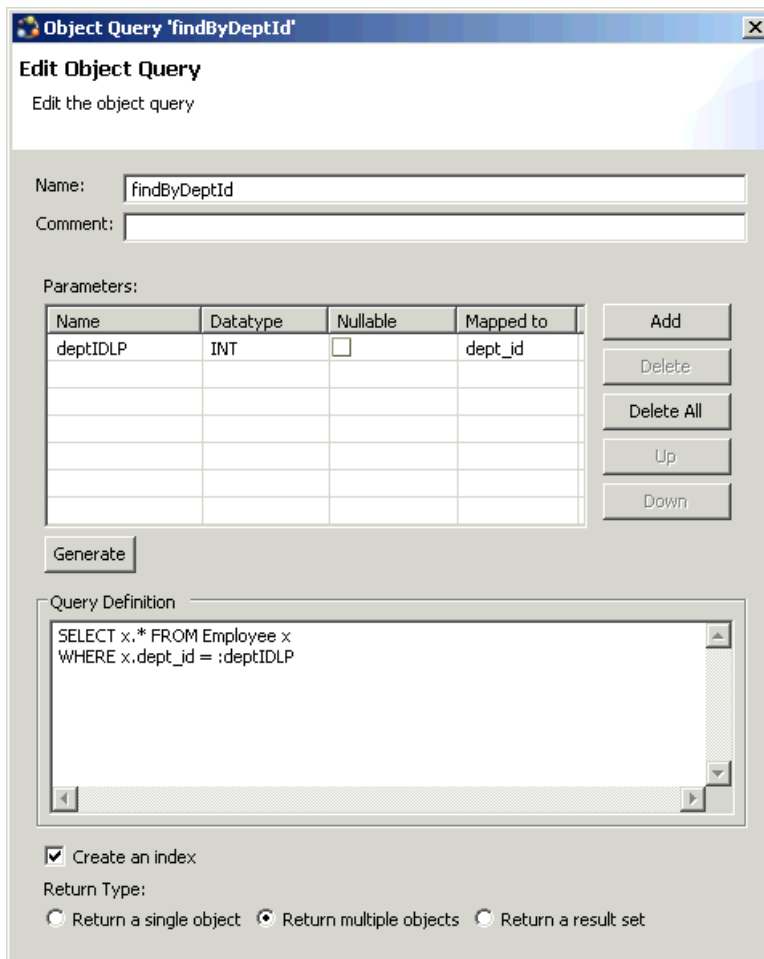
This example defines an MBO that retrieves employee benefit information for all employees of a given department based on the dept_id attribute using the findByDeptId object query.

1. From Unwired 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
```



Object Query 'findByDeptId'

Edit Object Query
Edit the object query

Name:

Comment:

Parameters:

Name	Datatype	Nullable	Mapped to
deptIDLP	INT	<input checked="" type="checkbox"/>	dept_id

Buttons: Add, Delete, Delete All, Up, Down

Generate

Query Definition

```
SELECT x.* FROM Employee x
WHERE x.dept_id = :deptIDLP
```

☒ Create an index

Return Type:

☐ Return a single object ☒ Return multiple objects ☐ Return a result set

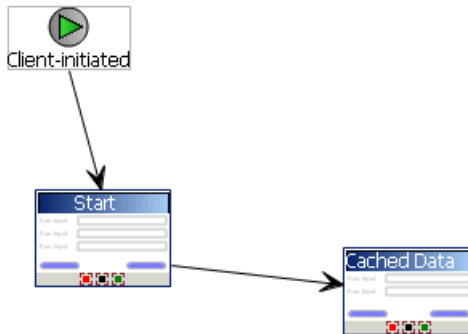
4. Deploy the project that contains the MBO to Unwired Server. Select **Message-based** in the deployment wizard.

Binding the findByDeptId Object Query to a Menu Action

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

1. From Unwired Workspace, launch the Mobile Workflow Forms Editor.
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.



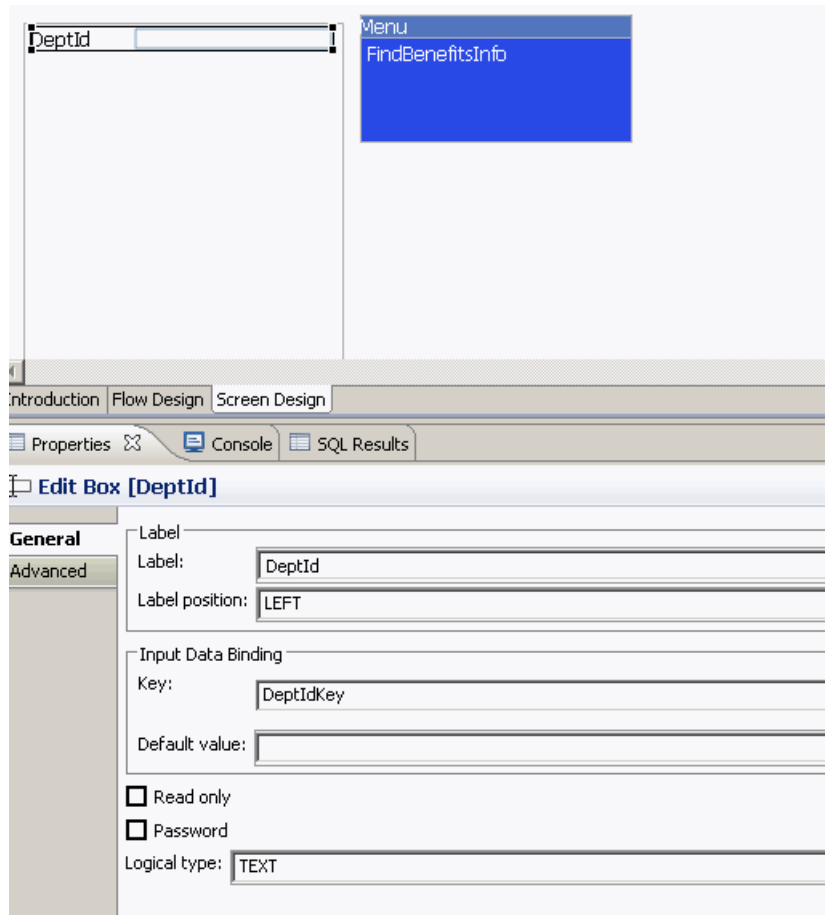
3. Highlight the menu item you want to map, or create a new menu item.
4. Define a Submit action named FindBenefitsInfo that invokes the findByDeptId object query:
 - a) In the Properties view, in the General properties for the selected menu item, select **Online Request** as the Type.
 - b) In the Details 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

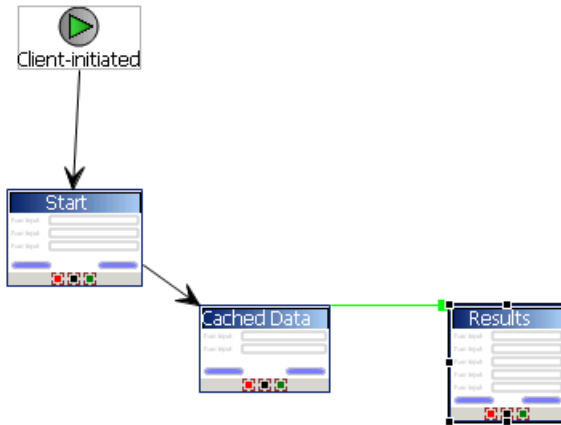
Add a control to pass the object query parameter to Unwired Server. Define a screen that displays the results returned from the Unwired Server cache.

1. Define a control that passes the object query parameter to Unwired 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**.

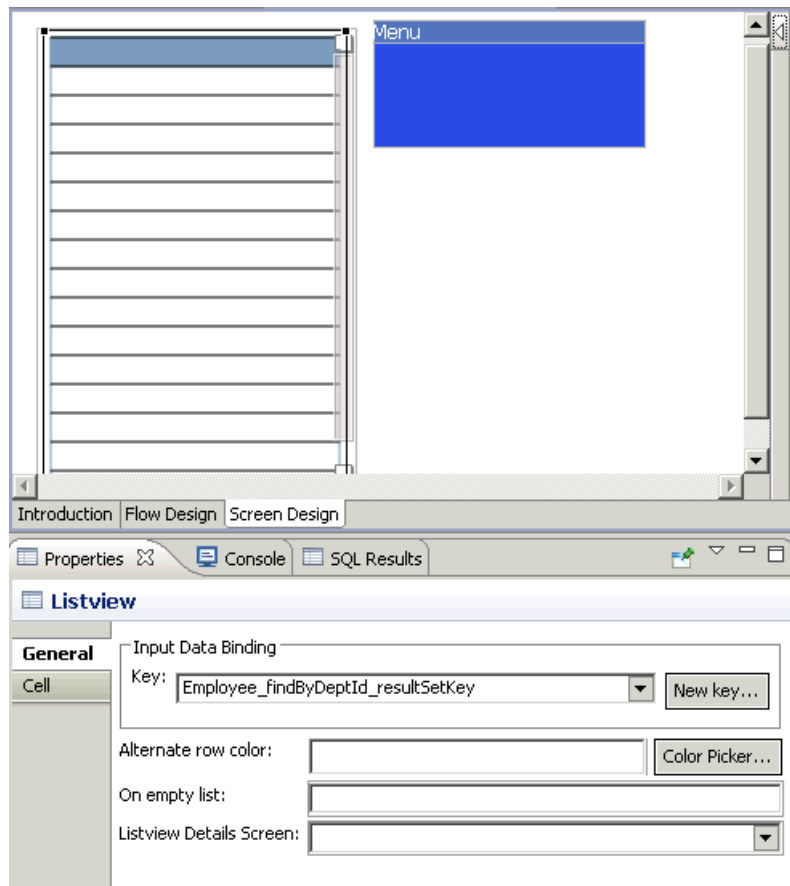


2. Select the FindBenefitsInfo menu item, and from the Parameter Mappings tab, map parameters to input keys defined for the controls. For example, map the deptIDLp parameter to the DeptIdKey key.
3. Define a screen that displays the results of the findByDeptId object query:
 - a) From the Flow Design 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 Unwired Server cache to the Results screen. The Flow Design window indicates the connection between the screens.



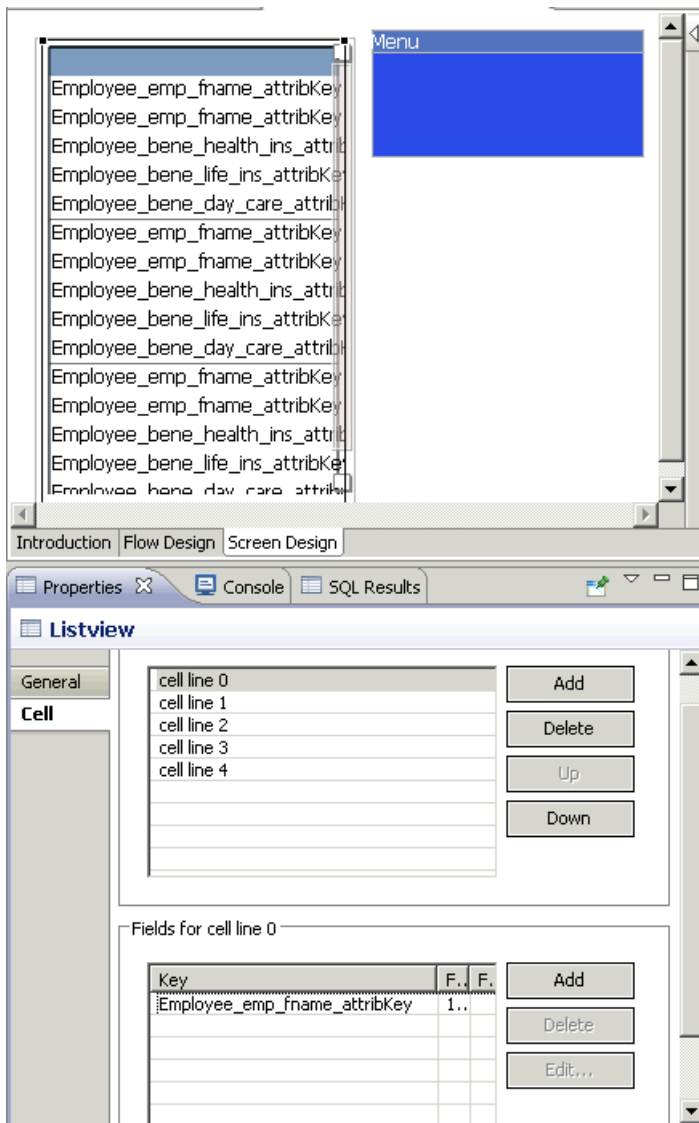
4. Configure the Results screen to display the results. In this example, the Employee MBO, contains seven attributes that identify the employee and their benefits. Create a Listview with a cell for each attribute to display the results returned from the cache as a list:
 - a) From the Flow Design window, 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).



- 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 workflow package 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 Mobile Workflow 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 Mobile Workflow Forms Editor from Unwired WorkSpace and create a new Mobile Workflow form:
 - a) Select **File > New > Mobile Workflow Forms Editor**.
 - 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 Mobile Workflow Forms Editor, map the personalization keys to the Mobile Workflow 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 Unwired Server cache (CDB) and returned to the client.

Mobile Workflow Application Configuration for Data Change Notification

This section contains details about developing workflow applications that take advantage of DCN updates.

Mobile workflow applications require a server-initiated starting point and defined matching rules, which allows Unwired Server to push changes to workflow application clients. See the topics *Starting Points* and *Adding Matching Rules*.

Extending Data Change Notification to Mobile Workflow Clients

Mobile Workflow data change notification (WF-DCN) requests allow Unwired 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 workflow application receives notification, it can retrieve data directly from the EIS or from the Unwired Server cache, keeping the application synchronized. DCN messages targeted for MBOs used in workflow 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 workflow 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 workflow server-initiated starting point, 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: workflow engine convert MBO data and WF-DCN message into workflow email, and push it to device mobile inbox

1. Operation name(op) :**upsert** or **delete**— same as regular DCN.
2. Message ID (id) of the Mobile Workflow – used for correlation (a **:delete** for a previously submitted request with **:upsert** is possible)
3. Username (to) – the Sybase Unwired 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 Sybase Control Center.

Note: The "To" field must match the Unwired Platform user name—not the user name used to register the device.

4. Subject (subject) – subject of the workflow message.
5. Originator <from> – who the workflow message is from.
6. Body of the workflow message <body> – it can embed customized information.

7. <received> – received time of the Mobile Workflow message.
8. <read> – whether the Mobile Workflow message is read.
9. <priority> – whether the Mobile Workflow 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>"
    }
  ]
}
```

Mobile Workflow 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, Unwired Server gets the **wf** cmd value from the request first, and:

1. Unwired Server invokes preProcessFilter if the DCN filter is specified.
2. Unwired 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 Mobile Workflow 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 workflow package in the “to” attribute of the Mobile Workflow message, then matches them with the receiver list.

For every receiver, Unwired Server generates multiple workflow messages (all workflow 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. Unwired Server invokes the `postProcessFilter`, if specified.
8. If no errors occur, Unwired 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, Unwired Server returns an HTTP 500 error with the body of the JSON log records.

Non HTTP Authentication Workflow DCN Request

You can send Mobile Workflow 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 Unwired Server Administrator, and *supPwd* represents the Administrator's password defined during Unwired Platform installation.

Sending Workflow DCN to Users Regardless of Individual Security Configurations

You can send Mobile Workflow DCN requests to users in other security configurations if you belong to the default security configuration.

If the workflow 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>
```

Mobile Workflow DCN Request Response

After processing of the Mobile Workflow DCN request, Unwired 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 Mobile Workflow 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"
    }
  ]
}
```

Workflow 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.

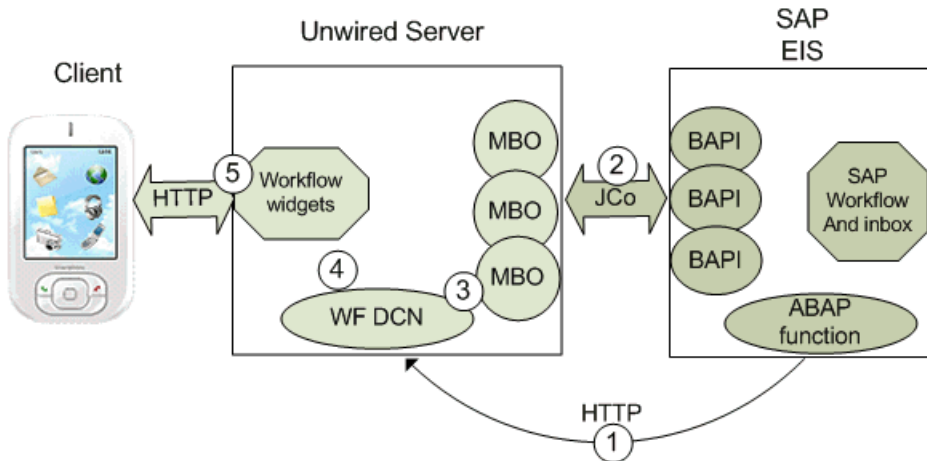
Comparing Workflow DCN With and Without Payload

This section compares the two types of WF-DCN and includes examples of each.

Mobile Workflow DCN Without Payload

Understand how to construct a workflow DCN without payload message.

This example illustrates data flow of a WF-DCN without payload using an SAP EIS:



1. The WF-DCN pushes new messages (workitems) to Unwired Server, which are then delivered to the device, for example, a workflow notification.
2. After the EIS sends a workitem id to Unwired Server, Unwired Server uses workitem MBO and workitem id to retrieve details of the workitem from the EIS.
3. After Unwired Server receives the message, a matching workflow 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 workflow 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 workflow 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:

```
{ "id": "", "op": "", "subject": "test", "to": "test", "from": "test", "read": , "priority": , "body": " MATCH: SUP_MWF, TaskID: TS97200149, WIID: 1470577, USER: PERF0111*#END*", "data": {} }
```

Unwired Server extracts information from the DCN message and retrieves details from the EIS.

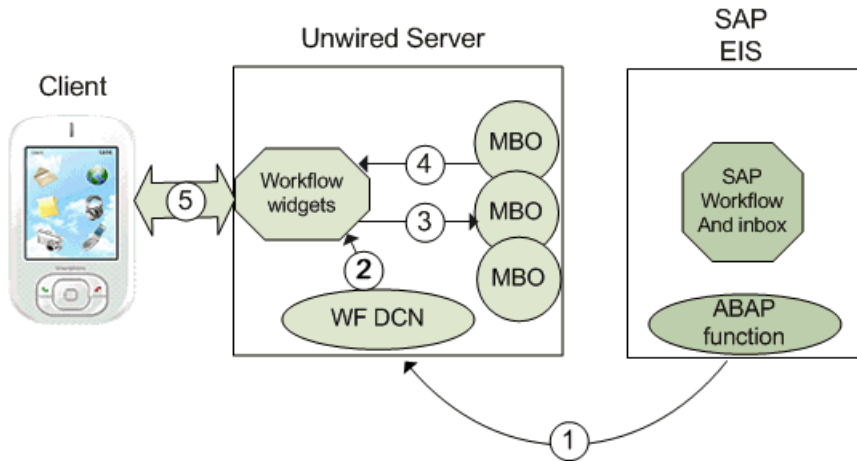
Processing the WF-DCN without payload message

After Unwired Server receives the message, a matching workflow 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.

Mobile Workflow DCN With Payload

Understand how to construct a workflow 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 Unwired 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 Unwired Server, the workflow engine evaluates the matching rule against all registered workflows. If a matching rule matches this message, the workflow server starting point for that workflow is triggered to process the message.
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 workflow 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 workflow engine converts the MBO data and WF-DCN message into a workflow notification, then pushes it to the device mobile inbox using Sybase 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":
{"attributel":"value1","attribute2":"value2","attribute3":"value3"}
}]
}]

```


The message must contain e-mail information: subject, to, from, and so on, and include the package name, 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": "dcbtest", "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": "A"
            }
          }
        ]
      }
    ]
  ]
}
```

Sample Java Function for Generating Workflow 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();

        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);
        }
    }
}

```

Mobile Workflow Development

Mobile Workflows support the occasionally connected user and addresses the replication and synchronization issues those users present for the back-end system.

A Mobile Workflow application requires an integration module on the server side, which is implemented by a static set of logic that processes Mobile Workflow-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 generate Mobile Workflow forms that work on these platforms:

- Apple iOS
- BlackBerry
- Windows Mobile Professional
- Android

See *Supported Hardware and Software* for supported version levels.

Develop a Mobile Workflow Using the Mobile Workflow Forms Editor

The Mobile Workflow Forms editor provides UI controls that make development of Hybrid Web Containers fast and easy.

The Mobile Workflow Forms Editor

The Mobile Workflow Forms Editor allows you to create the forms that can be assigned to the devices that have the Hybrid Web Container installed and that are registered in Sybase Control Center.

The Mobile Workflow Forms Editor contains these pages:

- **Introduction**

This optional page is the starting point for designing your mobile workflow form, and for accessing help.

By default, the Introduction page opens when you create a new mobile workflow or double-click an existing mobile workflow. To hide this page, click **Window > Preferences**, navigate to **Sybase, Inc > Mobile Development > Mobile Workflow Forms Editor**, and unselect the option, **Show the Introduction page when opening the designer**.

- **Flow Design**

The Flow Design page displays an empty canvas if you did not choose any of starting points in the New Mobile Workflow Forms Editor. If you selected a starting point when creating the new mobile workflow form, the selected starting point appears.

Use this page to specify how starting points and screens link together. The Palette view on the right shows available starting points, screens, and connections. You can drag and drop items from the Palette to the canvas. Screens and starting points that you drag to the Flow Design canvas appear as icons. Double-click a screen to open the Screen Design page.

Right-click anywhere in the Flow Design page to open a context menu.

- **Screen Design**

Use this page to design the screen of your mobile workflow application. The Palette view shows the menu items and controls you can use to design a new screen. Drag controls to design the user interface.

Setting Mobile Workflow Forms Editor Preferences

Use the Preferences dialog to set the appearance and device preferences for the Mobile Workflow Forms Editor.

1. Select **Window > Preferences**.
2. In the Preferences dialog, select **Sybase, Inc. > Mobile Development > Mobile Workflow Forms Editor**.
3. Select **Flow Design Page** to set the preferences for the Flow Design page.
4. Select **Screen Design Page** to set the preferences for the Screen Design page.
5. Click **Apply**.
6. Click **OK**.

Mobile Workflow Forms Editor Preferences

Use the Preferences dialog to set the preferences for the Flow Design and Screen Design pages.

Preferences determine the default behavior and style for the Flow Design and Screen Design pages and their objects. These settings affect all instances; to change the style and behavior of an individual object instance, modify its properties in the Properties view.

You can set the following preferences in the Mobile Workflow Forms Editor page and the Screen Design Page:

- **Optimize for performance** provides a more basic look and feel. Raw HTML controls are used to the extent possible (for example, there is no signature or slider control in HTML), with little to no CSS stylization or JavaScript enhancements to customize the look and feel. This option is always used for Windows Mobile Professional platforms.
- **Optimize for appearance** provides a more sophisticated, or native, look and feel, but may affect performance. This option is not available on Windows Mobile platforms.

- **Show the Introduction page when opening the designer** controls whether the Introduction page is visible or hidden. If this option is enabled, the Forms Editor opens to the Introduction page, which provides basic information and help. .

Screen Design Page

This page and its subpages control global settings for the Flow Design and Screen Design pages of the Mobile Workflow Forms Editor. The page includes these options:

- Show connector handles
- Show pop-up bars
- Enable animated layout
- Enable animated zoom
- Enable anti-aliasing
- Show status line

Appearance

Modify the colors and fonts for various Mobile Workflow objects by clicking the color controls, or by selecting **Change**, and setting the desired color and font.

Connections

When a new connection is created, it follows the default style defined in the preferences settings. You can change the style of the connection line individually in the Flow Design Properties view.

Select the default line style for new connections:

- Oblique (default) – the connection lines are not aligned with the horizontal or vertical axis and you can create breakpoints anywhere on the connection line. If you drag the connection line, a new breakpoint is created at the drag point and two lines are created.
- Rectilinear – the connection line always remains either horizontal or vertical.

Printing

Set the general printing preferences:

Section	Description
Orientation	Either portrait or landscape.
Units	Either inches or millimeters.
Size	Choose one of the available print size. Optionally, override the default width and height values.
Margins	Enter a value, in inches, for the margins.

Rulers and Grid

Set the preferences for the ruler and grids:

Section	Description
Rulers options	To show ruling lines, select Show rulers for new diagram , then select ruler units.
Grid options	<p>To modify grid settings, select any of these options:</p> <p>Show grid for new diagrams</p> <p>Snap to grid for new diagrams</p> <p>Snap to shapes for new diagrams</p> <p>Then, enter the grid spacing.</p>

Starting Points

Starting points specify which screen the user sees when they perform a certain action.

Starting points include:

- Client-initiated – this starting point is activated when the user opens the mobile workflow on the device.
- Activate – this is the starting point that is activated when the user first opens the mobile workflow. It does not activate on the second and subsequent times the application is started. Multiple workflow applications can share the same Activate starting point. Different workflows can specify an activation key, and workflows with the same activation key share their activation status. For example, if Workflow A and Workflow B both specify an activation key of AB (via the key attribute on the RequiresActivation tag), when Workflow A gets activated, it also activates Workflow B such that when Workflow B is invoked for the very first time, its activation screen will not be displayed.

Note: A mobile workflow application with an Activate starting point is considered to be successfully activated only if it is closed with a Submit Workflow menu item.

- Credential request – this starting point is activated when the mobile workflow application requires a user name and password and the program does not have one in its cache, or the values in the cache are no longer valid.
- Server-initiated – this starting point is activated when a notification message is sent to the device that matches the matching rules specified in the mobile workflow application. The developer can add conditions, based on the notification message, to determine which application screen opens first.

Each starting point has a list of keys with which it is associated. View the keys associated with the starting point in the Properties view for the starting point.

Adding a Starting Point Manually

Every screen must be part of a flow that connects to a starting point.

Note: If you manually create a server-initiated starting point that invokes an object query, the list key will not have any children. To create the children for the list key, you can use the New

Mobile Workflow Forms creation wizard to create the server-initiated starting point, or you can drag and drop the mobile business object onto the Flow Design page. Either of these two methods will create the associated children key for a server-initiated starting point that invokes an object query. Otherwise, you will have to manually set up the list key's children.

1. From the palette on the Flow Design page, select a starting point, then click on the Flow Design canvas.
2. Depending on the type of starting point you select:
 - If you are adding an Activate, Credential Request, or Client-initiated starting points, go to the next step.
 - If you are adding a Server-initiated starting point, the Notification Processing wizard opens. In the wizard, complete the steps to configure the object query and notification message to be used, and click **Finish** to create the starting point.
3. Add a screen to the starting point by either:
 - Dragging and dropping a data source, for example, a mobile business object or mobile business object operation, onto the Flow Design canvas, or
 - Selecting an empty screen from the Flow Design palette, then clicking on the Flow Design page.
4. Add a connection between the starting point and the screen.
5. Select **File > Save**.

You can also automatically create starting points based on selections you make in the New Mobile Workflow Form wizard when you create a new Mobile Workflow form.

Creating a Server-initiated Starting Point

Create a server-initiated starting point.

Note: Each Mobile Workflow Form package can have only one server-initiated starting point.

1. In the Mobile Development perspective, select **File > New > Mobile Workflow Forms Editor**.
2. Follow the instructions in the New Mobile Workflow Forms Editor wizard:

Field	Description
Enter or select the parent folder	Select the mobile application project in which to create the mobile workflow form.
File name	Enter a name for the mobile workflow form. The extension for mobile workflow forms is .xbw.
Advanced	Link the mobile workflow form to an existing file in the file system.

Field	Description
Link to file in the file system	<p>Click Browse to locate the file to which to link the mobile workflow form. 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.</p> <p>Click Variables to define a new path variable. Path variables specify locations on the file system.</p>

3. Click **Next**.
4. In the Starting Points page, select **Responds to server-driven notifications** and click **Next**.
5. In the Select a Mobile Business Object and Object Query page, click **Search**.
6. In the Search for Mobile Business Object page: click **Search**.
 - a) Click **Search**.
 - b) Accept the current project or choose a different one.
 - c) Click **Search**.
 - d) Select a Mobile Application Project from the list and click **OK**.
7. (Optional) In the Select a Mobile Business Object and Object Query page, select an object query and click **Next**.

Note: Object queries that return result sets are not supported.

8. In the Specify Sample Notification page, enter contents in the fields that you want to appear in the sample notification message, then click **Next**.
For example enter `Order (2001) created` as the Subject.
9. In the Specify Matching Rules page:
 - a) Select the text that will serve as the matching rule.
 - b) Right-click and select **Select as Matching Rule** to create the matching rule.
 - c) Click **Next**.
10. In the Identify Parameter Rules page, select an extraction rule and specify its properties.

Note: The supported syntax for the regular expression is that used by the C# Match() methods of the class System.Text.RegularExpressions.Regex.

Property	Description
Field	Select the field from the notification message from which the parameter receives data, for example, Subject. Selecting Subject indicates that the parameter receives data from the subject line of the notification message.
Start tag	Enter or select the text for the start tag regular expression, for example, "Approval Request\" (" .
End tag	Enter or select the text for the end tag regular expression, for example, "\") was sent " .
Format	Specify a format that supports locale-specific parameter extraction. For example, you can indicate that a given date parameter will be in the form of yyyy-MM-dd in the e-mail, and another date parameter will be in the form of yyyy-dd-MM. This applies to the DateTime type.
Sample notification field contents	Enter the value of the field previously specified on the Specify Sample Notification page of the Notification Processing wizard.
Value extracted from sample notification field contents	Specify the value the start tag and end tag matching rules, given the sample field value.

11. Click *Finish* to create default screens and starting points.

When the notification is sent to the client, the sample value is replaced with the specified key, and the client's form is populated with values returned by the object query

Server-initiated Starting Point Properties

View and configure the server-initiated starting point properties.

General

Select the **Server-initiated** starting point in the Flow Design page to view and configure its properties.

Property	Description
Mobile business object	The mobile business object with which the server-initiated starting point is associated.

Property	Description
Object query	<p>When you define an object query in the mobile business object associated with the Mobile Workflow form, the object query returns the specified instances of the mobile business object. For example, if a travel request to Hawaii is submitted, the object query returns an instance of that mobile business object so the approver can review the travel request details then modify the status and comment fields for that mobile business object.</p> <p>Object queries that return multiple instances are supported, as well as object queries that return only one instance.</p>
Generate old value keys	<p>By default, this is selected, which means when data is sent, there are two copies sent—one with the key names you select at design time, and one with the same names prefixed with "_old." The keys you selected at design time may get updated by the application, but the old values will not unless this option is selected. Those "_old" value keys may be critical when executing an update or parent update operation at a later time. If you unselect this option, old value keys are not generated.</p>
Jumpstart	<p>Launch the Notification Processing wizard, where you can compose a sample notification message from which to specify parameter value extraction rules and matching rules.</p>
Extraction rules	<p>Shows the notification extraction rules for the server-initiated starting point. Select a rule and click Edit to change the transformation rule.</p>
Error screen	<p>The screen to which the user navigates in case an asynchronous operation invocation fails.</p>
Subject	<p>If an error occurs, a notification is sent to the device with the appropriate error log keys and specified Subject. The value you specify here appears in the Subject field of the notification.</p>

Property	Description
From	When an error occurs and the notification appears in the inbox, what you specify here will appear in the From field of the notification.

Keys

Keys associated with the controls on the screen are shown.

Property	Description
Key name	The name of the associated key.
Type	The type of the associated key, for example, string.
Data binding	Shows the data source to which the key is bound. Click New to create a new key. Select any key and click Edit to edit the information for an existing key. Select a key and click Remove to remove an existing key.

Parameter Mapping

Use the Parameter Mapping section to bind the parameters (if any) of an operation or object query to keys or, context data, such as DeviceName, BackEndPassword, BackEndUser, and so on.

Personalization Key Mappings

If you have a mobile business object with load parameters mapped to transient personalization keys, you can specify values for those personalization keys when you invoke an operation or object query from the Mobile Workflow package. This can, for example, be useful for loading only specific data into the consolidated database from large data sources.

Start Screen(s)

This page contains 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.

Property	Description
Default screen	Select the default start screen. This is the start screen that is used if all of the conditions are false or absent.

Property	Description
Condition name	Click Add to add a condition. If this condition is determined to be true, the screen that matches the condition is used as the start screen.
Conditional success screen name	Select the screen that will be associated with the condition.

Notification Extraction Rules

Notification extractions rules are used to extract parameter values to keys, so you can then map the parameters to objects, such as keys.

This means that you can extract values from the notification without needing to use them right away as parameter values. You can instead bind them to controls and display them to the user, or use them later on in the workflow to submit a create/read/update/delete (CRUD) operation or object query.

Notification extraction rules include a regular expression that determines from where in the notification message the parameter value starts, and another regular expression that determines from where in the notification message the parameter value ends. Define the notification extraction rules when you create a server initiated starting point.

Some e-mail clients allow you to specify the To: and From: fields without quotes (for example, John Doe <john.doe@sybase.com>), the actual value received by the Unwired server may have quotes in it (for example, "John Doe" <john.doe@sybase.com>), which will need to be accounted for in any extraction rules that act on those fields.

Editing Notification Extraction Rules

Edit notification extraction rules in the server-initiated starting point General properties.

1. On the Flow Design page, select the Server-initiated starting point.
2. In the General Properties view, select the notification extraction rule to edit and click **Edit**.
3. In the Edit Notification Extraction Rule dialog:

Property	Description
Field type	Select the field in the notification from which the parameter value is extracted, for example, Subject, which indicates the parameter value is extracted from the subject line of the notification message.
Tag before parameter	Enter the tag to indicate what to extract from the beginning of the field's contents, for example, "Approval Request\" (".

Property	Description
Tag after parameter	Enter the tag that indicates what to extract from the end of the field's contents, for example, " <code>\)was sent</code> ".
Format	Format uses the C# ParseExact syntax, and applies to DateTime only. It supports locale-specific parameter extraction. For example, you can indicate that a given date parameter will be in the form of yyyy-MM-dd in the e-mail, and another date parameter will be in the form of yyyy-dd-MM. This applies to the DateTime type.
Key	The key that the value from the notification is extracted to.

Note: The supported syntax for the regular expression is that used by the C# Match() methods(s) of the class System.Text.RegularExpressions.Regex.

4. Click **OK**.

Editing the Server-initiated Starting Point

Edit the Server-initiated starting point.

1. In the Flow Design page, right-click the **Server-initiated starting point** to edit and select **Edit Server-initiated starting point**.
2. Make your changes in the Notification Processing wizard and click **Finish**.
3. From the main menu, select **File > Save**.

Adding Matching Rules

Matching rules determine how notification messages are redirected at runtime.

To see the Matching Rules section of the Properties view, verify there are no objects selected on the Flow Design page.

1. In the Properties view, click **Matching Rules**.
2. Click **Add**.
3. In the matching rules dialog, specify the type and regular expression for the matching rule, and click **OK**.

Creating Keys

Create new keys to bind to controls.

1. On the Screen Design page, select the control to which you want to bind the new key.

2. In General Properties, in the Input Data Binding section, click **New key**.
3. In the Key dialog, enter:

Property	Description
Name	Enter a name for the key. The key name must be unique across the scope of the workflow application.
Type	Select the key type.
Sent by server	Select this checkbox to input data binding for the key.
Input Data Binding	
MBO attribute	Select to bind the key to a mobile business object (MBO) attribute: <ul style="list-style-type: none"> Name Convert to UTC
MBO relationship	Bind the key to an MBO relationship.
MBO object query results	Bind the key to MBO object query results.
Hard coded value	When the message is sent by the server to the client, rather than getting the value of the key from a source like an MBO attribute, it uses the fixed value you input. <div> Note: The return of hard coded key values is supported from server-initiated starting points, only, not online requests. </div>
User defined	Bind the key to user-defined data.
Extraction rule	This is determined by the extraction rules you set up.

4. Click **OK**.

Keys

Keys are unique identifiers within the scope of a workflow application, used when setting or retrieving a value.

Keys can be bound to the mobile business object (MBO), whether it is to an MBO attribute, an MBO parameter, an MBO operation parameter, a subscription parameter, an object query, or an MBO relationship (list type keys only).

Keys can be of the following types:

- string

- int
- double
- decimal – NaN (not a number) values are not supported for keys of type decimal
- bool
- DateTime
- list – can be bound to relationships, or nothing. Keys of other types cannot be bound to relationships. List type keys can also be bound to MBO object query results, which returns multiple objects. List type keys have one or more child keys to which they are bound.

Binding Controls to Keys

Use the Properties view to bind keys to controls.

1. On the Screen Design page, select the control to which you want to bind the key.
2. In General Properties, in the Input Data Binding section, select the key with which to bind the control from the drop-down list.
3. (Optional) In Default Value, enter a default value for the key.
4. Select **File > Save**.

Specifying Values for Transient Personalization Keys

If you have a mobile business object with load arguments mapped to transient personalization keys, you can specify values for those transient personalization keys when you invoke an operation or object query from the Mobile Workflow package.

This is useful for loading only specific data into the consolidated database (CDB) from large data sources.

1. Create a mobile business object that has load arguments. (Set the load arguments in the MBO Properties view, from the Load Arguments tab.)
2. Create a corresponding transient personalization key for each load argument.
3. Map each load argument to the corresponding transient personalization key.
4. In the Mobile Workflow Forms Editor, map the personalization keys to the Mobile Workflow keys for the menu item (in the same way you map parameters to Mobile Workflow keys).

You can also fill the personalization key values from values extracted from the e-mail message, depending on from where you are invoking the object query.

Configuring Parameter Mappings

Configure mapping between parameters and keys or context data.

Parameter mappings for the selected operation or object query are shown in the Properties view.

1. In the Properties view for the menu item, select the **Parameters Mapping** tab.

2. Click **Add** to map a key or context data, such as BackEndUser, BackEndPassword, DeviceName, UserName, MessageID, and so on to a parameter.
3. Select a key, and click **Edit** to edit mapping between a parameter and a key or context data.
4. Select a key and click **Delete** to delete existing mapping between a key or context data and a parameter.

Generated Keys

Listview controls are bound to a list key, however, the listview may only display a subset of that list key's child keys.

For example, when you have created a listview screen by dragging and dropping a mobile business object (MBO), you get a list key that has child keys bound to attributes and child keys bound to the create, update and/or delete operations' parameters. The listview, however, displays only the keys bound to attributes.

If you create a row using parameters bound to attribute keys, you get a new row with non-empty values visible in the listview. If you create a row using parameters bound to parameter keys, you get a new row with empty values visible in the listview, the non-empty values are in keys not displayed in the listview. For some MBO operations it would not be advisable to bind parameters to attribute keys (for example, an MBO with first_name and last_name attributes and a create operation that had one full_name parameter); but for many MBO operations, specifically those automatically generated for MBOs with database backends, it is appropriate.

Flow Design Page Properties

These are the Flow Design page properties shown in the Properties view when no objects in the Flow Design page are selected.

General

Property	Description
Module name	The name of the module.
Module version	The version of the module.
Module description	Enter a description for the module.
Module display name	Enter the name that appears on the workflow application when it is viewed on the client.

Property	Description
Delete message after processing	If this option is selected, server-initiated workflow messages are removed from the device Inbox after they have been processed. A message is considered to have been processed if the corresponding workflow application is closed as a result of an asynchronous submit menu item being activated.
Mark message as processed after processing	If this option is selected, server-initiated workflow messages are marked as processed in the device Inbox (indicated by a green check mark over the original icon) after they have been processed. A message is considered to have been processed if the corresponding workflow application is closed as a result of an asynchronous submit menu item being activated.
Client icon	Select the icon that appears on the client.
Credentials cache key	Specify a key to use for the credential starting point. If you do not enter anything, each workflow will have a different credentials key. If there is more than one workflow with the same credentials key, then all of those workflows can share the same credentials so that authentication for each workflow does not have to be separately defined.
Requires activation key	Specify a key to use for activation. If you do not enter anything, each workflow will have a different activation key, which means that each workflow will have to be activated separately. If you specify an activation key, then when you activate one workflow, all the other workflows sharing the specified activation key are automatically activated.
Shared storage key	Specify a storage key to share storage between Mobile Workflow packages. This storage key is passed to the manifest.xml file used by the container.

Rulers and grid

Property	Description
Display	<ul style="list-style-type: none"> • Show ruler • Show grid • Grid in front
Measurement	<ul style="list-style-type: none"> • Ruler units • Grid spacing
Grid line	<ul style="list-style-type: none"> • Color • Style
Snap to grid	Automatically align objects to the grid lines on the canvas.
Snap to shapes	Position objects to grid lines that go through the vertical and horizontal edges of other objects.

Localization

Property	Description
New	Launch the New Locale wizard and create a new locale file for the mobile workflow application.
Update	Update the currently loaded locale file.
Load	Load a locale properties file to the Mobile Workflow Designer.
Validate	Select a locale properties file and click to validate.
Edit	Select the locale properties file, then click to edit the locale properties file.
Remove	Select the locale properties file to remove, then click to remove.

Matching rules

Any existing matching rules appear in the matching rules Properties view. Matching rules determine how notifications are redirected at runtime.

Property	Description
New	Create a new matching rule.
Delete	Select a matching rule and click to delete.
Edit	Select a matching rule and click to edit.

Authentication

Property	Description
Use static credentials	Select when the authentication field is the same for all users accessing the resource.
Use SUP server connection profile authentication	Use properties from the connection profile to supply credential data for authentication. This is similar to hard-coded credentials but the credentials are taken from the specified server connection profile instead of being specified in the Mobile Workflow Forms Editor.
Use hard-coded credentials	Specify a user name/password combination for authentication.
User name	This field is editable only if you selected the hard-coded credentials option. Enter the user name to use for authentication.
Password	This field is editable only if you selected the hard-coded credentials option. Enter the password to use for authentication.
Use certificate-based credentials	<p>Click Generate from Certificate to select a certificate file from which to generate authentication. Once you choose the certificate to use, these fields are filled in automatically.</p> <ul style="list-style-type: none"> • 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 to – the date the certificate is valid to

Property	Description
Unwired Server domain	<p>Optionally specify the Sybase Unwired Platform domain to use. If specified, a context variable is created in the application for which the initial value is set to this domain. At runtime, when determining authentication, <i>@<domain></i> is added to the user name, thus allowing the user access to that domain.</p> <hr/> <p>Note: When you create a new security configuration that includes the SAPSSOTokenLoginModule, and deploy it to a new domain, if the mobile workflow application uses the MBOs associated with the new security configuration, you must specify an Unwired Server domain that corresponds to the domain using the security configuration. See the Sybase Unwired Platform <i>Security</i> guide for more information about security configurations.</p>

Screen Design Page Properties

These are the Screen Design page properties shown in the Properties view when no objects in the Screen Design page are selected.

General

These are the general properties for the screen.

Property	Description
Name	The name of the screen.
Key	The key associated with the screen.
Background color	<p>Click Color picker to choose a background color for the screen.</p> <p>You can also enter the value manually. The possible values are black, blue, brown, gray, green, orange, pink, purple, red, white, yellow, any string with the format #rrggbb, where rr, gg and bb are the hexadecimal values, in the range 0 to ff(255), for the red, green and blue.</p>

Property	Description
Background image	Click Browse to choose an image to set as the background image for the screen. Note: The image must be located in the project.
Background image repeat	Select from these options: <ul style="list-style-type: none"> • Repeat – the background image will repeat on all screens in the mobile workflow. • No repeat – the image does not repeat on each screen • Inherit – child screen inherit the background image from the parent screen • Repeat-X – the background image repeats horizontally only • Repeat-Y – the background image repeats vertically only
OK action	Select the action to perform when the user clicks OK on the screen.

Rulers and grid

The ruler and grid properties are applied to the Screen Design page, or canvas.

Property	Description
Display	<ul style="list-style-type: none"> • Show ruler • Show grid • Grid in front
Measurement	<ul style="list-style-type: none"> • Ruler units • Grid spacing
Grid line	<ul style="list-style-type: none"> • Color • Style
Snap to grid	Automatically align objects to the grid lines on the canvas.
Snap to shapes	Position objects to grid lines that go through the vertical and horizontal edges of other objects.

Keys

Keys associated with the controls on the screen are shown.

Property	Description
Key name	The name of the associated key.
Type	The type of the associated key, for example, string.
Data binding	Shows the data source to which the key is bound. Click New to create a new key. Select any key and click Edit to edit the information for an existing key. Select a key and click Remove to remove an existing key.

Mobile Workflow Controls

Controls are objects that users can interact with to view, enter, or manipulate data.

Supported controls are determined by the selected device. The set of supported controls depends on the target device platform.

Control	Supported Platform
Checkbox	<ul style="list-style-type: none"> • Apple iOS • BlackBerry • Android • Windows Mobile Professional
Editbox	<ul style="list-style-type: none"> • Apple iOS • BlackBerry • Android • Windows Mobile Professional
Choice	<ul style="list-style-type: none"> • Apple iOS • BlackBerry • Android • Windows Mobile Professional

Control	Supported Platform
Slider	<ul style="list-style-type: none"> • Apple iOS • BlackBerry • Android <p>Note: When the Optimize for performance option is set in preferences, the Slider control is supported only on BlackBerry.</p>
Signature	<ul style="list-style-type: none"> • Apple iOS • BlackBerry 6.0 • Android <p>Note: When the Optimize for Performance option is set in preferences, the Signature control is not supported.</p>
HtmlView	<ul style="list-style-type: none"> • Apple iOS • BlackBerry • Android • Windows Mobile Professional
Listview	<ul style="list-style-type: none"> • Apple iOS • BlackBerry • Android • Windows Mobile Professional
AttachmentViewer	<ul style="list-style-type: none"> • Apple iOS • BlackBerry • Android • Windows Mobile Professional

Control	Supported Platform
Button	<ul style="list-style-type: none"> • Apple iOS • BlackBerry • Android • Windows Mobile Professional <p>Note: The button control works on all platforms without an image and with only label settings; or with an image without label settings.</p> <p>If the button has both label and image settings, it shows only the image on iOS and Windows Mobile (no text), and it shows both the image and text on BlackBerry 6 and Android.</p>
Label	<ul style="list-style-type: none"> • Apple iOS • BlackBerry • Android • Windows Mobile Professional
Link	<ul style="list-style-type: none"> • Apple iOS • BlackBerry • Android • Windows Mobile Professional
Image	<ul style="list-style-type: none"> • Apple iOS • BlackBerry • Android • Windows Mobile Professional <p>Note: Dynamic images are not supported on Windows Mobile.</p>

See the topic *Configuring Control Properties* in *Sybase Unwired WorkSpace – Mobile Workflow Package Development* online help for information about how to configure controls.

Menu Items

Menu items function as actions and are associated with mobile business object operations.

Each action in the Mobile Workflow application results in a native menu item being generated when you generate the files for the Mobile Workflow package. Native menu support is provided on BlackBerry, Android, and Windows Mobile platforms. On the iOS platform, menus are provided from HTML controls designed to emulate the iPhone navigation bar and toolbar.

- Save screen – closes the current screen, validates the current input and, if validation succeeds, saves it.
- Cancel screen – discards any changes on the current screen and closes it. No validation occurs.
- Close workflow – discards any changes on any open screens and closes all open screens. No validation occurs.
- Open screen – opens a different screen.
- Online request – creates a synchronous call to the server, validates the data input on all open screens, saves it, sends it to the server and waits for a response from the server before continuing.
- Submit workflow – creates an asynchronous call to the server, validates the data input on all open screens, closes the application, and sends the data to the server. You cannot use a Submit workflow action to invoke an operation on a nested, or child, element.
- Select certificate – opens a dialog where the user can choose a certificate to use for the credentials.
- Create key collection – opens the specified screen in the context of adding a new key collection to the list key bound to the list view.
- Add/Update/Delete Key Collection – adds, updates, or deletes a key from a key collection from a list key bound to a Listview control. These operations are not invoked immediately, but are instead invoked when a Submit operation is made on the parent MBO to which this MBO is related. A key collection can be added only to screens that are navigated to as the result of a Listview details navigation, for example, by clicking a row in a Listview.
- Custom – write your own custom code that dictates its behavior.

Configuring a Menu Item or Custom Action

Configure a menu item or custom action item in the Mobile Workflow application.

Prerequisites

From the Screen Design Palette, add the item that you want to customize. See Adding Palette Objects to the Screen Design.

Task

Whether you define a menu item or a custom action, the action results in a method being generated when you generate the files for the Mobile Workflow package. If the action is associated with a menu item, a native menu is generated for the supported BlackBerry, Android, and Windows Mobile platforms. On the iOS platform, menus are provided from HTML controls that are designed to emulate the iPhone navigation bar and toolbar.

If you create a menu item or action that invokes an object query, it automatically creates an output key, which is a list and which has keys that are bound to the attributes of the mobile business object.

Menu Item and Custom Action Properties

The Properties view shows properties for a selected item in the menu custom action box on the Screen Design page.

General

Property	Description
Name	Enter a valid name for the menu or custom action. Note: To avoid wrapping of menu item names on Apple iOS devices, keep the name length to 10 or fewer characters.
Key	Select the key with which to bind the item.
Default	For a menu item, select to display the menu item prominently on the screen.

Property	Description
Type	<ul style="list-style-type: none"> • Save screen – closes the current screen, validates the current input and, if validation succeeds, saves it. • Cancel screen – discards any changes on the current screen and closes it. No validation occurs. • Close workflow – discards any changes on any open screens and closes all open screens. No validation occurs. • Open screen – opens a different screen. • Online request – creates a synchronous call to the server, validates the data input on all open screens, saves it, sends it to the server and waits for a response from the server before continuing. • Submit workflow – creates an asynchronous call to the server, validates the data input on all open screens, closes the application, and sends the data to the server. You cannot use a Submit workflow action to invoke an operation on a nested, or child, element. • Select certificate – opens a dialog where the user can choose a certificate to use for the credentials. • Create key collection – opens the specified screen in the context of adding a new key collection to the list key bound to the list view. • Add/Update/Delete Key Collection – adds, updates, or deletes a key from a key collection from a list key bound to a Listview control. These operations are not invoked immediately, but are instead invoked when a Submit operation is made on the parent MBO to which this MBO is related. A key collection can be added only to screens that are navigated to as the result of a Listview details navigation, for example, by clicking a row in a Listview. • Custom – write your own custom code that dictates its behavior.

Property	Description
Details	You can configure actions in the Details section. The properties available depend on the action type. See the topics for configuring the different menu or custom action types for information about configuring these properties.

Parameter Mapping

Use the Parameter Mapping section to bind the parameters (if any) of an operation or object query to keys or, context data, such as DeviceName, BackEndPassword, BackEndUser, and so on.

Personalization Key Mappings

If you have a mobile business object with load parameters mapped to transient personalization keys, you can specify values for those personalization keys when you invoke an operation or object query from the Mobile Workflow package. This can, for example, be useful for loading only specific data into the consolidated database from large data sources.

Output Keys

Lists the keys of which values will be filled by the server and returned to the client. Output keys are used only with an Online Request menu or custom action type.

Creating a Save, Open, Close, or Cancel Menu Item

Create a menu item that performs a save, open, close, or cancel action on the device application screens.

Note: The first screen in each mobile workflow form should have a Close or Cancel menu action assigned to it so that the user can exit from the mobile workflow on the device. On a Windows Mobile device, the user can exit by clicking the **OK** button, but not all devices have that function.

1. On the Screen Design page, from Palette, click **Menuitem**, then click the Menu box.
2. Select the menu item to configure the properties in the Properties view.
3. In the Properties view:

Property	Description
Name	Enter a valid name for the menu item. Note: To avoid wrapping of menu item names on Apple iOS devices, keep the name length to 10 or fewer characters.
Key	Select the key with which to bind the menu item.

Property	Description
Default	Select to display the menu item prominently on the screen.
Type	<p>Select the type of action the menu item will perform when selected:</p> <ul style="list-style-type: none"> • Save – allows the user to close the current open screen and save the input made to the collection of response values to be sent to the server. • Close – allows the user to leave the client-side component no matter which screen the client-side component currently shows. • Cancel – allows the user to close the current open screen without performing any data validation. The cancel action type immediately navigates the user back to the underlying screen. • Open – indicates the screen to be opened with a GoTo connection. The Open type menu item must be associated with a screen on the Flow Design.
Screen	This field appears in the Details section only if the menu item type is Open. Select the screen that will open from the GoTo connection.

4. Select **File** > **Save**.

Creating a Submit Workflow Menu Item

A Submit Workflow menu item allows the user to update, delete, or create data even when not connected to the network. When the user connects, data is automatically sent and received.

The submit workflow menuitem of the first screen in mobile workflow forms editor must be associated with a mobile business object (MBO) operation or an invoke parent update.

If the user has multiple screens open at the time of the submit execution, and one of the underlying screens fails validation, the user is prompted with the configured warning message and the submit process stops.

1. From the palette on the Screen Design page, click **MenuItem**, then click the Menu box.
2. Select the new menu item.
3. In the Properties view of the menu item, configure the properties for the submit menu item:

Property	Description
Name	Enter a valid name for the menu item. Note: To avoid wrapping of menu item names on Apple iOS devices, keep the name length to 10 or fewer characters.
Key	Select the key with which to bind the submit menu item.
Default	Select to display the menu item prominently on the screen.
Type	Select Submit Workflow .

4. If the menu item is an operation associated with a mobile business object, you can configure these properties in Details:

Property	Description
Mobile business object	The MBO with which the menu item is associated. Click Search to locate the MBO.
Invoke parent update	No changes are made to the parent MBO, but the appropriate operations (create, update, delete) are performed on its descendants.
Invoke operation	Invoke the selected MBO operation with which the menu item is associated. Select the operation from the list, which is populated when you select the MBO.
Submit confirmation message	(Optional) Enter text to show in a message box on the screen when the submit operation finishes successfully.
Resubmit confirmation message	If the option Mark message as processed after processing is enabled in the Flow Design page properties, show the resubmit confirmation message if the user attempts to submit the workflow after it has already been previously submitted.
Error screen	The screen to which the user navigates in case an asynchronous operation invocation fails. Select an existing screen, or click Generate Error Screen to create a new one. If no error screen is specified, the client application receives no notification of the failure — failures appear only in the log files.

Property	Description
Subject	<p>If an error occurs, a notification is sent to the device with the appropriate error log keys and specified Subject.</p> <hr/> <p>Note: The subject can reference the screen name and action name using macros (except in server-initiated workflows). For example:</p> <p>"An error occurred trying to perform the <code>\${action}</code> action on the <code>\${screen}</code> screen."</p> <hr/>
From	<p>When an error occurs and the notification appears in the inbox, the comment appears in the From field of the notification.</p> <hr/> <p>Note: The subject can reference the screen name and action name using macros (except in server-initiated workflows). See the Subject example.</p> <hr/>

5. Select **File > Save**.

Creating an Online Request Menu Item

Create an online request menu item for synchronous calls to Unwired Server, which means the device client waits for a response from the server.

1. From the palette on the Screen Design page, click **Menuitem**, then click the Menu box.
2. In the Properties view, select the menu item for which to configure the properties.
3. In the Properties view, configure the properties for the online request menu item:

Property	Description
Name	<p>Enter a valid name for the menu item.</p> <hr/> <p>Note: To avoid wrapping of menu item names on Apple iOS devices, keep the name length to 10 or fewer characters.</p> <hr/>
Key	Select the key with which to bind the online request menu item.
Default	Select to display the menu item prominently on the screen.
Type	Select Online Request .

4. If the menu item is an operation associated with a mobile business object (MBO), you can configure these properties in the Details section:

Property	Description
Mobile business object	The MBO with which the menu item is associated. Click Search to locate the MBO.
Invoke parent update	No changes are made to the parent MBO, but the appropriate operations (create, update, delete) are performed on its descendants.
Invoke operation	Invoke the selected MBO operation with which the menu item is associated. Select the operation from the list, which is populated when you select the MBO.
Invoke object query	Invoke the object query from the menu item.
Generate old value keys	By default, this is selected, which means old value keys are generated. If you need the old value keys for update or parent update operations, you should leave this selected. If you unselect it, old value keys will not be generated.
Show credential screen on authentication failure	If selected, the credential screen is shown when authentication fails. If unselected, the authentication failure is treated the same as other types of failures.
Submit error message	Enter the message the user sees if there is an error during the submit operation. When an operation is invoked, the user can receive any error messages that may occur on the client application. The error messages are a single key of the List type, with a single key of the String type as its child. Each child instance corresponds to an error message.
Timeout	Enter the allowed amount of time, in seconds, to wait for the server to respond before timing out and giving an error message.
On device cache timeout	Represents the time, in seconds, that query results are cached on the device. If the same query is made with the same parameter values within the specified timeout, the results are pulled from the device cache and no call to the server is made.

Property	Description
Error screen	<p>The screen to go to if there is an error when submitting the data to the server.</p> <p>Click Generate Error Screen to automatically generate error screens.</p>
Success screen	<p>Choose the Default Success Screen from the drop-down. This is the screen to go to upon successful communication with the server.</p> <p>Click Add to add a conditional navigation screen. A conditional navigation adds a callout during Mobile Workflow package generation to the customConditionalNavigation function, which returns a boolean. Each piece of custom logic is evaluated, in order, one at a time, and when the first condition comes back as true, the user is navigated to the matching screen for that condition.</p>

5. (Optional) Click **Parameter Mappings** to define the keys to map to parameters.
You see the parameters defined for the MBO and used to generate the object query. In addition to Key, you can map parameters to BackEndPassword, BackEndUser, DeviceId, DeviceName, DeviceType, UserName, MessageId, ModuleName, ModuleVersion, and QueueId.
6. (Optional) Click **Personalization Key Mappings** to select a personalization key and the key to which it maps.
You can also fill the personalization key values from values extracted from the notification, depending on from where you are invoking the object query.
7. (Optional) Click **Output Keys** to view the keys that will be created and populated based on the selected action.
8. Select **File > Save**.

Creating a Key Collection Menu Item

Create and configure a Key Collection menu item.

Key collection menu items can be used only on screens with a Listview control.

1. In the Screen Design page, from the palette, click **MenuItem**, then click the Menu box.
2. Click the menu item to select it and configure the properties in the Properties view.
3. In the Properties view:

Property	Description
Name	Enter a valid name for the menu item. Note: To avoid wrapping of menu item names on Apple iOS devices, keep the name length to 10 or fewer characters.
Key	Select the key with which to bind the key collection menu item.
Default	Select to display the menu item prominently on the screen.
Type	Select one of these types: <ul style="list-style-type: none"> • Add Key Collection • Update Key Collection • Delete Key Collection

4. If the menu item is an operation associated with a mobile business object, configure these properties in Details.

Property	Description
Mobile business object	The mobile business object with which the menu item is associated. Click Search to locate the mobile business object with which to associate the menu item.
Invoke parent update	No changes are made to the parent MBO, but the appropriate operations (create, update, delete) are performed on its descendants.
Invoke operation	The mobile business object operation with which the menu item is associated. Select the operation from the list, which is populated when you select the mobile business object.
Invoke object query	Invoke the object query from the menu item.
Generate old value keys	By default, this is selected, so that old value keys are generated. If you need the old value keys for update or parent update operations, leave this selected. If you unselect it, old value keys are not generated, which results in a much smaller message, which improves performance.
ListView	Select the ListView from the drop-down list.

5. Select **File > Save**.

Defining a Custom Menu Item or Custom Action

Associate a custom action with a menu item, custom action item, or control, and write your own code to dictate its behavior.

Prerequisites

From the Screen Design Palette, add the item that you want to customize.

Task

1. In the Screen Design page of the Mobile Workflow Forms Editor, select the menu item, control, or custom action that you want to configure.
2. In the Properties view Type field, select **Custom**.
3. Edit the `Custom.js` file, adding custom code in the `customBeforeMenuItemClick` to handle the action.
In your code, call the custom function that is autogenerated in `workflow.js`
4. Save the `Custom.js` file.

Connections

Use connections to connect screens with actions in the Flow Design.

Connection actions describe the navigation direction the screen should go after the user performs an action, cancels an action, or uses a control. Alert actions use actions defined in the Flow Design.

Note: You can use only one connection of each type from a given source screen, except for the GoTo and Operation Success connection types. You can have as many Operation Success connections on a source screen as you have synchronous operations on the source screen. You can have as many GoTo connection types as you want to on a source screen.

Screens and Navigations

Navigations represent transitions from a starting point to a screen, or from one screen to another.

New screens are opened when the user clicks a menu item or a row in a Listview control. Screens can also be opened upon success or failure of an operation.

Screens series are designed from a starting point. Screens appear one at a time on the device. Screens are opened on a stack, and closed in the reverse order in which they were added. Each screen contains one or more controls.

Adding and Configuring the AttachmentViewer

The AttachmentViewer control allows you to download and view attachments.

1. In the Mobile Workflow Forms Editor, select **Screen Design > Palette > Controls**.

2. Click **AttachmentViewer** and then click in the screen.

3. From the Properties view, configure the Attachment Viewer:

- a) Select the **AttachmentViewer** control.
- b) Click the **Properties** tab.

If you do not see the Properties tab, right-click anywhere in the Screen Design page, and select **Show Properties View**.

- c) Click **General** in the left pane to configure the general properties for the AttachmentViewer.

You can bind the AttachmentViewer control to data in one of two ways:

- Keys, or
- Object queries

Note: MBO attributes displayed as attachments in a mobile workflow application must be binary type, no matter what the corresponding backend data type is.

4. (Optional) Click **Parameter Mappings** to bind the parameters (if any) of an operation or object query to keys or context data, such as DeviceName, BackEndPassword, BackEndUser, and so on

5. (Optional) Click **Personalization Key Mappings** if you have a mobile business object with load parameters mapped to transient personalization keys, and want to specify values for those personalization keys when you invoke an operation or object query from the Mobile Workflow package.

This can be useful for loading only specific data into the consolidated database from large data sources.

6. Select **File > Save**.

Supported Attachment Types

This shows which device platforms support which attachment file type.

Device Platform	File Types
iPhone	*.jpg, *.jpeg, *.png, *.tif, *.tiff, *.gif, *.doc, *.docx, *.xls, *.xlsx, *.ppt, *.pptx, *.pdf, *.htm, *.html, *.txt
BlackBerry	*.gif, *.jpg, *.png, *.bmp, *.txt, *.html Install an attachment viewer like Documents To Go Premium Edition to view additional file types, such as *.pdf, *.doc, *.xls, *.xlsx, *.ppt, and *.pptx.

Device Platform	File Types
Windows Mobile	*.doc, *.xls, *.xlsx, *.ppt, *.pptx, *.gif, *.jpg, *.png, *.bmp, *.txt, *.html, *.mp3 PDF support requires the user to install a PDF viewer on the device.
Android	*.gif, *.jpg, *.png, *.bmp, *.txt, *.html Other file types, like Microsoft and Adobe, can be supported by installing additional applications on the device.

AttachmentViewer and Image Limitations

There are some limitations on the size of the attachments and images that you can include as part of the Mobile Workflow message.

These limitations vary by platform.

Platform	Size Limit
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, the maximum size is 2MB, which means that the maximum size of attachment included as part of a Mobile Workflow message must be less than that. Keep in mind that you have to include the rest of the Mobile Workflow message, in addition to the attachment contents, and also include allowances for an increase in the size of the attachment when it is Base64-encoded to allow it to be included in the XML message.
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.
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.

Note:

When accessing very large binary (image) data in the mobile business object associated with the mobile workflow, ensure that the attribute set in the mobile business object is a **BigBinary** datatype, rather than Binary.

AttachmentViewer Properties

Select AttachmentViewer on the Screen Design page to view and configure its properties.

General

Property	Description
Label	<ul style="list-style-type: none">Label – enter the text that describes the control on the user interface.Label position – select the position of the label in relation to the control it describes.
Input data binding	<p>Select the key with which to bind the AttachmentViewer control, or click New key to create a new key. The key represents a binary attribute, which will be the contents of the attachment.</p> <ul style="list-style-type: none">Key – the unique identifier within the scope of the workflow form, used when setting or retrieving a value.Default value – default value for the control.
Read-only	Select to make the control read-only.

Property	Description
Content	<p>The other way you can bind the Attachment-Viewer to data is by using an object query and mapping the parameters of the object query to keys. Indicate which binary attribute of the object query corresponds to the contents of the attachment.</p> <ul style="list-style-type: none"> • Use object query – select to invoke a specified object query. <ul style="list-style-type: none"> • Mobile business object – select the mobile business object that contains the object query to invoke. • Object query – specify the object query to invoke. • Attribute –specify the binary attribute value that corresponds to the attachment contents. <hr/> <p>Note: Ensure that the MBO binary attribute has a size specified that is big enough to support the size of the attachment when developing the MBO. To do this, you must change the MBO's binary attribute from BINARY to BINARY(%) where % is larger than the largest binary content you want to store.</p> <p>See <i>Sybase Unwired WorkSpace - Mobile Business Object Development > Develop > Working with Mobile Business Objects > Mobile Business Object Data Properties > Datatype Support</i> for information about setting the binary attribute size.</p> <hr/> <ul style="list-style-type: none"> • MIME type key – MIME type for attachments that are supported, for example, *.doc, *.xls and so on. • File name. • File size. <hr/> <p>Note: Attachments requested on demand are limited to 5MB after Base64 encoding.</p> <hr/>

Parameter Mapping

Use the Parameter Mapping section to bind the parameters (if any) of an operation or object query to keys or, context data, such as DeviceName, BackEndPassword, BackEndUser, and so on.

Personalization Key Mappings


If you have a mobile business object with load parameters mapped to transient personalization keys, you can specify values for those personalization keys when you invoke an operation or object query from the Mobile Workflow package. This can, for example, be useful for loading only specific data into the consolidated database from large data sources.

Deploy the Mobile Workflow Package to Unwired Server

Use the Mobile Workflow Package generation wizard to generate the Mobile Workflow package and deploy it to Unwired Server to make it available for device clients.

Generating the Files for a Mobile Workflow Package

Use the Mobile Workflow Package Generation wizard to generate the files for the mobile workflow package, optionally deploy the generated package files to the server, and assign the package to one or more users' devices.

1. Right-click in either the Flow Design or Screen Design page of the Mobile Workflow Forms Editor and select **Generate Mobile WorkflowPackage**, or click the code generation icon  on the toolbar.
2. In the Mobile Workflow Package Generation wizard, enter or select:

Option	Description
Favorite configurations	(Optional) Select a configuration.
Package Generation and Deployment	

Option	Description
Update generated code	<p>Generate the mobile workflow package and its files. When this option is unselected, the mobile workflow package files are not regenerated, so that modifications made to files that are normally regenerated are not overwritten. This also means, however, that changes made in the Mobile Workflow Forms Editor are not reflected in the generated files.</p> <hr/> <p>Note: The <code>manifest.xml</code> and <code>work-flow_package.zip</code> files are generated even if this is not selected.</p>
Generate into the project	Place the generated mobile workflow package and its files in the current project.
Generate to an external folder	Place the generated ZIP file containing the mobile workflow package and its generated files into a location outside of the current project. Click Browse to select the alternate location.
jQuery Mobile theme	Choose a theme for devices that use the jQuery Mobile as the UI framework. The default is theme B. See the jQuery Mobile documentation at http://jquerymobile.com/ for information about the jQuery Mobile themes.
Unwired Server profile	Select the Unwired Server profile with which to associate the mobile workflow and extract the user name and password credentials if you are using static authentication.
Deploy to an Unwired Server	Deploy the mobile workflow package to an Unwired Server.
Deploy mode	<p>The deploy mode is automatically set; you cannot change it.</p> <ul style="list-style-type: none"> • New • Replace • Update

Option	Description
Assign workflow to user(s)	<p>The mobile workflow must be assigned to a device user before the mobile workflow is visible on the user's device. You can assign the same mobile workflow to multiple users. Separate multiple users with a comma. Device users must be registered in Sybase Control Center.</p> <p>Click Get Users to select device users from the list. You must have registered device users in Sybase Control Center to populate this.</p>
Validate controls as soon as the user tries to change focus away from them	<p>If this option is unselected, validation occurs only when the screen is saved. If selected, validation occurs as soon as the control loses focus. If validation fails, a help element appears and shows the error message.</p>
Optimize JavaScript in the generated workflow package	<p>The public JavaScript files (API.js, Callbacks.js, Camera.js, and so on) contain the client API functions that you can access for use with your Mobile Workflow package customization. By default, the wizard generates a single JavaScript file (such as SUP0.js, SUP1.js, or SUP2.js), that concatenates these files. Unselect this option if you prefer to use the JavaScript files separately.</p>

3. Click **Finish**.

A ZIP file containing the application and its generated files is created and placed in the specified location.

Deployment Modes

These are the deployment modes when you generate the mobile workflow package.

New

The New deployment mode initially generates and deploys the mobile workflow package.

Replace

The Replace deployment mode removes an installed mobile workflow package and installs a new mobile workflow package with the same name and version. The Replace deployment mode acquires a list of assigned devices for the original package, uninstalls the original package, installs the new package with the same name and version, then assigns the original device list to the new package, thus preserving any device assignments associated with the original package.

Use the replace deployment mode for minor changes and updates to the mobile workflow, or during initial development.

Update

The Update deployment mode installs a new mobile workflow package with the original package name and assigns a new, higher version number than the existing installed mobile workflow package. During the update operation, a list of assigned devices is acquired from the original package, the new package is installed and assigned a new version number and then the administrator specifies device assignments for the new package from the acquired list of assigned devices. Existing notifications are preserved.

Use the Update deployment mode for major new changes to the mobile workflow.

Install and Configure the Mobile Workflow Container On the Device

To enable deploying mobile workflow packages to a device, you must download, install, and configure the Mobile Workflow container on the device.

Deploy the Mobile Workflow container to devices and register the devices with Unwired Server. You can use Afaria® to install the container on devices for enterprise deployment. For information on setting up an Afaria environment, see *System Administration > Device and Application Provisioning > Provisioning With*.

See the configuration procedure for your device type.

Preparing Android Devices for the Mobile Workflow Package

Install the Mobile Workflow container on the Android device using the Android SDK. In the Settings for your Android device, disable all keyboards except the Android keyboard.

Installing Sybase Mobile Workflow on Android Devices

Use the Android SDK Manager to install Sybase Mobile Workflow application files.

To install Sybase Mobile Workflow on your Android device:

1. Connect the device.
2. Install the Android SDK.
3. Run `platform-tools\adb` and install `SybaseDataProvider.apk` and `Workflow.apk`, which are located in `<UnwiredPlatform_InstallDir>\UnwiredPlatform\MobileSDK\HybridWeb\Android`.

Configuring the Android Emulator

Configure an Android emulator for testing a Sybase Mobile Workflow package.

Note: This tutorial was developed using one of the supported Android SDK versions. If you use a different version, the interface might differ in some details.

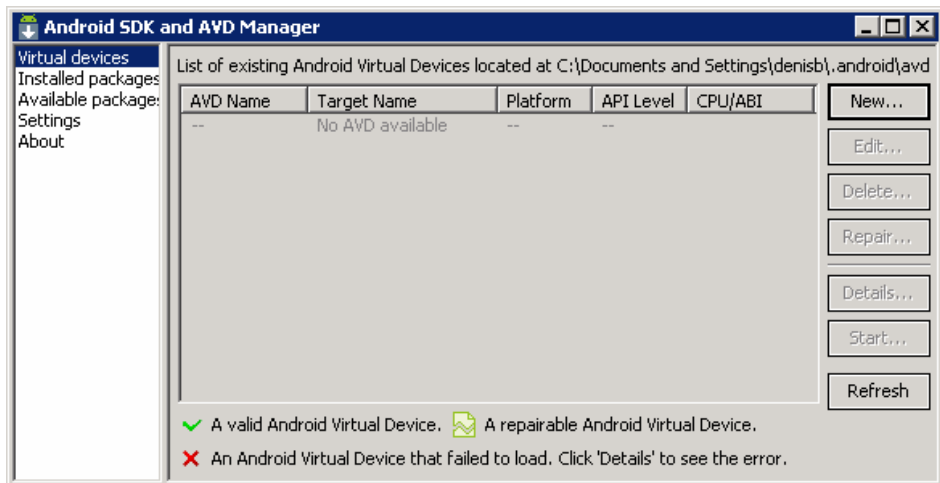
1. Install the Android SDK.

Go to <http://developer.android.com/sdk/> to download and install the Android SDK.

Follow the instructions on the Android page, with these exceptions:

- See *Supported Hardware and Software* for the most current version information for mobile device platforms and third-party development environments.
- When specifying the install location, consider choosing a path that does not contain spaces, such as C:\Android\android-sdk. Some versions of the Android SDK do not work correctly when installed in the default *drive:\ Program Files* location.
- If the Android installer stops with a message that the required Java JDK is not found on your system (even when the JDK is installed), try clicking **Back** and then **Next**, one or more times, until the installer detects the JDK.

2. Click **Start Programs > Android SDK Tools > SDK Manager**.



3. Add a device:

- In the Android SDK and AVD Manager, click **Virtual devices**, then click **New**.
- In the Create new Android Virtual Device window, enter a name.
- For the target, select a supported Android version.
- Set any other available options you want, then click **Create AVD**.

Create new Android Virtual Device (AVD)

Name:

Target:

CPU/ABI:

SD Card:

☒ Size: MiB

☐ File:

Snapshot:

☐ Enabled

Skin:

☒ Built-in:

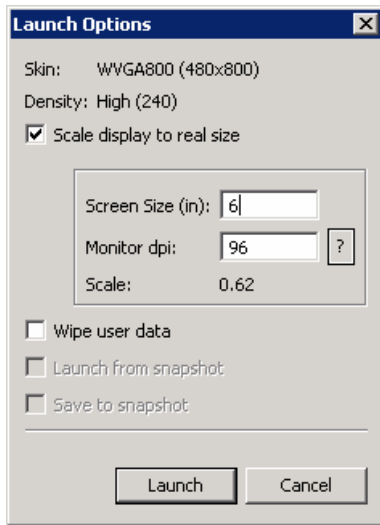
☐ Resolution: x

Hardware:

Property	Value	
Abstracted LCD density	240	
Max VM application heap size	24	

☐ Override the existing AVD with the same name

4. Select the new virtual device and click **Start**.
5. In Launch Options, optionally modify the default display scaling, then click **Launch**.



6. When the Android screen finishes loading, open a command prompt and run the **adb** command to install the Sybase application package files. SybaseDataProvider.apk and Workflow.apk to the virtual device. Install the files located in `<UnwiredPlatform_InstallDir>\UnwiredPlatform\MobileSDK\HybridWeb\Android`.

The `adb.exe` program is located in `<Android_InstallDir>\android-sdk\platform-tools`, and the apk files are located in `<UnwiredPlatform_InstallDir>\MobileSDK\HybridWeb\Android`

For example:

```
C:\Android\android-sdk\platform-tools>adb install ^
C:\Sybase\UnwiredPlatform\MobileSDK\HybridWeb\Android
\SybaseDataProvider.apk
```

```
C:\Android\android-sdk\platform-tools>adb install ^
C:\Sybase\UnwiredPlatform\MobileSDK\HybridWeb\Android
\Workflow.apk
```

The Sybase Mobile Workflow package is available in the emulator.

Preparing iOS Devices for the Mobile Workflow Package

Install the Mobile Workflow client on the device using the App Store, or use the source code provided for the Mobile Workflow container to deploy to the iOS simulator from the Xcode project.

Complete these prerequisites before provisioning the Mobile Workflow application:

- Determine your security policy – Unwired Platform provides a single administration console, Sybase 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 Sybase Control Center – application connections pair an application with a device. See *Sybase Control Center for Sybase Unwired Platform* documentation.

Apple Push Notification Service

Sybase Unwired Platform provides support for Apple Push Notification Service by pushing notifications to Mobile Workflow applications when the Mobile Workflow application is offline.

With APNS, each device establishes encrypted IP connections to the service and receives notifications about availability of new items awaiting retrieval on Unwired 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 *System Administration for Sybase Unwired Platform > System Administration > Device Provisioning > Apple Provisioning for iOS*.

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 Mobile Workflow is assigned to the device.
 - A Mobile Workflow DCN message is sent to Unwired Server, targeting a particular user and the Mobile Workflow is not running.

If you want to use APNs for the Mobile Workflow application, you can:

- Use the .p12 located in <Unwired_Platform_InstallDir>\UnwiredPlatform\Servers\MessagingServer\bin\ with the pre-built Workflow application that is available from the App Store.

These .p12 certificates are provided:

- MobileWorkflowPushDistCert.p12 – for Sybase Mobile Workflow – Free, 2.0, or 2.0.1
- MobileWorkflow21PushDistCert.p12 – for Sybase Mobile Workflow 2.1
- MobileWorkflow212PushDistCert.p12 – for Sybase Mobile Workflow 2.1.2
- Use the Apple Provisioning Portal to create your own .p12 certificate if you build your own Mobile Workflow application using the source code included in <UnwiredPlatform_InstallDir>\UnwiredPlatform\MobileSDK\HybridWeb\iOS.

After creating the .p12 certificate, you must configure the APNs settings in Sybase Control Center.

Provisioning iOS Devices

Use this procedure to provision your iOS device for APNs if you build your own Mobile Workflow application using the source code provided in

`UnwiredPlatform_InstallDir\UnwiredPlatform\MobileSDK`
`\HybridWeb\iOS\MobileWorkflow-2.1.2.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 Mobile Workflow applications to devices, you must create a certificate in your developer account and provision your device. See the *Apple Local and Push Notification Programming Guide* at <http://developer.apple.com/library/ios/#documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/ProvisioningDevelopment/ProvisioningDevelopment.html> for details.
2. Use the iPhone Provisioning Portal at <http://developer.apple.com/devcenter/ios/index.action> 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**.

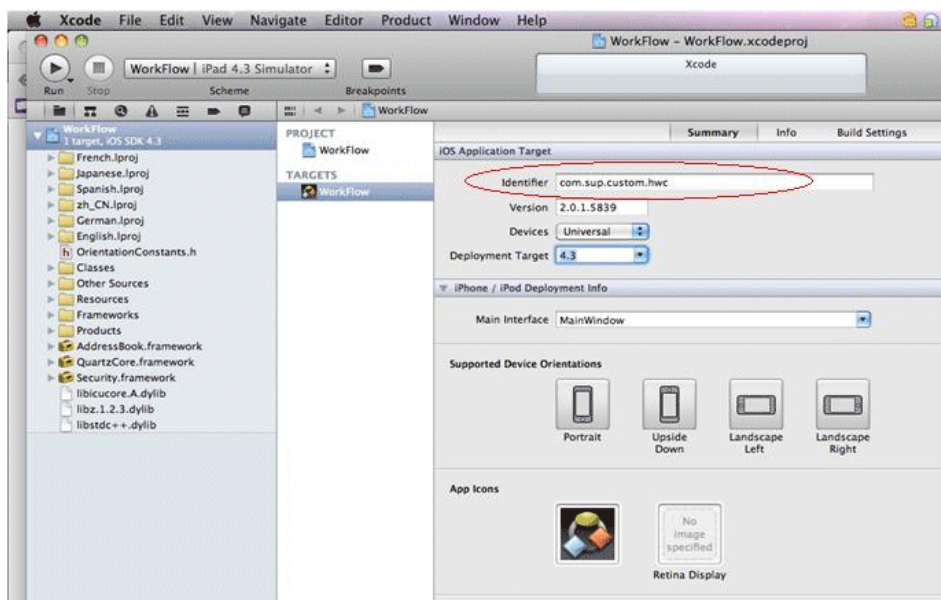
The certificate is downloaded onto your Mac and the Keychain utility appears and the certificate is imported into the "login" keychain.
7. Verify that the certificate is associated with a private key.
8. Create and install a Provisioning profile for the Mobile Workflow application.
9. In XCode, open the MobileWorkflow project.

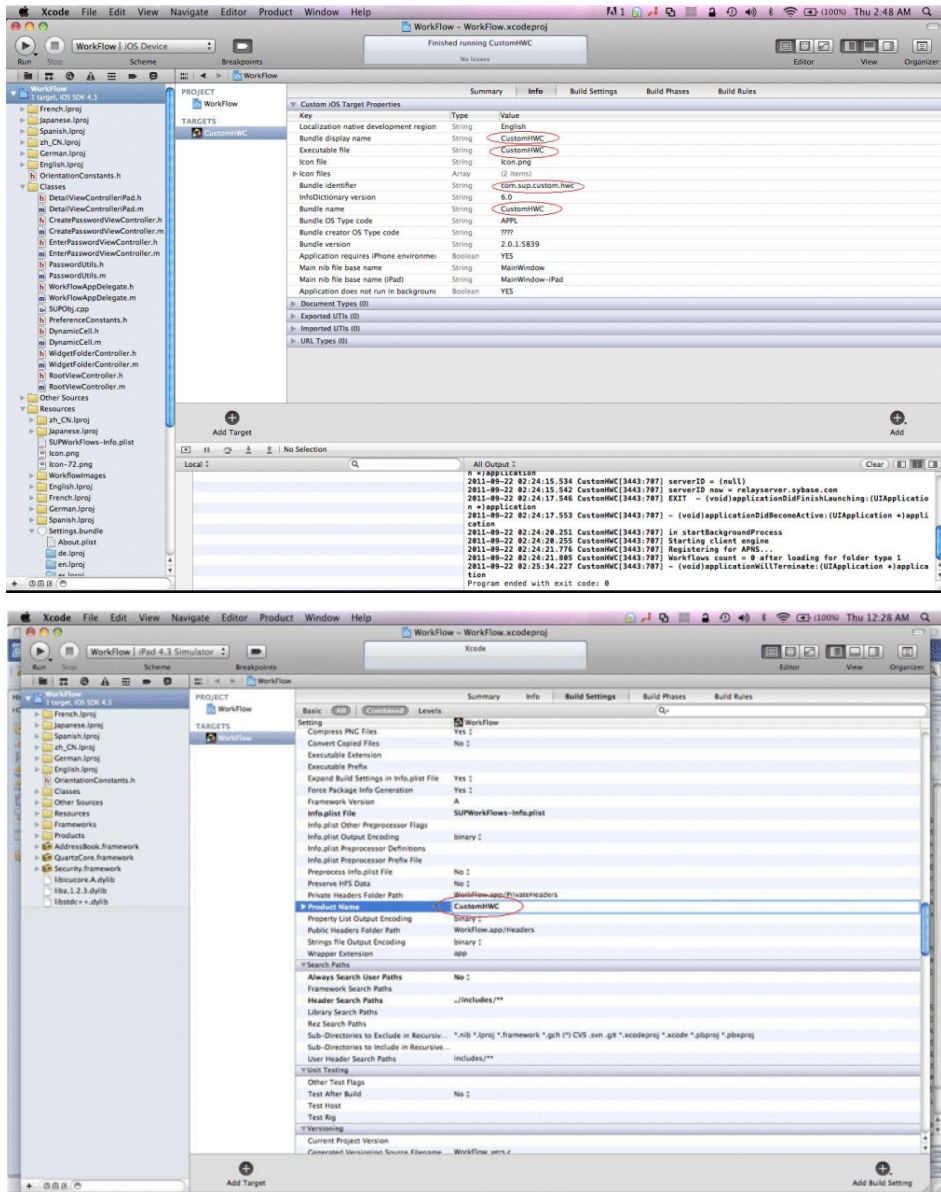
Note: Note the product name. This is used to configure the mobile workflow in Sybase Control Center and corresponds to the Application Name property in SCC. By default, the application name is HWC. This needs to be configured in the properties for the target. There is a 15-character limit for the product 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 Workflow XCode project.

Note: The Bundle Identifier must correspond to the Bundle identifier specified in the App ID. By default, the project comes with a bundle ID of `com.sybase.mobileworkflow<xxx>`. Change it to something unique.





11. Copy the exported *certificate_name*.p12 certificate to the machine where Sybase Control Center is installed and follow the instructions in Configuring Apple Push Settings for the Mobile Workflow Application 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 Mobile Workflow Application

The certificate that was exported from the keychain corresponding to Apple Push settings must be configured with the correct application name in SCC.

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. After making these changes, you must restart your machine.

1. In the left navigation pane, expand the **Servers** folder and select a server.
2. Select **Server Configuration**.
3. In the Messaging tab, select **Apple Push Configuration**.
4. Click **New**.
5. Enter the **Application name**. Make sure this name matches the AppId entered in the `Branding.strings` file.

Note: The application name is **HWC**.

6. Select **Use existing certificate** to use a security certificate file that already exists on the server.

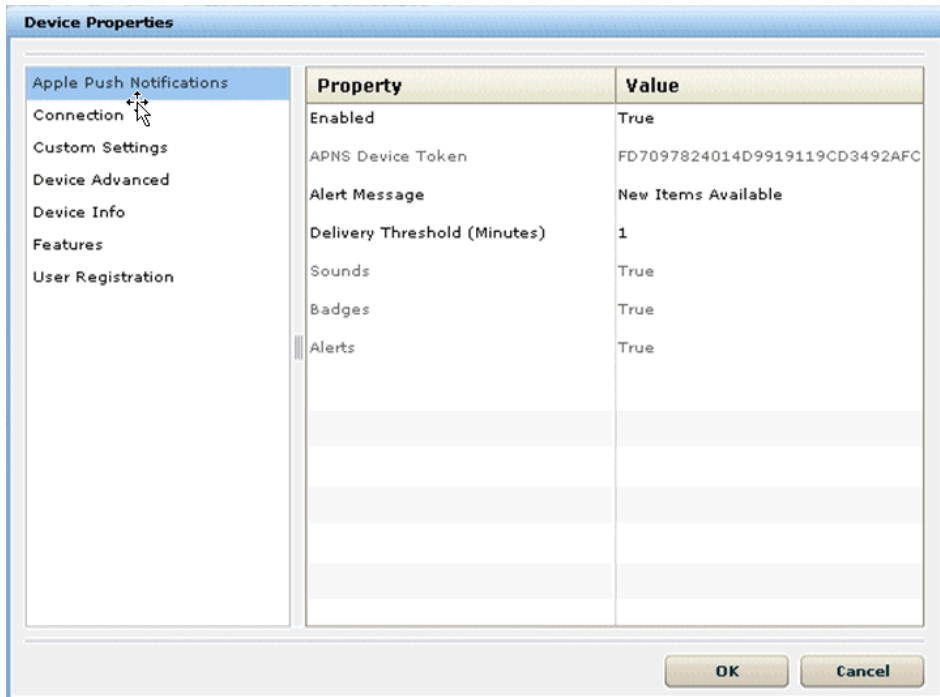
When you select this option, the list of available certificates appears in the **Certificate name** menu.

- a) Select the desired certificate from the list, for example, `MobileWorkflow212PushDistCert.p12`, which is located in `<Unwired_Platform_InstallDir>\UnwiredPlatform\Servers\MessagingServer\bin`.
- b) Enter and confirm the certificate password.

Note: If you are using the **Sybase Mobile Workflow <version>** version of the Workflow from the App Store, enter the password: `M0bileW0rkf10w;SUP`.

7. Select **Use new certificate** to create a new certificate on the server.
 - a) Enter a name for the new certificate.
 - b) Specify a Base64-encoded string by choosing one of these:
 - **Browse from file** – select a security certificate file on the server that contains the Base64-encoded string.
 - **Base64-encoded string** – manually enter the Base64-encoded string.
 - c) If you selected a file from the server for the Base64-encoded string, you can overwrite the existing certificate file with the details you specify during new certificate creation. To do so, select the box adjacent to **Overwrite existing certificate**.

- d) Enter and confirm the certificate password.
8. Click **OK**.
9. You can verify that the device is configured for APNS correctly by verifying that the device token has been passed from the workflow application after the workflow application runs once on the device.



Use the **Send a Notification** tool inside the Mobile Workflow Forms editor to send a test notification.

APNS Trace Files

<UnwiredPlatform_InstallDir>\Unwired Server\log\trace
 \APNSProvider has tracing information.

Increase the trace level to **Debug** in Sybase Control Center.

Apple Push Notification Properties

Apple push notification properties allow iOS users to install client software on their devices. This process requires you to create a different e-mail activation message using the appropriate push notification properties.

- **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

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

Installing the Mobile Workflow Application on Your iOS Device

How you install the Mobile Workflow application 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 Mobile Workflow application. The possible methods include:

- Downloading and installing the free version of the Mobile Workflow application 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 Sybase Unwired Platform installation.

Building the Mobile Workflow Container Using the Provided Source Code

The mobile workflow container referenced in this procedure is a sample container. You can use the provided source code in Xcode to build your own customized user interface and configure other resources.

Prerequisites

- Register the device in Sybase Control Center.
- You must have a Mac with the iOS SDK installed.
- Install the Xcode.

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 Sybase Unwired 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 `MobileWorkflow<version>.tar.gz` archive from `<UnwiredPlatform_InstallDir>\UnwiredPlatform\MobileSDK\HybridWeb\iOS\` to a location on your Mac.
In the archive file name, `<version>` is the current Unwired Platform version number. For example, `MobileWorkflow-2.1.2.tar.gz`.
3. Unpack `MobileWorkflow<version>.tar.gz`.
The extraction creates a `Workflow` directory.
4. In the `Workflow` directory, double-click **Workflow.xcodeproj** to open it in the XCode IDE.
5. If necessary, select **Project > Edit Active Target > ProjectName > General** to add the following frameworks from the SDK to the project:
 - `Security.framework`
 - `AddressBook.framework`
 - `QuartzCore.framework`
 - `CoreFoundation.framework`
 - `libcucore.A.dylib`
 - `libz.1.2.3.dylib`
 - `libstdc++.dylib`
6. In XCode, select **Build > Build** to build the project.

Installing the Mobile Workflow Container from the Apple App Store

Install the Mobile Workflow container from the Apple App Store.

This is a free version of the Sybase Mobile Workflow and should not be used for enterprise deployment.

1. On your device, on the iOS home page, tap **App Store**.
2. Search for **Sybase Mobile Workflow**.
3. In the search results find the version of the Sybase Mobile Workflow container to install and click **Free**.
4. Tap **Install** to download the application.
5. In **Settings > Workflow<version>**, for Connection Info, enter:

a) In Connection Info, enter:

- Server Name – the machine that hosts the server where the mobile application project is deployed.
- Server Port – Unwired Server port number. The default is 5001.
- Farm ID – the farm ID you entered when you registered the device in Sybase Control Center.
- User Name – the user you registered in Sybase Control Center.
- Activation Code – the activation code for the user, for example, 123.
- Protocol - HTTP or HTTPS. The protocol to connect to the Relay Server with. The default is HTTP.
- Password – enter your password so that the Workflow container registers using the automatic registration option.

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.

- URL Suffix (Optional) – the URL suffix used to connect to a Relay Server.
6. Scroll to the page that contains the **Workflow** icon, then tap to launch.
 7. Enter your personal identification number (PIN). Choose the number that you need to enter to start the Mobile Workflow application. This PIN is a security measure to safeguard your company's data.
 - 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 **Workflows** page appears.

8. Click the **Messages** tab bar, then tap **Messages** to view the Workflows.
9. (Optional) If instructed by your system administrator, enable notifications on your device.

Installing the Mobile Workflow Application Using iTunes

Install the Mobile Workflow application using iTunes.

1. Launch iTunes.
2. Download the application from your corporate network to your Applications library.
3. Sync the Mobile Workflow application to your Apple mobile device.
4. Specify the connection settings in **Settings > Workflow**.

Installing the Mobile Workflow Container on Windows Mobile Devices

Install and configure the Hybrid Web Container required to prepare a Windows Mobile device to run Mobile Workflow packages.

1. Navigate to <UnwiredPlatform_InstallDir>\UnwiredPlatform\MobileSDK\HybridWeb\WM.
2. Copy the Windows Mobile Professional device file, SybaseMobileWorkflow.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 SybaseMobileWorkflow.cab file from the Windows Mobile device. This installs the container.
6. In Programs, click the Workflow Settings icon to open the Mobile Workflow settings connection screen.
7. In the Workflow Settings Connection screen, enter the connection settings. These settings should match the values you used when you registered the device in Sybase Control Center.
 - Server Name – the machine that hosts the server where the mobile application project is deployed.
 - Server Port – Unwired Server port number. The default is 5001.
 - Farm ID – the farm ID you entered when you registered the device in Sybase Control Center..
 - User Name – the user you registered in Sybase Control Center.
 - Activation Code – the activation code for the user, for example, 123.
 - Enable Automatic Registration – when you select this option, the Registration 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.

- URL Suffix (Optional) – the URL suffix used to connect to a Relay Server.
-

Note: Select the right arrow icon (→) to view the container log. This is useful for checking the connection, or retrieving other debugging information.

8. From the Start menu, select Sybase Mobile Workflow.
9. Select a Mobile Workflow package from the list of available packages.



Preparing BlackBerry Devices for the Mobile Workflow Package

Install the Mobile Workflow container on the BlackBerry device using BlackBerry Desktop Manager.

Prerequisites

For prerequisites and complete information about provisioning BlackBerry devices see *BES Provisioning for BlackBerry* in *System Administration > Device Provisioning*.

Task

1. Connect the BlackBerry device to the computer that contains the Mobile Workflow 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 mobile workflow .cod and .alx container files, <UnwiredPlatform_InstallDir>\UnwiredPlatform\MobileSDK\HybridWeb\BB.
 - CommonClientLib
 - MessagingClientLib
 - MocaClientLib
 - Workflow
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 Mobile Workflow Container on BlackBerry Devices Over the Air

Your system administrator must provide the appropriate information for installing the Mobile Workflow container over the air, and the BlackBerry Exchange Server (BES) must be available.

Note: For complete information about provisioning BlackBerry devices see *BES Provisioning for BlackBerry* in *System Administration > Device Provisioning*.

Your administrator sends you either:

- A message that contains a link to the Mobile Workflow container installation file. Select the link, which starts the installation process.

- A URL to the Mobile Workflow container installation file, which you indicate in the BlackBerry browser. Select **Download** to start the installation process.

Configuring the BlackBerry Simulator for Mobile Workflow Packages

Copy the .cod files 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 <UnwiredPlatform_InstallDir>\UnwiredPlatform\MobileSDK\HybridWeb\BB.
4. Select the required .cod files, then click **OK**. These include:
 - `CommonClientLib.cod` – shared code that can be used by native Sybase Unwired Platform BlackBerry applications.
 - `MessagingClientLib.cod` – the main Messaging library, shared code that can be used by native SUP BlackBerry applications.
 - `MocaClientLib.cod` – messaging library.
 - `Workflow.cod` – the main Mobile Workflow application and Sybase Settings, where the user enters the server connection information.

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 BlackBerry Connection Settings

Configure the connection settings for the Mobile Workflow application.

1. Click the Workflow icon on the applications screen, then press the Menu key and select **Settings**.
2. Enter the settings for the Mobile Workflow application:
 - **Server Name** – the machine that hosts the server where the mobile application project is deployed.
 - **Server Port** – Unwired Server port number. The default is 5001.
 - **Farm ID** – the farm ID you entered when you registered the device in Sybase Control Center..
 - **User Name** – the user you registered in Sybase Control Center.

- Activation Code – the activation code for the user, for example, 123.
- Enable Automatic Registration – when you select this option, the Registration 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.

- URL Suffix (Optional) – the URL suffix used to connect to a Relay Server.

Select **Menu > Save** to save the settings.

3. Start the Mobile Workflow application, then view the settings log to verify that the connection is active.

In Workflows, select **Settings**. On the Connection settings screen, select **Show Log**.

Configuring Android Connection Settings

Configure the connection settings for the Mobile Workflow application.

1. Click the Workflows icon on the applications screen, then select **Settings**.
2. Enter the settings for the Mobile Workflow application:
 - Server Name – the machine that hosts the server where the mobile application project is deployed.
 - Server Port – Unwired Server port number. The default is 5001.
 - Farm ID – the farm ID you entered when you registered the device in Sybase Control Center..
 - User Name – the user you registered in Sybase Control Center.
 - Activation Code – the activation code for the user, for example, 123.
 - Enable Automatic Registration – when you select this option, the Registration 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.

- URL Suffix (Optional) – the URL suffix used to connect to a Relay Server.

Select **Save** to save the settings.

3. Start the Mobile Workflow application, then view the settings log to verify that the connection is active.

From the Mobile Workflow application, tap **Settings > Show Log**.

Configuring Windows Mobile Connection Settings

Configure the connection settings.

Prerequisites

Install the Sybase messaging runtime client.

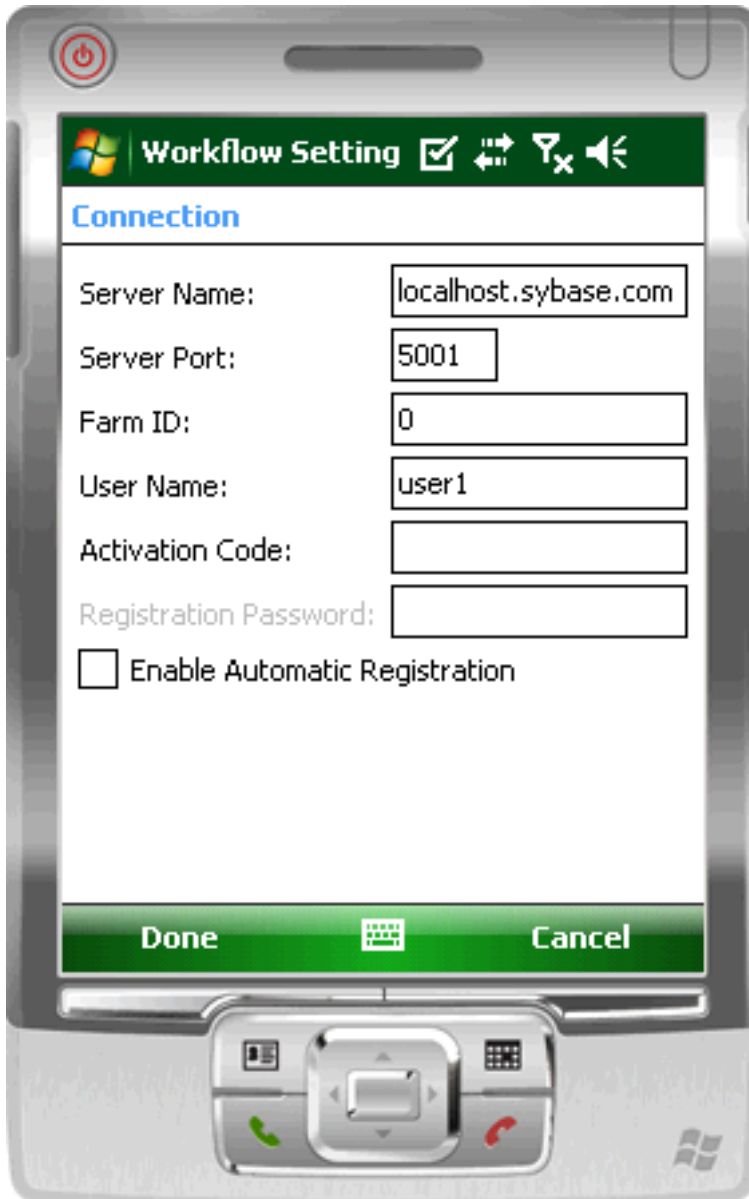
Task

1. Select **Start > Programs > Sybase Settings**.
2. In the Sybase Settings screen, click **Connection**.
3. 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 – Unwired Server port number. The default is 5001.
 - Farm ID – the farm ID you entered when you registered the device in Sybase Control Center..
 - User Name – the user you registered in Sybase Control Center.
 - Activation Code – the activation code for the user, for example, 123.
 - Enable Automatic Registration – when you select this option, the Registration 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.

- URL Suffix (Optional) – the URL suffix used to connect to a Relay Server.

Note: The URL Suffix setting is in Advanced settings.



Note: "localhost" is name of your machine.

4. Click **Done**.
5. Start the Mobile Workflow application, then view the settings log to verify that the connection is active.

Select tap **Start > Programs > Sybase Settings > Menu > Show Log**

Configuring iOS Connection Settings

Configure the settings for the Mobile Workflow application.

1. Go to the device Settings screen and click **WorkFlows**.
2. Enter the settings for the Mobile Workflow application:
 - Server Name – the machine that hosts the server where the mobile application project is deployed.
 - Server Port – Unwired Server port number. The default is 5001.
 - Farm ID – the farm ID you entered when you registered the device in Sybase Control Center..
 - User Name – the user you registered in Sybase Control Center.
 - Activation Code – the activation code for the user, for example, 123.
 - Protocol - HTTP or HTTPS. The protocol to connect to the Relay Server with. The default is HTTP.
 - Password – enter your password so that the Workflow container registers using the automatic registration option.

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.

- URL Suffix (Optional) – the URL suffix used to connect to a Relay Server.

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

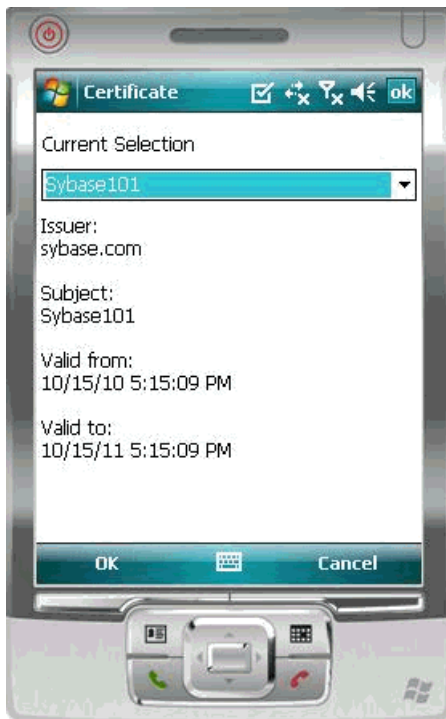
Select an X.509 certificate to use for Mobile Workflow application user authentication.

Prerequisites

1. Create a Mobile Workflow application that prompts the user to specify a certificate as credentials.
2. Package and assign the Mobile Workflow application to a Windows Mobile device user.

Task

1. From **Inbox > Workflows**, select the Mobile Workflow.
2. Select the **Specify Certificate Credentials** menu item from the Certificate Picker.
3. Select the certificate (for example, Sybase101) and continue with the Mobile Workflow.



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 Mobile Workflow application user authentication.

Prerequisites

1. Create a Mobile Workflow application that prompts the user to specify a certificate as credentials.
2. Package and assign the Mobile Workflow application to an Android device user.

Task

1. On the Android device or emulator, in applications, click **Workflow**.
2. Select the Mobile Workflow 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

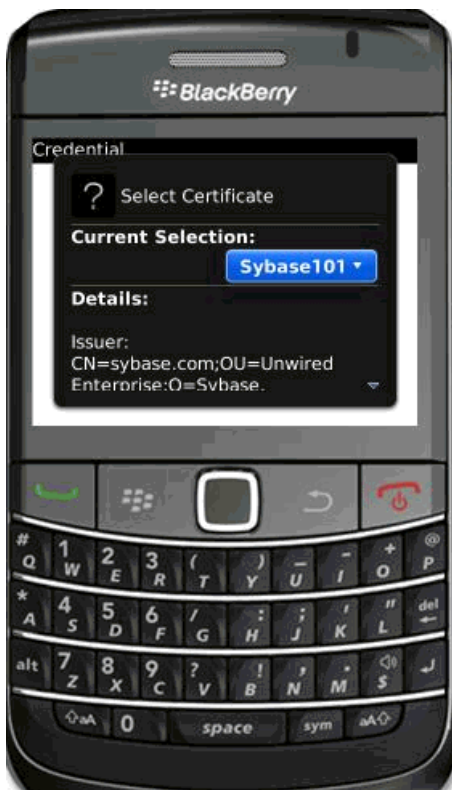
Select an X.509 certificate to use for workflow application user authentication.

Prerequisites

1. Create a Mobile Workflow application that prompts the user to specify a certificate as credentials.
2. Package and assign the Mobile Workflow application to a BlackBerry device user.

Task

1. From **Inbox > Workflows**, select the mobile workflow.
2. From the Certificate Picker, select the **Specify Certificate Credentials** menu item.
3. Select the certificate (for example, Sybase101) and continue with the workflow application.



Installing X.509 Certificates on iOS Devices

Add an authentication screen to the workflow client from which you can authenticate with a generated X.509 certificate instead of a user name and password combination.

1. Copy the X.509 certificate used for authentication into a directory on the same host as Unwired Server. For example, `c:\certs`.
2. Create a registry string value on Unwired Server at `HKLM\Software\Sybase\Sybase Messaging Server\CertificateLocation` and populate it with the path. For example, `c:\certs`.
3. Name the X.509 certificate file as `domain_user.p12`, where *domain* is the Unwired Server domain and *user* is the certificate user. The user must have read permission for `.p12` file.
4. The system administrator must ensure the specified domain\user has “log on as batch job” permission on the Windows machine on which Unwired Server runs:
 - a) Double-click **Control Panel > Administrative Tools > Local Security Policies**.
 - b) Expand **Local Policies** and select **User Rights Assignment**.
 - c) Right-click **Log on as a batch job** and select **Properties**.
 - d) Select **Add User or Group** and add the domain\user.
5. The account under which Unwired Server runs must have adequate permissions to impersonate the domain\user, for example, the Administrator account for the domain.

Testing X.509 Certificates on iOS Devices and Simulators

Select an X.509 certificate for Mobile Workflow application user authentication to test.

Prerequisites

1. Create a Mobile Workflow application that prompts the user to specify a certificate as credentials.
2. Package and assign the Mobile Workflow application to an iOS device user.

Task

1. During device application development, define and add a screen that has a Certificate Picker menu item.
2. Generate and deploy the application to the iPhone client.
3. Select **Certificate Picker** from the iPhone client.
4. Enter Windows credentials and certificate password in the dialog and click **Done**. Make sure the format is *domain|user*.
5. Submit the credentials to Unwired Server.

Manage a Mobile Workflow Package

The Workflows node in Sybase Control Center allows administrators to view and manage deployed Mobile Workflow packages, including Mobile Workflow display name, module name, and module version.

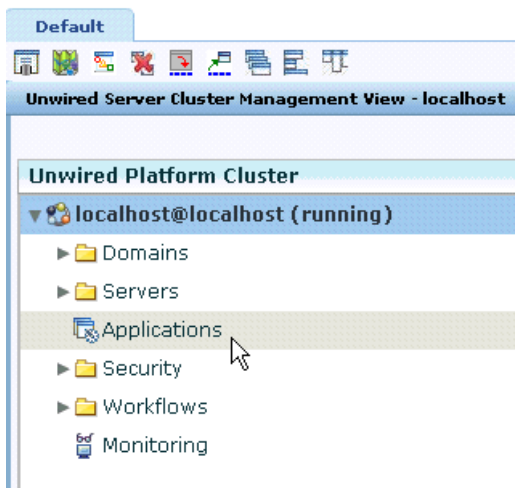
Administrators deploy Mobile Workflow packages into the Unwired Platform cluster through this node, as well as manage e-mail settings configuration.

Registering and Reregistering Mobile Workflow Application Connections

Use Sybase Control Center to trigger the registration and application activation process for Mobile Workflows.

Note: When using a Windows Mobile emulator or BlackBerry simulator to register an application connection in Sybase Control Center, the device ID changes each time you reset the emulator to factory settings and reinstall the client. Before reinstalling, you must delete the original application connection from Unwired Server. Then, reregister the application connection. Otherwise, the device log shows a `Wrong Device for Code` error when the device attempts to connect after registration. This problem occurs with Windows Mobile emulators and BlackBerry simulators.

1. Log in to Sybase Control Center.
2. In the left navigation pane, click the **Applications** node.



3. In the right administration pane, click the **Application Connections** tab.

4. Click **Register** to register the Mobile Workflow application connection to a device, or **Reregister** to update the application connection for an existing application connection.
5. In the **Register Application Connection** or the **Reregister Application Connection** dialog:
 - a) For new registrations only, type the name of the user that will activate and register the Mobile Workflow application. For reregistrations or clones, the same name is used and cannot be changed.
 - b) Select the **HWC** template.
The template you choose supplies initial values in the subsequent activation fields.
6. Change the default activation field values for the template you have chosen.
If you are using a relay server, ensure the correct values are used.
 - **Server name** – the DNS name or IP address of the primary Unwired Server, such as "myserver.mycompany.com". If using relay server, the server name is the IP address or fully qualified name of the relay server host.
 - **Port** – the port used for messaging connections between the device and Unwired Server. If using relay server, this is the relay server port. Default: 5001.
 - **Farm ID** – a string associated with the relay server farm ID. Can contain only letters A – Z (uppercase or lowercase), numbers 0 – 9, or a combination of both. Default: 0.

Note: If the device uses relay server to connect to Unwired Server, the farm ID should be the name of the Unwired Server farm configured in the relay server for messaging-based synchronization applications. If the device connects to Unwired Server directly, the farm ID should be 0.

 - **Activation code length** – the number of characters in the activation code. If you are reregistering or cloning an application connection, this value cannot be changed.
 - **Activation expiration** – the number of hours the activation code is valid.
7. (Optional) Select the check box adjacent to **Activation Code** to enter the code sent to the user in the activation e-mail. This value can contain letters A – Z (uppercase or lowercase), numbers 0 – 9, or a combination of both. Acceptable range: 1 to 10 characters.
If the activation code is automatically generated, the code for the application connection can be retrieved from the **Connections** group of the **Application Connection Properties** dialog
8. Click **OK**.

Enabling and Configuring the Notification Mailbox

Configure the notification mailbox settings that allow Unwired Server to transform e-mail messages into mobile workflows.

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 mobile workflow 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 Sybase Control Center.
2. In the left navigation pane, click **Workflows**.
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 Unwired Server and the e-mail server in your environment.
 - **Server** and **Port** – configure these connection properties so Unwired 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 Unwired 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 Mobile Workflows

Assign mobile workflow packages to make them available to a device user. Unassign them when a package is no longer required.

1. In the left navigation pane of Sybase Control Center, click **Workflows > <Mobile_WorkFlow_Package>**.
2. In the right administration pane, click the **Application Connections** tab.
3. Locate the device to assign a mobile workflow package to, then:
 - a) Click **Assign Workflow**.
 - b) List the activation users to assign the mobile workflow package to.

By default, no users are listed in this window. 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.

- c) Select the user or users from the list to which to assign the mobile workflow package.
- d) Click **OK**.

4. To unassign a mobile workflow package, select the User and click **Unassign Workflow**.

Activating the Workflow

The menu items on a Workflow screen can be either a Submit Workflow (asynchronous) or Online Request (synchronous) menu item type.

To complete the mobile workflow activation process, the last screen in the mobile workflow application must have a Submit Workflow menu item. This is necessary for the device and server-side to activate the mobile workflow for the device.

Mobile workflows are considered to have been processed and/or activated only if they are closed with a Submit Workflow menu item, and which may, or may not, have a corresponding mobile business object (MBO) operation tied to it.

Configuring Context Variables for Mobile Workflow 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, click **Workflows > <Workflow_Package>:<Workflow_Version>**.
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 mobile workflow application user ID that Unwired Server uses to authenticate the user. Depending on the security configuration, Unwired Server may pass that authentication to an EIS.

Context Variable	Description
SupUnrecoverableErrorRetryTimeout	After sending a JSON request to Unwired Server, if you receive an EIS code that indicates an unrecoverable error in the response log, the mobile workflow 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.
SupRecoverableErrorRetryTimeout	After sending a JSON request to Unwired Server, if you receive an EIS code that indicates a recoverable error in the response log, the mobile workflow 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 mobile workflow application user password that Unwired Server uses to authenticate the user. Depending on the security configuration, Unwired 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 deployed Mobile Workflow package. This cannot be changed.
SupMaximumMessageLength	Use this property to increase the allowed maximum Mobile Workflow message size. Limitations vary depending on device platform: <ul style="list-style-type: none"> • For BlackBerry 5, the limit is 512KB. • For Windows Mobile the limit is 500KB. • 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.

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, click **Workflows** > **<Workflow_Package>:<Workflow_Version>**.
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 Mobile Workflow Package

The administrator can change the credential certificate assigned at design time.

Note: Sybase recommends that you use Internet Explorer to perform this procedure.

1. In the left navigation pane, click **Workflows** > **<Workflow_Package>:<Workflow_Version>**.
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.

Security

Set up static or dynamic authentication, and configure the Mobile Workflow application to use credentials.

Credentials

You can use either dynamic or static credentials in a mobile workflow form.

See the *Developer Guide for Mobile Workflow Packages* and *System Administration* for more detailed information about implementing security and certificates.

The user name and password values are required when the mobile workflow application invokes a mobile business object operation. These authentication values can be provided statically (at design time), or dynamically (by the user at runtime).

Dynamic credentials mean that the user sets the user name and password on a screen that is pointed to by the credential request starting point. The text fields must have the corresponding

Credential Cache User Name and Password checkbox checked to indicate the value is to be used to provide the user name and password on the client. When the user logs in, the credentials are authenticated using the stored credentials.

Note: If an e-mail triggered workflow form has dynamic cached credentials, the cached credentials are not cached between invocations of the workflow form through an email 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 mobile workflow application can be extracted from the selected Sybase Unwired Platform profile user name and password when the mobile workflow application is generated, or they can be hard-coded using the Properties view. After deployment, you can change static credentials in the Sybase Control Center.

The application can also have a credential screen (Credential Request) that appears if the mobile workflow application 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 SUP Server connection profile authentication – selected by default. When the code is generated for the mobile workflow application, the user name and password associated with the SUP connection profile are used.
 - 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 – when you select this option, you can use a certificate to generate authentication credentials.
4. (Optional) If you selected the **Use hard-coded credentials** option 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 when you want 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 mobile workflow, 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 mobile workflow.

1. In the Mobile Development perspective, select **File > New > Mobile Workflow Forms Editor**.
2. Follow the instructions in the New Mobile Workflow Forms Editor wizard:

Field	Description
Enter or select the parent folder	Select the mobile application project in which to create the mobile workflow form.
File name	Enter a name for the mobile workflow form. The extension for mobile workflow forms is .xbw.
Advanced	Link the mobile workflow form to an existing file in the file system.

Field	Description
Link to file in the file system	<p>Click Browse to locate the file to which to link the mobile workflow form. 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.</p> <p>Click Variables to define a new path variable. Path variables specify locations on the file system.</p>

3. In the Starting Points page, select **Credentials (authentication) may be requested dynamically from the client application**.
4. Continue with the New Mobile Workflow wizard as appropriate to create the type of mobile workflow application you want to create. Click **Finish**.
5. When the Mobile Workflow Forms Editor opens, click **Flow Design**.
The Credential Request starting point and its associated Credential Request screen appear on the Flow Design page.
Select the Credential Request starting point. You see the two pre-defined keys (cc_username and cc_password) in the Properties view.
6. Double-click the **Credential Request** screen to go to the Screen Design page.
Two editbox controls, which are bound to the pre-defined cc_username and cc_password keys appear on the screen.
7. Select the **Username** editbox, then click **Advanced** on the left side of the Properties view.
The Username editbox has the **Credential cache username** checkbox selected. Select the Password editbox and note that it has the **Credential cache password checkbox** checked.
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 the Properties.
 - c) In the Properties view, from Type, choose **Select Certificate**.
When the user selects the menu item on the device, a dialog is opened that allows the user to select a certificate to use for credentials.
9. Select **File > Save**.
The first time the mobile workflow is started following deployment, the credential screen is shown. The username and password values are then cached in the credential cache.

Note: If an e-mail triggered workflow form has dynamic cached credentials, the cached credentials are not cached between invocations of the workflow form through an email trigger.

Configuring the Workflow Application to Use Credentials

Configure a Mobile Workflow application to pass user credentials, which are authenticated by Unwired Server and the EIS.

For information about configuring and implementing X.509 and SSO2 on the server, see the Sybase Unwired Platform *Security* documentation.

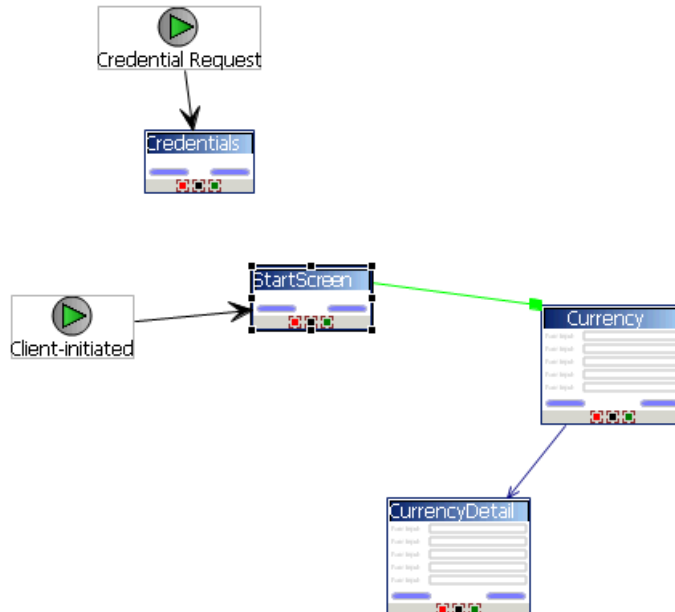
Configuring the Workflow Application to Use X.509 Credentials

Add a screen that contains a Specify Certificate Credentials menu item to the Credential Request starting point from which a workflow application user selects a certificate to gain access to the MBO and related resources.

1. In the Mobile Workflow Forms Editor, add a **Credential Request** starting point to the workflow application.
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 Workflow type.

On the device, the Specify Certificate Credentials action prompts the user for a *.p12 certificate and passes it to Unwired 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 Workflow Application to Use Static X.509 Credentials

When using static credentials, the workflow application does not prompt the user for credentials, instead it passes the credentials to Unwired Server automatically and displays the workflow application's start screen.

1. Remove the Credential Request starting point and screen from the workflow application (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 Mobile Workflow package, and reassign it to a device.

Propagating a Client's Credentials to the Back-end Data Source

Use client credentials 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 Unwired 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.

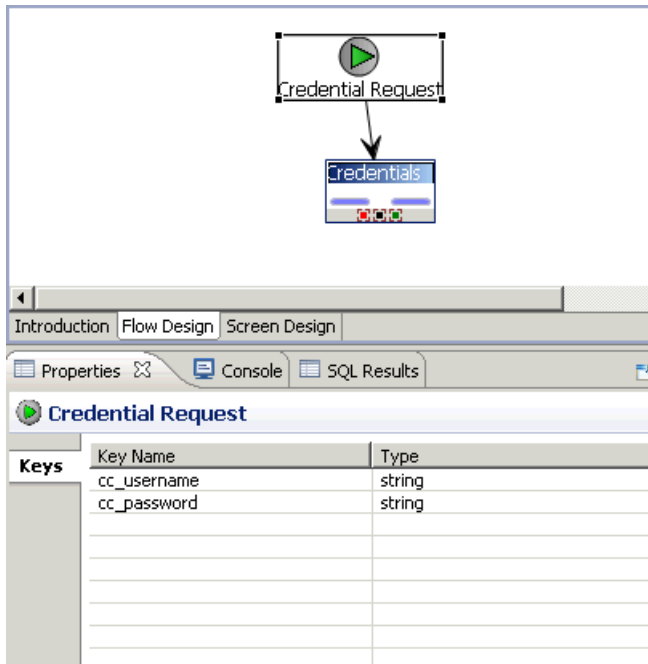
Note:

- 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 Sybase Control Center (SCC).
 - When you create a new security configuration that includes the SAPSSOTokenLoginModule, and deploy it to a new domain, if the mobile workflow application uses the MBOs associated with the new security configuration, you must specify an Unwired Server domain that corresponds to the domain using the security configuration. See the Sybase Unwired Platform *Security* guide for more information about security configurations
-

Configuring a Workflow Application to Use SSO2 Tokens

Configure a Credential Request starting point from which a workflow application user can pass a user name and password to gain access to the MBO and related resources.

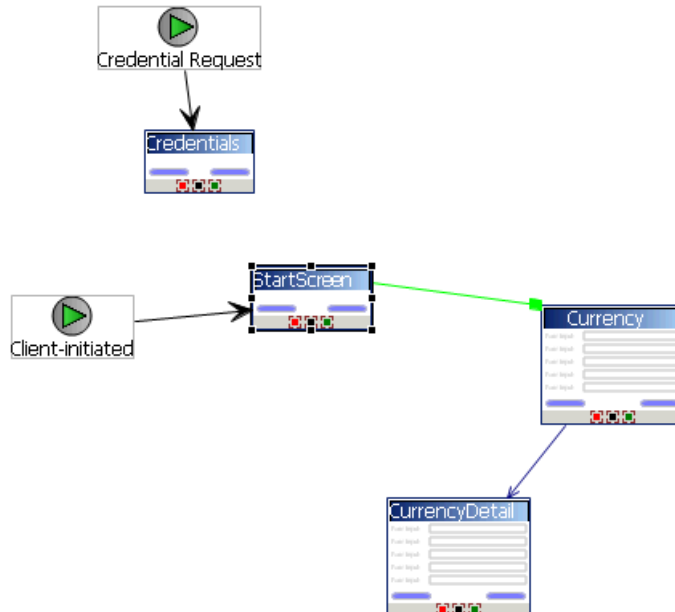
1. In the Mobile Workflow Forms Editor, add a **Credential Request** starting point to the workflow application.
2. Add two keys to the Credential Request named `cc_username` and `cc_password`.
3. Add a screen named `Credentials` and connect it to the Credential Request starting point.



4. Double-click **Credentials** to open it in the Screen Design. 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 workflow application. You could also add a **Submit workflow** menu item instead of **Save screen**.

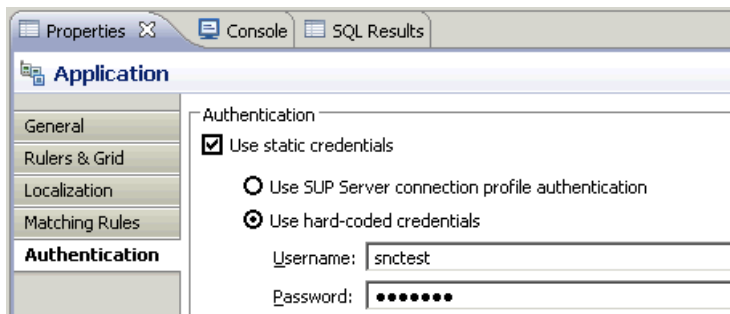
5. Add a Client-initiated starting point to which you add screens that contain the Submit menu items used to run MBO operations and object queries, return and display results, and so on. These actions all use the same credentials created in the previous steps.



Configuring the Workflow Application to Use a Static SSO2 Token

When using static credentials, the workflow application does not prompt the user for credentials, instead it passes the credentials to Unwired Server automatically and displays the workflow application's start screen.

1. Remove the Credential Request starting point and screen from the workflow application (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 Sybase Control Center for the server connection (for example: snctest/snctest).



3. Save and regenerate the workflow package, and reassign it to a device.

Modify Certificate Information for Workflow Packages

If using static credentials, either SSO token or static x.509 certification, you can replace the workflow package certificate using either Sybase Control Center or the SUPMobileWorkflow.replaceMobileWorkflowCertificate() API. To replace a certificate, you must have access to the certificate file and password.

Replacing the Mobile Workflow Certificate Through Sybase Control Center

If using static credentials, you can set or modify the context variable certificate settings for a mobile workflow package from Sybase Control Center.

The mobile workflow certificate password context variable is read-only. You can modify this only by using the Admin Java API method

SUPMobileWorkflow.replaceMobileWorkflowCertificate().

1. From Sybase Control Center, navigate to **Workflows > WorkflowName**, where *WorkflowName* is the name of the workflow package.
2. On the Context Variables tab, verify that SupUser and SupPassword contain valid credentials for the specified security configuration, for workflow packages that do not use certificate-based authentication.
3. For workflow packages that use certificate based authentication, you can view these context variables:
 - SupCertificateIssuer
 - SupCertificateSubject
 - SupCertificateNotAfter
 - SupCertificateNotBefore

Replacing the Mobile Workflow Certificate Using the Admin API

Use the SUPMobileWorkflow.replaceMobileWorkflowCertificate() method to set or modify the certificate password context variable for the workflow package.

```
InputStream is = workflowRL.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();
MobileWorkflowIDVO workflowID = new MobileWorkflowIDVO();
workflowID.setWID(4);
workflowID.setVersion(1);

workflow.replaceMobileWorkflowCertificate(workflowID,
    baos.toByteArray(), "password");
```

Content Security on Devices

This explains how the files that make up the mobile workflow container are protected when stored on the device, and under what circumstances the files are stored in plaintext.

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 `<workflow_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 Workflow 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 `<workflow_package_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 workflow application 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 mobile workflow application 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 mobile workflow application 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 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 into a SQLite database on the device.

The crypto 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 `<workflow_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 SQLite database, 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. The Content Provider URL is further obfuscated with a randomly generated number that is required on the URL when the files are requested.

Attachments

If attachments, such as *.docx, *.pdf, and so on, are part of the `<workflow_package_name>.zip` deployed to the device, they are stored in the encrypted SQLite database after they have been encrypted through the Google/Android crypto libraries.

- When the JavaScript requests these attachments for viewing, they are read from the SQLite database, and temporarily written unencrypted to the device's flash memory for the external viewers to display them.
- Once the mobile workflow 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 mobile workflow 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 crypto 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 crypto 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 Workflow 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 crypto libraries before it is stored into the SQLite database.

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 `<workflow_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 Workflow zip package. When the iOS device's browser control requests these Web files, they are read from the encrypted SQLite database and passed to the browser control into memory, which means there are no temp files.

Attachments

If attachments, such as *.docx, *.pdf, and so on, are part of the `<workflow_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 mobile workflow 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 mobile workflow 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 Workflow 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 a Workflow 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, Sybase 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\Sybase\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\Sybase\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 <workflow_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 Workflow 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 <workflow_package_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 mobile workflow application closes, these temporary attachment files are immediately removed.

Images

Images are stored unencrypted on the file system, then removed when the Workflow application 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 Workflow 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 Mobile Workflow Forms Editor, such as the names of screen controls, screens, and mobile business objects.

You can localize the mobile workflow by creating locale properties files. You can then load, update, and generate localized mobile workflow applications.

All the localizable strings in the Mobile Workflow Forms Editor XML model work as resource keys in the localization properties file. All the localization properties files are in the same directory as the Mobile Workflow packages (.xbw files).

Resource keys are divided into these categories, which include all the elements of the Mobile Workflow Forms Editor XML model:

- Menus
- Controls
- Screens

Localization consists of two levels of localization—the Mobile Workflow Forms Editor XML model localization and the Mobile Workflow client localization.

All locale properties files are saved in the same directory as the Mobile Workflow package.

To ensure that the correct locale is picked up for the Mobile Workflow 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\workflow*.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\workflow*.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\workflow*.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\workflow*.html`, that file is used and the HTTP lang parameter is set to "".

Localization Limitations

Some restrictions for the locale properties files apply:

- Traditional Chinese characters are not supported on iOS.
- Mobile workflow applications 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 mobile workflow 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 Mobile Workflow Package

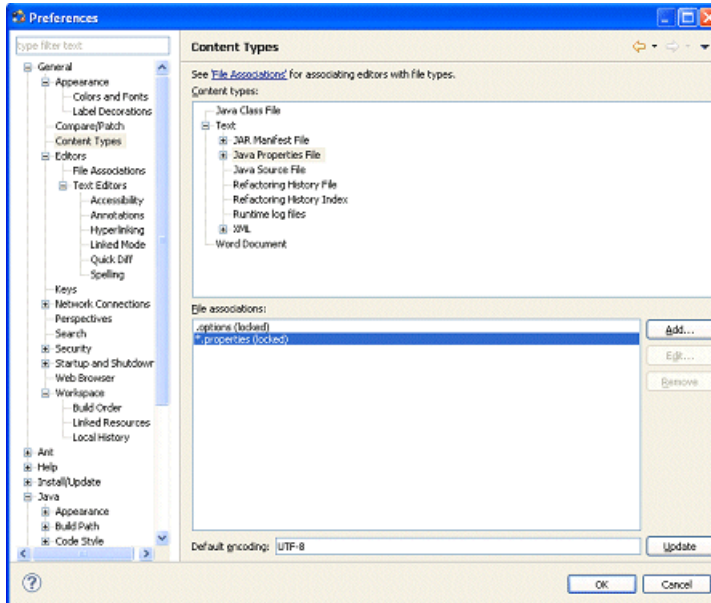
Use the Mobile Workflow Forms Editor to complete these tasks to localize Mobile Workflow packages (.xbw files).

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 Sybase Unwired 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(locked)`.
5. In the Default encoding field, change ISO-8859-1 to **UTF-8**, and click **Update**.



Creating and Validating a New Locale Properties File

Goal: Create a locale properties file as the default locale.

Prerequisites

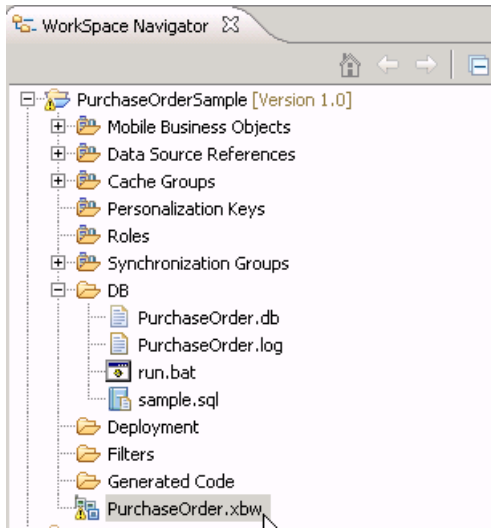
You must have an existing Mobile Workflow package before you create the locale properties file.

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 <mobile_workflow> .xbw file to open the Mobile Workflow Forms Editor.



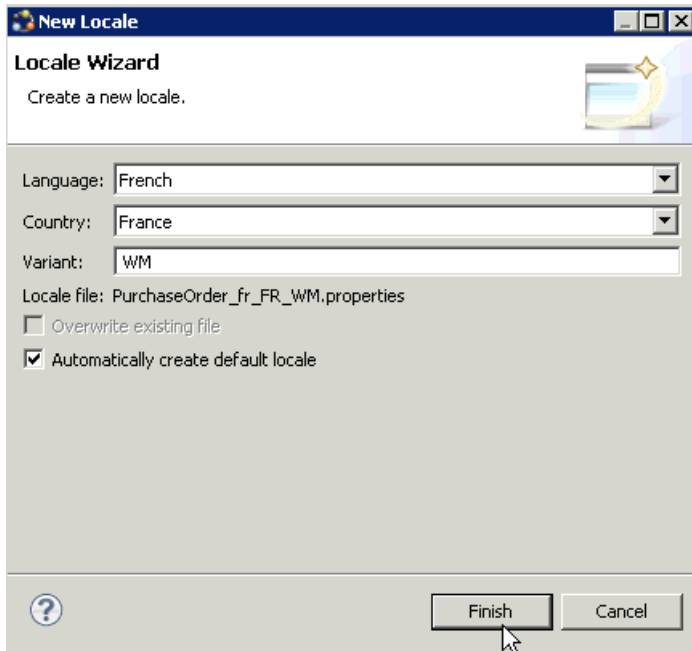
2. Click the **Flow Design** tab.
3. Right-click in a blank area on the Flow Design page, and select **Show Properties View**.
4. In the Properties view, on the left, click the **Localization** tab.
5. In the right pane, click **New**.
6. Select or enter the information for the new locale, select **Automatically create default locale**, and click **Finish**.

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.

Option	Description
Automatically create default locale	Automatically create the default locale properties file. For example, if you specify the language as "English" and the country as the "United States" for a device application called test, then both test_en_uS.properties and test.properties files are created.

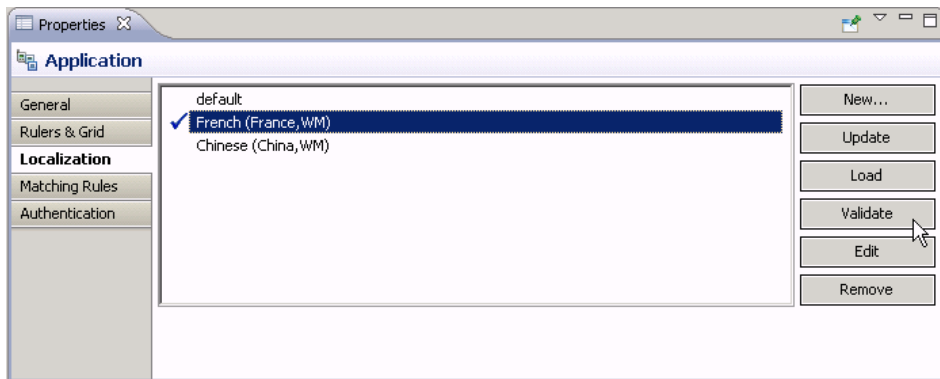
For example:

- 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**.



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

Goal: 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.

Deploying the Mobile Application Project

Goal: Deploy the mobile business objects to Unwired Server.

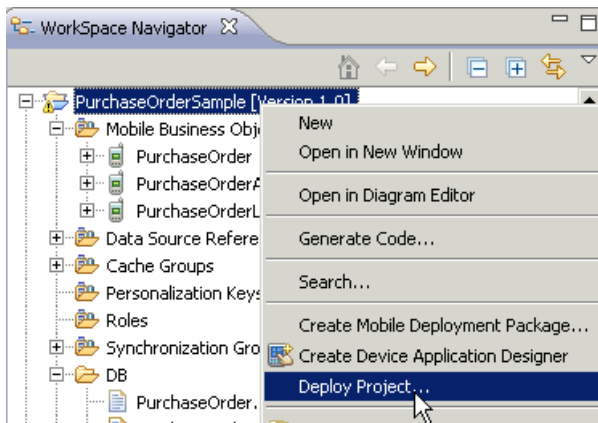
Prerequisites

Unwired Server must be running.

Task

Use this procedure to deploy any mobile application project you are using.

1. In WorkSpace Navigator, right-click the **<mobile_application_project>** folder, and select **Deploy Project**.



2. In the Deploy Mobile Application Project wizard, make your selections and click **Next**.
3. In the Contents page, select all the mobile business objects (MBOs) and click **Finish**.
4. In the Deployment status window, click **OK**.
5. When the deployment completes, click **File > Close**.
6. Generate the files for the Mobile Workflow package.

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 Mobile Workflow Designer.

If the locale properties file does not already exist, it is created. If the current locale is not defined in the mobile workflow application file, the updated locale is used as the default, and the file name is `{device_application}.properties`. Otherwise, the locale defined in the mobile workflow application 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**.

Mobile Workflow Package Internationalization

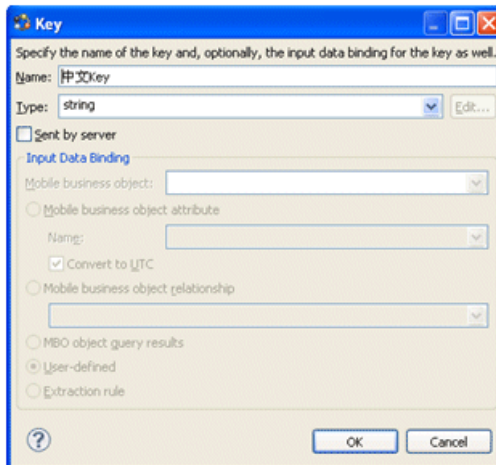
The internationalization feature depends on the internationalization setting on the operating system where Sybase Unwired Platform Mobile Workflow is running.

In the Mobile Workflow Forms Editor, you can use international data in:

- Matching rules for notifications.

The image shows two screenshots from the Mobile Workflow Forms Editor. The top screenshot displays the 'Application' window with the 'Matching Rules' tab selected. It shows a table with columns 'Type' and 'Matching Rule (regular expression)'. The first row has 'Subject' in the 'Type' column and '订单申请.*' in the 'Matching Rule' column. Buttons for 'Add', 'Delete', and 'Edit...' are on the right. The bottom screenshot shows a 'Send Notification To A Device User' dialog box. It contains fields for 'Unwired Server profile' (My Unwired Server), 'To:' (manager1), 'Cc:' (manager1), 'Bcc:' (manager1), 'From:' (manager1), 'Subject:' (订单申请 (15) 待审核), and 'Body:'. A 'Get Device Users' button is at the bottom right, and 'Send' and 'Cancel' buttons are at the bottom.

- Key names – you can create keys with names in other languages and map them to mobile business object parameters.



The 'Key' dialog box is used to specify the name and input data binding for a key. It includes fields for Name, Type, and Input Data Binding, along with checkboxes for 'Sent by server' and 'Convert to UTC'.

Specify the name of the key and, optionally, the input data binding for the key as well.

Name:

Type: [Edit...](#)

☐ Sent by server

Input Data Binding

Mobile business object:

☐ Mobile business object attribute

Name:

☒ Convert to UTC

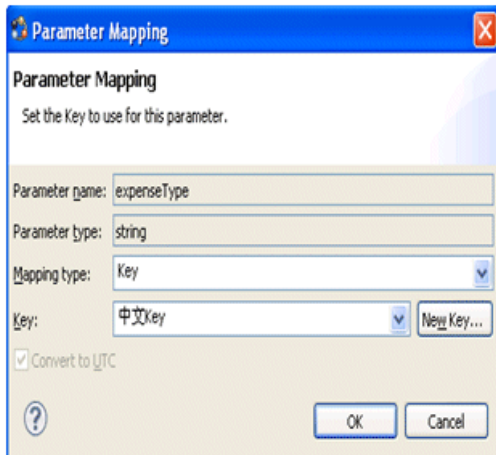
☐ Mobile business object relationship

☐ MBO object query results

☒ User-defined

☐ Extraction rule

[?](#) [OK](#) [Cancel](#)



The 'Parameter Mapping' dialog box is used to set the key to use for a parameter. It includes fields for Parameter name, Parameter type, Mapping type, and Key, along with a checkbox for 'Convert to UTC'.

Parameter Mapping

Set the Key to use for this parameter.

Parameter name:

Parameter type:

Mapping type:

Key: [New Key...](#)

☒ Convert to UTC

[?](#) [OK](#) [Cancel](#)

- 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 Mobile Workflow 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:



Test Mobile Workflow Packages

Test a Mobile Workflow on a device or simulator.

1. Launch and/or connect to the mobile device or emulator.
2. Deploy the Mobile Workflow package to the device.
3. Establish the connection to the server on the device.
4. For user-initiated Mobile Workflow packages, go to the Workflows menu of the e-mail inbox and click on the appropriate Workflow package.

5. For e-mail subscription Mobile Workflow 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 Mobile Workflow Packages

Test a server-initiated Mobile Workflow package.

1. In the Mobile Workflow Forms editor, open the Mobile Workflow form, `<workflow_form>.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 Unwired 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 Mobile Workflow form.
5. Click **Send**.
6. On the client, from the applications screen, open SUPWorkflows.
7. In the client application, click **WorkFlow Messages**. This contains the server-initiated Mobile Workflow form.

Viewing Workflow Messages on the Device

Where Workflow 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 Workflow connection settings before any Workflow packages appear on the device.

BlackBerry

To see Workflow messages on BlackBerry devices and simulators:

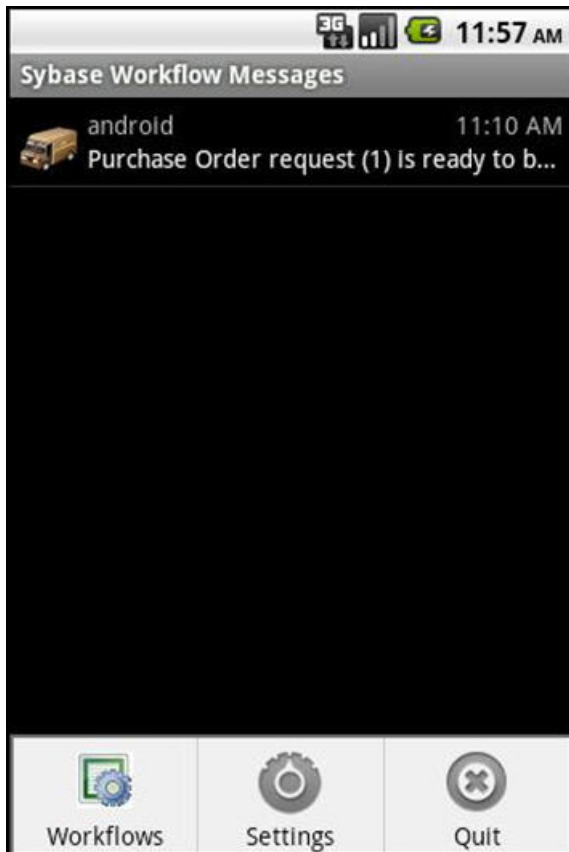
1. Close the Mobile Workflow application.
2. Go to the Messages inbox. The mobile workflow messages are in the inbox.
3. Select **Workflows** from the menu.

Android

To see Workflow messages on Android devices and simulators:

1. Open the **Workflows** application.
2. Open the Mobile Workflow application for which you want to view messages.

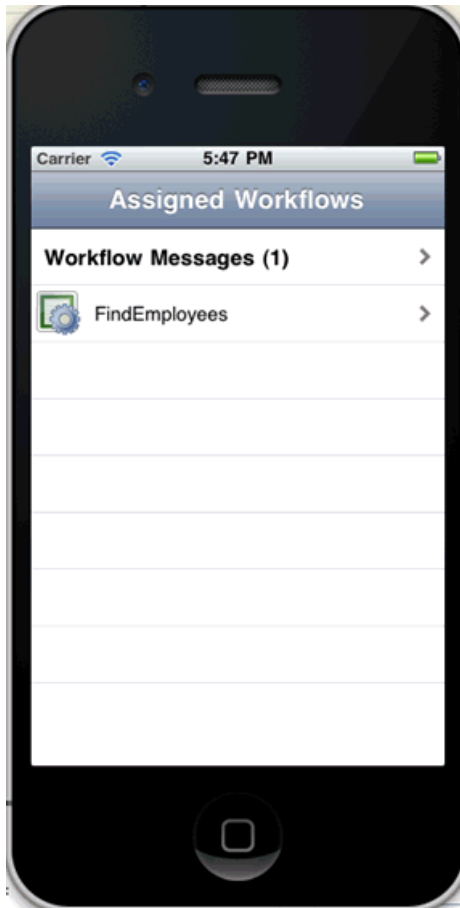
3. Click the message to view.



iOS

To see Workflow messages on iOS devices and simulators:

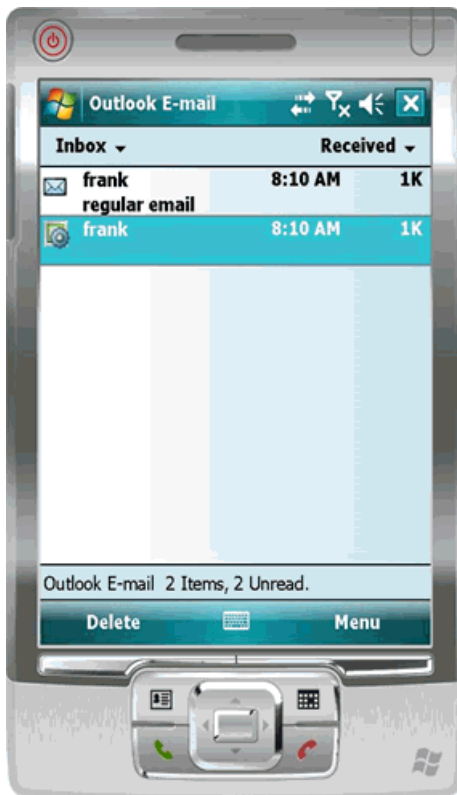
1. Open the **Workflows** application.
2. Click **Mobile Workflow Messages** to view messages.



Windows Mobile

To see Workflow messages on Windows Mobile devices and emulators:

1. Open the Outlook E-mail inbox.
2. Workflow messages are in the inbox, with a mobile workflow icon.



Launching a Server-initiated Mobile Workflow on the Device

Server-initiated Mobile Workflow messages show up as an e-mail message on the Windows Mobile or BlackBerry device inbox.

On Windows Mobile platforms, Mobile Workflow packages are integrated with the Outlook Mail inbox. On iOS, messages are sent to a container that is provided by the Workflow device client.

Click the e-mail message to launch the Mobile Workflow container and display the Mobile Workflow associated with the e-mail message.

When you click the **Workflows** menu item in the device inbox, only the latest version of the Workflows appear. When you click the Workflow icon for a particular workflow, the Workflow version that is associated with the e-mail notification is launched, whether it is the latest version or not.

Example

You develop a Mobile Workflow application that has both client-initiated and server-initiated starting points. You deploy the initial version, which is called version 1, and a Mobile Workflow e-mail notification is sent.

Next, make some changes and deploy a second version, version 2. Again, a Mobile Workflow e-mail notification is sent.

There are now three ways that this Mobile Workflow application can be launched, and the way that it is launched determines which version of the Workflow is launched:

- If you launch the application from the **Workflows** menu item, the last deployed version of the Workflow, in this case, version 2, is launched. Although version 1 of the Mobile Workflow application still exists somewhere on the device it is never used as long as you launch the Workflow from the Workflows menu.
- If you launch the Workflow by opening the initial e-mail 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 Workflow 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 Mobile Workflow package html and js 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 Mobile Workflow to Debug.
2. Open the browser to use for debugging and open the Java Console.

If you are using Chrome:

- a) Add the following command line option to the shortcut used to start Chrome:

```
..\chrome.exe" --allow-file-access-from-files
```

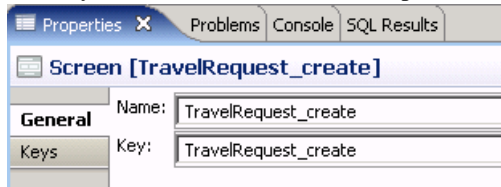
3. You can debug a client-initiated Mobile Workflow application up until the point where a menu item of the Submit Workflow type is performed. If the menu item action is an Online Request, place the XMLWidgetMessage (available in the WorkflowClient trace log located in <UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\logs\WorkflowClient) that is the expected response message into an rmi.xml file and place it at the same level as the generated workflow.html file.

Note: Control characters are not parsed correctly when using rmi.xml and Chrome to debug the Mobile Workflow. Do not format the content of the rmi.xml when debugging the Mobile Workflow 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 workflow.html file for the Mobile Workflow application 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/MobileWorkflow101/Generated%20Workflow/
travelrequest/html/workflow.html?
screenToShow=TravelRequest_create
```

7. To simulate an e-mail message triggered Mobile Workflow application:

- 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 Mobile Workflow application you are debugging, place the `transform.xml` file at the same level as the generated `workflow.html` file (`Generated Workflow\<Workflow_application_name>\html`).
- c) Add a **?loadtransformdata=true** parameter to load the data into the Workflow application.

Configuring Mobile Workflow Tracing in SCC

Change the tracing level for Mobile Workflow packages in SCC.

1. Log on to Sybase Control Center.
2. In the Unwired Platform Cluster view, select **Servers > <host_name> > Log**.
3. Click **Message Server > Settings**.
4. Select **WorkflowClient** and click **Properties**.
5. From Level, select **Debug** and click **OK**.

Mobile Workflow Package Customization

The designer-based user interface is customizable using HTML, JavaScript and CSS Web technologies.

Adding Custom Code

Use JavaScript code to customize the Mobile Workflow application.

1. Use the Mobile Workflow Package Generation wizard to generate the Mobile Workflow package and its files.

When the Mobile Workflow package is generated, the `Custom.js` file is generated if not already present in the project. The `Custom.js` file is located in `Generated Workflows\<workflow_project_name>\html\js`.

2. Right-click the `Custom.js` file and select the editor with which to open the file.
3. Add your JavaScript code.

You can also add your own separate JavaScript files to `Generated Workflows\<workflow_project_name>\html\js`, then add custom code to the `Custom.js` file that calls the functions in the JavaScript files you added. This prevents the `Custom.js` file from becoming extremely long, which makes it difficult if multiple developers are working on the same Mobile Workflow application simultaneously.

4. Save and close the `Custom.js` file.

Since the `Custom.js` file is generated only if it is not already present in the Mobile Workflow project, this file will not be re-generated if you subsequently re-generate the Mobile Workflow package, so any modifications you make are preserved.

5. Deploy the Mobile Workflow package to Unwired Server.

Generated Mobile Workflow Files

When you generate mobile workflow package files, some files are generated every time and others are generated only under certain conditions.

These files are generated every time you generate the mobile workflow package:

- `manifest.xml` – describes how the contents of the Mobile Workflow package .zip file are organized.
- `workflow_name.zip` – contains all of the Mobile Workflow files, including the Web application files, look and feel files, the JavaScript files, and so on.

These files are regenerated only if you select the **Generate** option in the Mobile Workflow Package Generation wizard:

- `workflow_name.html` – an HTML file that describes simple workflow screens and forms. Default name: `workflow.html`.
- `workflow_name_CustomLookAndFeel.html` – a workflow html file that adds Sybase JavaScript functions and CSS styles.
- `workflow_name_jQueryMobileLookAndFeel.html` – a workflow html file that adds jQuery Mobile functions and CSS styles.

- `WorkflowClient.xml` – contains metadata that specifies how to map the data in the workflow message to and from calls to Mobile Business Object (MBO) operations and object queries.
- `workflow_name.xml` – look and feel file that uses the basic `workflow_name.html` file.
- `Resources.js` – allows you to access localized string resources.
- `Workflow.js` – contains functions for common menu, screen, and database operations.

These files are generated only if you select the **Generate** option and the files do not exist:

- `API.js` and `Utils.js` – provide Mobile Workflow functions used to communicate with the Hybrid Web container.
- `Custom.js` – enables you to add JavaScript code to customize the Mobile Workflow application. Your file is preserved each time you regenerate the package.
- `WorkflowMessage.js` – provides functions to access Workflow Message resources.
- All `*.css` files – defines formatting rules to render the screens in HTML.

When you generate the mobile workflow package into the current project, the ZIP file containing the mobile workflow application and its files is placed in the Generated Workflow folder in the project, for example, `C:\Documents and Settings\username\workspace\Project_Name\Generated Workflow`. The files are shown in the Workspace Navigator. This shows the generated file structure for a project named PurchaseOrder.

The Generated Workflow\project_name folder contains:

- `html` – this folder includes:
 - `workflow.html` – contains all the screens in the Mobile Workflow application, each in its own `<div>` section. This is used on BlackBerry, Android, and iOS platforms with the **Optimize for performance** look and feel. On Windows Mobile, it is used for all looks-and-feels.
 - `workflow_customlookandfeel.html` – contains all the screens in the Mobile Workflow application. This is used with the **Optimize for appearance** look and feel on BlackBerry 5.0
 - `workflow_jquerymobilelookandfeel.html` – contains all the screens in the Mobile Workflow application. This is used with the **Optimize for appearance** look and feel on iOS, BlackBerry 6.0, and Android.
 - `js\Custom.js` – edit this file to customize the Mobile Workflow application. Since you can modify this file, it is generated only once, or when not already present in the generated files. This ensures that it is not overwritten if you subsequently regenerate the Mobile Workflow package. Examples of ways you can customize the Mobile Workflow application include:
 - Manipulating HTML elements.

- Writing code that is called before or after generated behavior is invoked for menu items.
- Implementing custom validation logic.
- `<project_name>.zip` – contains the mobile workflow application and its files, including the images, user interface, and controls

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-1.0.css` – located in Generated Workflow \<mobile_workflow_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 Workflow \<mobile_workflow_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 Workflow \<mobile_workflow_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.

BlackBerry 6.0, Android, and iOS Look and Feel

The default look and feel for BlackBerry 6.0, Android, and iOS is provided by the jQuery Mobile framework.

For this look and feel, the layout of the HTML at a high level is:

- Each screen has a block, contained in a `<div>` element, with a `data-role` of "page" and a `data-theme` of 'a'. Each `<div>` has a `<div>` with a `data-role` of "header," 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>
```

- The menu has one anchor, <a>, 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 <div> with a data-role of "content," a child element where the controls are hosted. The content <div> has a child <div> with a data-role of "scroller." This <div> in turn has a form with a number of <div>s. The "content" <div> is where you can do customizations, for example, 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> is a block for use to display help, a 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.

If you look into Custom.js file, it is recommended that you add customizations such as branding to the <div> "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 <div>s in the form correspond to the controls put on that screen during design time in the Mobile Workflow Forms editor. You might see, for example, a <div> that holds a label, <label>, and a textbox, <input>. 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.

A typical mobile workflow with this look and feel, without extraneous attributes, looks similar to this:

```
<html>
  <body onload="onWorkflowLoad();" >
    <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>
    </body>
</html>
```

BlackBerry 5.0 Look and Feel

A custom look and feel is used, by default, for BlackBerry 5.0.

Each screen has a block, a <div>. Each <div> has a form, <form>, where the controls are hosted. Each form has a number of divs. The first div has a block put aside for use to display help, a 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. All the divs in the form correspond to the controls put on that screen in the Mobile Workflow Forms Editor. You might get, for example, a <div> that holds a label, <label>, and a textbox, <input>.

A typical mobile workflow with this look and feel, without extraneous attributes, looks similar to this:

```
<html>
  <body onload="onWorkflowLoad();">
    <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>
```

Optimized for Performance Look and Feel

This is a simple look and feel you can use on all platforms.

Note: Windows Mobile 6.x Professional platforms always use the Optimized for performance look and feel, as this platform is not supported by jQuery Mobile.

Choose the **Optimized for performance** option when you configure Mobile Workflow Forms Editor 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.
- 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 Mobile Workflow Forms editor. 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 Mobile Workflow Forms Editor with this look-and-feel.

A typical mobile workflow with this look and feel, without extraneous attributes, looks similar to this:

```
<html>
  <body onload="onWorkflowLoad();">
    <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_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 Workflow client API.

Workflow Client API

Sybase Unwired Platform Mobile Workflow applications include a JavaScript API that open Mobile Workflow applications to customization, from including client-side business logic to changing the presentation layer.

Use the client API to build custom applications to support Sybase Unwired Platform Mobile Workflow features and functionality.

Public JavaScript Functions

The JavaScript files contain the functions that you can access for use with your Mobile Workflow package customization.

These JavaScript files are also included:

- Utils.js – does not contain public functions to call
- Workflow.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>

API.js

The API.js file contains several different types of functions.

They include:

General Utility Functions

This file gives you access to the Mobile Workflow general utility functions.

All of general utility functions are synchronous.

Method	Description
guid()	Generates a unique string.
S4()	This function is for use in generating a GUID.
trimSpaces(str)	Removes spaces from the specified string. str – the specified string.
convertToValidJavaScriptName(str)	Converts the specified string to one that is a valid JavaScript function name. str – the specified string.
escapeValue(str)	Replaces all instances in the specified string: <ul style="list-style-type: none"> • Of the & character with '&amp;'. • Of the < character with '&lt;'. • Of the > character with '&gt;'. • Of the " (quotation mark) character with '&quot;'. • Of the ' (apostrophe) character with '&apos;'. str – the specified string.

Method	Description
unescapeValue(str)	<p>Replaces all instances in the specified string:</p> <ul style="list-style-type: none"> • Of the '&amp;' substring with '&'. • Of the '&lt;' substring with '<'. • Of the '&gt;' substring with '>'. • Of the '&quot;' substring with '"'. • Of the '&apos;' substring with "'. <p>str – the specified string.</p>
isIOS()	Returns <code>true</code> if the Mobile Workflow application is running on an iOS platform.
isBlackBerry()	Returns <code>true</code> if the Mobile Workflow application is running on a BlackBerry platform.
isBlackBerry5()	Returns <code>true</code> if the Mobile Workflow application is running on the BlackBerry 5 platform.
isBlackBerry5WithTouchScreen()	Returns <code>true</code> if the Mobile Workflow application is running on the BlackBerry 5 platform with touchscreen capabilities.
isBlackBerry6NonTouchScreen()	Returns <code>true</code> if the Mobile Workflow application is running on a BlackBerry 6 platform without touchscreen capabilities.
isWindowsMobile()	Returns <code>true</code> if the Mobile Workflow application is running on a Windows Mobile Professional platform.
isWindows()	Returns <code>true</code> if the Mobile Workflow application is running on a Windows platform.
isAndroid()	Returns <code>true</code> if the Mobile Workflow application is running on the Android platform.
isAndroid3()	Returns <code>true</code> if the Mobile Workflow application is running on the Android 3.0 platform.
isLocaleDatetimeFormat(htmlElement)	<p>Returns <code>true</code> if the specified HTML element has an attribute indicating that it should use a locale-specific display.</p> <ul style="list-style-type: none"> • <code>htmlElement</code> – the specified HTML element. • <code>attributeName</code> – the attribute name.

Method	Description
getAttribute(htmlElement, attributeName)	Reliably returns the specified attribute value for the specified HTML element. <ul style="list-style-type: none"> htmlElement – the specified HTML element. attributeName – the attribute name.
getElementsByTagName(htmlElement, tagName)	Reliably returns the list of elements with the specified tag name, searching only the subtree underneath the specified element. <ul style="list-style-type: none"> htmlElement – the specified HTML element. tagName – the specified tag name.
isSomeFormOfParent(htmlElement, htmlElementToTest)	Determines whether the second specified HTML element is an ancestor of the first specified HTML element. <ul style="list-style-type: none"> htmlElement – the specified HTML element. htmlElementToTest – the HTML element to test.
getFormElementById(formElement, elementID)	Returns the form element with the specified ID. <ul style="list-style-type: none"> formElement – the form element. elementId – the specified ID.
getXMLHttpRequest()	Reliably returns an XMLHttpRequest object. Note: This method is supported only on BlackBerry and Windows Mobile platforms.
getURLParam(paramName)	Returns the specified parameter value from the current URL (window.location.href). paramName – the specified parameter name.

Mobile Workflow Utility Functions

Methods that allow you to access the Mobile Workflow utility functions.

All of the Mobile Workflow utility functions are synchronous.

Method	Description
getISODateString(date)	Returns a string representation of the specified date (currently in yyyy-mm-dd format only). date – the specified date.

Method	Description
<code>getLocaleDateString(date)</code>	Returns a string representation of the specified date using a locale-specific display. date – the specified date.
<code>getISODateTimeStringToDisplay(datetime, precision)</code>	Returns a string representation of the specified date with the specified precision (currently in yyyy-mm-ddThh, yyyy-mm-ddThh:mm or yyyy-mm-ddThh:mm:ss format only, depending on the precision string (HOURS, MINUTES, SECONDS)). <ul style="list-style-type: none"> datetime – the specified datetime. precision – (optional) the specified precision, determines the precision used when rounding.
<code>getLocaleDateTimeString(datetime)</code>	Returns a string representation of the specified datetime using a locale-specific display. datetime – the specified datetime.
<code>getISOTimeString(time, precision)</code>	Returns a string representation of the specified time with the specified precision (currently in hh, hh:mm, or hh:mm:ss format only, depending on the precision string—HOURS, MINUTES, or SECONDS). <ul style="list-style-type: none"> time – the specified time. precision – (optional) the specified precision, determines the precision used when rounding.
<code>getLocaleTimeString(time)</code>	Returns a string representation of the specified time using a locale-specific display. time – the specified time.

Method	Description
<code>getTimeStringToDisplayFromStr(datetime, precision)</code>	<p>Returns a string representation of the specified datetime string as a time with the specified precision—HOURS, MINUTES, or SECONDS.</p> <ul style="list-style-type: none"> <code>datetime</code> – the specified datetime. <code>precision</code> – (optional) the specified precision, determines the precision used when rounding.
<code>getDateFromExpression(toolingStr)</code>	<p>Returns a date for the specified string, which must be either a string representation of a date or of the form “today” or “today+d” or “today-d”, where <i>d</i> is a number of days.</p> <p><code>toolingStr</code> – the date as specified in the Mobile Workflow Forms editor.</p>
<code>parseBoolean(value)</code>	Returns true if the specified string is equal, in a case-insensitive way, to true.
<code>parseDateTime(value)</code>	Returns a date that corresponds to the specified string.
<code>parseTime(value)</code>	Returns a date that corresponds to the specified string.
<code>convertToSUPTYPE(typeAttribute)</code>	<p>Returns the <code>XmlWorkflowMessage</code> type for the type attribute value.</p> <ul style="list-style-type: none"> <code>typeAttribute</code> – the type of the specified HTML element.
<code>getHTMLValue(htmlElement, typeAttribute)</code>	<p>Returns a string representation of the specified HTML element’s value.</p> <ul style="list-style-type: none"> <code>htmlElement</code> – the specified HTML element. <code>typeAttribute</code> – the type of the specified HTML element.

Method	Description
setHTMLValue(htmlElement, value, screenName, adjustForUTC)	<p>Sets the value of the specified HTML element from the specified string representation of the value.</p> <ul style="list-style-type: none"> • htmlElement – the specified HTML element. • value – the new value. • screenName – the screen on which the HTML element appears. • adjustForUTC – hardware clock will adjust for coordinated universal time (UTC)
resetHTMLValue(htmlElement, screenName)	<p>Resets the value of the specified HTML element.</p> <ul style="list-style-type: none"> • htmlElement – the specified HTML element. • screenName – the screen on which the HTML element appears.

Workflow UI Functions

Functions that allow you to access the Mobile Workflow user interface (UI).

The Mobile Workflow UI functions are synchronous.

Method	Description
getCurrentScreen()	Returns the key of the current (open) screen.
setCurrentScreen(screenKey)	<p>Sets the value of the current (open) screen.</p> <hr/> <p>Note: This does not open the specified screen—this function is called only after the screen has already been opened.</p> <hr/>
getPreviousScreen()	Returns the key of the screen that was open previous to the current screen being opened, if applicable.
getListViewKey(screenName)	<p>Returns the key of the first listview on the specified screen.</p> <p>screenName – the specified screen.</p>

Method	Description
<code>navigateForward(screenKey, listViewKey)</code>	<p>Navigates from the current (open) screen to a new screen with the specified key.</p> <ul style="list-style-type: none"> • <code>screenKey</code> – the screen to open. • (optional) <code>listViewKey</code> – the listview row for which the details screen is being opened.
<code>navigateBack(isCancelled)</code>	<p>Closes the current screen and returns to the previous screen, if applicable. If the specified parameter value is false, the values on the open screen are persisted to the Mobile Workflow message if they pass validation.</p> <p><code>isCancelled</code> – true for a Cancel action, false for a Save action.</p>
<code>updateUIFromMessageValueCollection(screenName, values)</code>	<p>Updates the values of the controls on the given screen based on the contents of the specified <code>MessageValueCollection</code>. This function will rarely, if ever, need to be called.</p> <ul style="list-style-type: none"> • <code>screenName</code> – the screen. • <code>values</code> – the message value collection.
<code>updateMessageValueCollectionFromUI(values, screenName, keys, keyTypes, updateModifiedValue)</code>	<p>Updates the contents of the specified <code>MessageValueCollection</code> based on the values of the controls on the given screen. In most cases, <code>saveScreen</code> is called instead of this function.</p> <ul style="list-style-type: none"> • <code>screenName</code> – the screen. • <code>values</code> – the message value collection. • (optional) <code>keys</code> – an array of keys, which is a list of only the keys to be updated. • (optional) <code>keyTypes</code> – an array of types for the list of keys, if supplied. • <code>updateModifiedValue</code> –
<code>removeModifiedMessageValuesBasedOnCurrentScreen(values, screenName)</code>	<p>Removes the modified contents of the specified <code>MessageValueCollection</code>. This function is called when a screen is cancelled.</p> <ul style="list-style-type: none"> • <code>screenName</code> – the screen. • <code>values</code> – the message value collection.

Method	Description
saveScreen(values, screenKey, needsValidation)	<p>Saves the contents of the specified screen to the specified MessageValueCollection. This function differs from updateMessageValueCollection in these ways:</p> <ul style="list-style-type: none"> • If directed, it first performs validation on the screen. • It supports customization. • It is capable of handling the credential request screen. <p>Parameters include:</p> <ul style="list-style-type: none"> • values – the current message value collection. • screenKey – the current screen. • (optional) needsValidation – false if validation should not be done before saving, true (or unspecified) if validation should be done before saving. Returns true if saving (and validation, if requested) was successful, otherwise, returns false.
saveScreens(skipValidation)	<p>Saves the contents of all open screens if they are successfully validated.</p> <p>skipValidation –</p>

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();
}

```

Mobile Workflow Native Device Functions

Access the native features of the device using the native device functions.

Method	Description
setScreenTitle(screenKey, screenTitle)	<p>Sets the specified screen's title based on its sup_screen_title attribute value.</p> <ul style="list-style-type: none"> screenKey – the screen. (optional) screenTitle – an explicit screen title to use rather than the sup_screen_title attribute value.
closeWorkflow()	Closes the Mobile Workflow application.
addMenuItem(menuItemName, functionName, subMenuName, screenToShow)	<p>Allows the user to add a native menu item with the specified name and with the specified callback, which is invoked when the menu item is clicked.</p> <ul style="list-style-type: none"> menuItemName – the specified menu item name. functionName – the specified callback name. (optional) subMenuName – the specific submenu name for Windows Mobile platforms. screenToShow –
removeAllMenuItems()	<p>Removes all native menu items.</p> <hr/> <p>Note: Removes all menu items from the current screen. On iOS devices, all the menu buttons on the header and footer bars are removed. On Blackberry and Android devices, all native menus are removed. It is recommended that this method be called from customAfterShowScreen.</p> <hr/>
clearCache()	Allows the user to clear the contents of the on-device request result cache for the current workflow.

Method	Description
<code>clearCacheItem(cacheKey)</code>	<p>Allows the user to clear an item from the contents of the on-device request result cache for the current Mobile Workflow form.</p> <p><code>cacheKey</code> – the key for the item to be removed. See the <code>Workflow.js</code> file for details on how to construct this.</p> <hr/> <p>Note: This function is for the "Online Request" cache only and does not apply to cached credentials.</p> <hr/>
<code>logToWorkflow(message, level, notifyUser)</code>	<p>Allows the user to log a message to the device trace log, which can be remotely retrieved from the server.</p> <ul style="list-style-type: none"> • <code>message</code> – message to log. • <code>level</code> – level at which to log (ERROR, WARN, INFO, or DEBUG). • <code>notifyUser</code> – if true, an alert dialog is shown before logging commences.
<code>showCertificatePicker()</code>	<p>Opens a form on the device that allows the user to specify the credentials through the use of certificate-based authentication.</p>
<code>showUrlInBrowser(url)</code>	<p>Open the supplied URL in the browser.</p> <p>Use this method only for opening external HTML files. To open a local HTML file, you must add an <code>HtmlView</code> control to the screen and use something similar to the following for the control's value:</p> <pre><iframe src="./local_file_name.html" width="100%" height="80%"> <p>Your browser does not support iframes.</p> </iframe></pre>
<code>showAttachmentContents(contents, mimeType, fileName)</code>	<p>Shows the given file contents in a content-appropriate way. The content type is supplied by either the MIME type or the file name, at least one of which must be supplied. The content itself should be presented as a Base64-encoded string.</p> <ul style="list-style-type: none"> • <code>contents</code> – the Base64-encoded version of the binary content of the attachment to show. • <code>mimeType</code> – (optional) the MIME type of the file. • <code>fileName</code> – (optional) the name of the file.

Method	Description
<code>showAttachmentFromCache(uniqueKey, mimeType, fileName)</code>	<p>Shows the given file contents in a content-appropriate way. The content type is supplied by either the MIME type or the file name, at least one of which must be supplied. The content itself is a unique key supplied earlier to a call to <code>doOnlineRequest</code>.</p> <ul style="list-style-type: none"> • <code>uniqueKey</code> – the unique key for the attachment. • (optional) <code>mimeType</code> – the MIME type of the file. • (optional) <code>fileName</code> – the name of the file.
<code>showLocalAttachment(key)</code>	<p>Shows a local attachment.</p> <p><code>key</code> – the key.</p>
<code>doOnlineRequest(screenKey, requestAction, timeout, cacheTimeout, errorMessage, errorCallback, workflowMessageToSend, cacheKey)</code>	<p>Allows the user to cause an operation/object query to be invoked.</p> <ul style="list-style-type: none"> • <code>screenKey</code> – the specified screen on which the submit is occurring. • <code>requestAction</code> – the specified action for the submit. • <code>timeout</code> – specifies the time, in seconds, to wait for a response. • <code>cacheTimeout</code> – specifies the time, in seconds, since the last invocation with the same input parameter values, to use the same response as previously retrieved without making a new call to the server. • <code>errorMessage</code> – specifies the string to show if an online request fails. • <code>errorCallback</code> – name of the function to be called if an online request fails. • <code>workflowMessageToSend</code> – Mobile Workflow message that is sent as the input in an online request. • <code>cacheKey</code> – string used as the key for this request in the on-device request result cache.

Method	Description
doAttachmentDownload(screenKey, requestAction, workflowMessageToSend, attachmentKey, requestGUID, downloadCompleteCallback)	<ul style="list-style-type: none"> screenKey – the specified screen on which the submit is occurring. requestAction – the specified action for the submit. workflowMessageToSend – the Mobile Workflow message that is sent as the input in an online request. attachmentKey – the specified key of the result is not returned in the workflow message but is, instead, stored on the device for later access. requestGUID – if specified, represents a unique key that can be used to store/access the cached key value from the request results. Must be specified if keyToCache is specified. Used to support attachments. downloadCompleteCallback – if specified, is a function that is invoked when the cached value has been downloaded to the device and is ready to be accessed. Must be specified if keyToCache is specified. Used to support attachments.
doSubmitWorkflow(screenKey, requestAction, submitMessage, resubmitMessage)	<p>Allows the user to cause an operation/object query to be invoked. Will close the workflow application when finished.</p> <ul style="list-style-type: none"> screenKey – the specified screen on which the submit is occurring. requestAction – the specified action for the submit. submitMessage – specifies the string to show if an asynchronous request is successfully submitted. resubmitMessage – specifies the string to show if an asynchronous request is not submitted because the workflow has already been processed.
showAlertDialog(message, title)	Brings up a message dialog with the specified message, or, optionally, on iOS only, the specified title.
showConfirmDialog(message, title)	Shows a confirmation dialog.

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 to Sybase Mobile Workflow</b><br>
<br>Your activation was successful, the newly created Workflow
requests will automatically be pushed to you.<br>
<br>For more information contact your administrator or visit us
at:<br>
<br>
<a href="javascript:showUrlInBrowser('http://www.sybase.com/
unwiredenterprise')">The Unwired Enterprise</a>
</body>
</html>
```

View an attachment such as an image, a Word document, a .pdf file, and so on as part of the Mobile Workflow package. This example uses an image file.

1. Generate the Mobile Workflow package and its files.
2. In WorkSpace Navigator, go to the location where the generated Mobile Workflow files are located and add an images folder under the html folder, for example, Generated Workflow\<Workflow_name>\html\images.
3. Copy an image to the images folder.
4. In the Mobile Workflow Forms editor, add a menu item to the Mobile Workflow.
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 Mobile Workflow package to Unwired Server.

Workflow Message Data Functions

Access the mobile workflow application message data functions.

A mobile workflow application 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 `getWorkflowMessage()`. 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.

Method	Description
<code>getCurrentMessageValueCollection(listKey)</code>	Recommended for listviews so that the user gets the appropriate part, or values, of the message, for the current screen. Each row in the listview corresponds to a Values section. So, for example, if the user is on the details screen and <code>getCurrentMessageValueCollection</code> is called, the portion that matches the listview row the user clicked appears. If <code>getWorkflowMessage</code> is called, the entire message appears.
<code>getWorkflowMessage()</code>	Allows the user access to the Mobile Workflow message.
<code>getMessageValueCollectionForOnlineRequest(screenKey, requestAction, keys, keyTypes)</code>	Gets the message value collection to be sent in an online request.

A typical mobile workflow 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_attribKey"
        .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_paramKey")
        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

```

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;
}
```

Workflow Validation Functions

Workflow validation methods allow you to access the Mobile Workflow application validation functions.

The Mobile Workflow validation methods are synchronous.

Method	Description
<code>setValidationText(helpElement, helpMessage)</code>	Sets the text for the specified help element.
<code>validateRegularExpression(value, regularExp, userSuppliedMsg, helpElement)</code>	<p>Returns true if the specified value matches the specified regular expression.</p> <ul style="list-style-type: none"> • value – the specified value. • regularExp – the specified regular expression. • (Optional) userSuppliedMsg –the message to use if the specified value is invalid. • (Optional) helpElement – the help element.
<code>validateDate(value, required, minValue, maxValue, userSuppliedMsg, helpElement)</code>	<p>Returns 0 if the specified value represents a valid date, given the constraints specified; otherwise, returns an error code.</p> <ul style="list-style-type: none"> • value – the specified value. • (Optional) required – true if the value must be specified, otherwise, false. • (Optional)minValue – the minimum allowable value. • (Optional) maxValue – the maximum allowable value. • (Optional) userSuppliedMsg – the message to use if the specified value is invalid. • (Optional) helpElement – the help element.

Method	Description
<code>validateDateTime(value, required, minValue, maxValue, userSuppliedMsg, helpElement)</code>	<p>Returns 0 if the specified value represents a valid datetime, given the constraints specified; otherwise, returns an error code.</p> <ul style="list-style-type: none"> • <code>value</code> – the specified value. • (Optional) <code>required</code> – true if the value must be specified, otherwise, false. • (Optional) <code>minValue</code> – the minimum allowable value. • (Optional) <code>maxValue</code> – the maximum allowable value. • (Optional) <code>userSuppliedMsg</code> – the message to use if the specified value is invalid. • (Optional) <code>helpElement</code> – the help element.
<code>validateTime(value, required, minValue, maxValue, userSuppliedMsg, helpElement)</code>	<p>Returns 0 if the specified value represents a valid time, given the constraints specified; otherwise, returns an error code.</p> <ul style="list-style-type: none"> • <code>value</code> – the specified value. • (Optional) <code>required</code> – true if the value must be specified, otherwise, false. • (Optional) <code>minValue</code> – the minimum allowable value. • (Optional) <code>maxValue</code> – the maximum allowable value. • (Optional) <code>userSuppliedMsg</code> – the message to use if the specified value is invalid. • (Optional) <code>helpElement</code> – the help element.

Method	Description
validateNumber(value, required, minValue, maxValue, numOfDecimals, maxLength, userSuppliedMsg, helpElement)	<p>Returns 0 if the specified value represents a valid number, given the constraints specified; otherwise, returns an error code.</p> <ul style="list-style-type: none"> • value – the specified value. • (Optional) required – true if the value must be specified, otherwise, false. • (Optional) minValue – the minimum allowable value. • (Optional) maxValue – the maximum allowable value. • (Optional) numOfDecimals – the maximum allowable number of digits after the decimal place. • (Optional) maxLength – the maximum number of characters. • (Optional) userSuppliedMsg – the message to use if the specified value is invalid. • (Optional) helpElement – the help element.
validateText(value, required, maxLength, userSuppliedMsg, helpElement)	<p>Returns 0 if the specified value represents a valid text, given the constraints specified; otherwise, returns an error code.</p> <ul style="list-style-type: none"> • value – the specified value. • (Optional) required – true if the value must be specified, otherwise, false. • (Optional) maxLength – the maximum number of characters. • (Optional) userSuppliedMsg – the message to use if the specified value is invalid. • (Optional) helpElement – the help element.

Method	Description
<code>validateEmail(value, helpElement)</code>	<p>Returns 0 if the specified value represents a valid email; otherwise, returns an error code.</p> <ul style="list-style-type: none"> • <code>value</code>– the specified value. • (Optional) <code>helpElement</code> – the help element.
<code>validateControl(screenKey, controlKey, control)</code>	<p>Validates the specified control.</p> <ul style="list-style-type: none"> • <code>screenKey</code> – the screen the control is on. • <code>controlKey</code> – the control's key. • <code>control</code> – the HTML element for the control.
<code>validateScreen(screenName, values, keysToValidate)</code>	<p>Validates the specified screen.</p> <ul style="list-style-type: none"> • <code>screenName</code>– the specified screen. • <code>values</code> – the current message value collection. • <code>keysToValidate</code> –
<code>validateAllScreens()</code>	Validates all open screens.

Credential Functions

Access the Mobile Workflow credential functions.

Method	Description
<code>saveLoginCredentials(userName, password)</code>	<p>Saves login credentials to the credential cache.</p> <ul style="list-style-type: none"> • <code>userName</code> – the user name to save • <code>password</code> – the password to save
<code>saveLoginCertificate(certificate)</code>	<p>Saves login credentials from a certificate. The common name is used for the user name, and the signed certificate is used for the password.</p>

Callbacks.js File

This file contains callback functions.

Callback functions are typically used for event handlers that are asynchronous.

Function	Description
<code>CallbackSet()</code>	Invoked to instantiate a new <code>CallbackSet</code> object.
<code>CallbackSet.prototype.registerCallback()</code>	Invoked asynchronously to handle callbacks from the container.
<code>CallbackSet.prototype.callbackHandler()</code>	Invoked asynchronously to handle callbacks from the container.

Camera.js

Using the `Camera.js` file, you can take a picture from the camera, or pick one from the photolibrary and use the picture in the mobile workflow.

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 mobile workflow application is restored. If the device does not have a camera application, the function reports that it is not supported.

Function	Description
getPicture(onGetPictureError, onGetPictureSuccess, options)	<ul style="list-style-type: none"> onGetPictureError – If an error occurs with the picture chooser or the device camera, an appropriate error code is returned in the onGetPictureError function. onGetPictureSuccess (fileName, response) – the return value is sent to this function <hr/> <p>Note: The function defined must use the format onGetPictureSuccess(fileName, response). fileName is required.</p> <hr/> <ul style="list-style-type: none"> options – picture options: PictureOption.SourceType <ul style="list-style-type: none"> CAMERA – specifies the built-in camera as the image source where image content is not persisted by the device PHOTOLIBRARY – specifies the photo library as the image source where image content is already persisted on the device BOTH – Specifies the built-in camera as the image source where image content is persisted by the device <hr/> <p>Note: The BOTH source type is not supported on Android or Windows Mobile.</p> <hr/> <ul style="list-style-type: none"> PictureOption.DestinationType <ul style="list-style-type: none"> imageUri – returns uniform reference identifier for the image imageData – Deprecated. It is recommended that you use imageUri.

Function	Description
onGetPictureError(errorcode)	<p>This is a user-defined function. Error codes include:</p> <ul style="list-style-type: none">• PictureError.NO_ERROR = 0;• PictureError.NOT_SUPPORTED = -1; – getPicture() not implemented, camera not present,• PictureError.IN_PROGRESS = -2; – getPicture() has already been requested but has not yet completed.• PictureError.USER_REJECT = -3; – the user has canceled the request.• PictureError.BAD_OPTIONS = -4; – supplied options were not recognized.• PictureError.TOO_LARGE = -5; – the returned image size was too large to be handled by JavaScript• PictureError.UNKNOWN = -6; – an unknown error occurred.

Function	Description
onGetPictureSuccess(fileName, response)	<p>This is a user-defined function. The return value is sent to the onGetPictureSuccess function, in one of the following formats, depending on the GetPicture options you specify. You can take this value and set it as a value into the MessageValueCollection function.</p> <ul style="list-style-type: none"> File name – file name of the image response – the response will be either a Base64-encoded JPG string or a URI. <p>In the code, onGetPictureSuccess(fileName, response) will be used as onGetPictureSuccess(fileName, imageURI) or onGetPictureSuccess(fileName, imageData). The parameter should coincide with the PictureOption.DestinationType. There are two conditions:</p> <ul style="list-style-type: none"> destinationType: PictureOption.DestinationType.IMAGE_URI – returns a uniform reference identifier for the image onGetPictureSuccess(fileName, imageURI) destinationType: PictureOption.DestinationType.IMAGE_DATA – returns Base64-encoded string onGetPictureSuccess(fileName, imageData)

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 Workflow 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.PHOTO_LIBRARY`, 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 Workflow 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 workflow, the image contents come back from the online request, which can result in too large of a workflow message for the container to handle. It is recommended that you use the submit workflow 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

```
getWorkflowMessage().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 this example:

```
var options = { destinationType:
PictureOption.DestinationType.IMAGE_URI, sourceType:
PictureOption.SourceType.PHOTO_LIBRARY };

getPicture( onGetPictureError, onGetPictureSuccess, options );

function onGetPictureSuccess(fileName, imageUri){
    // Set file for upload
    var fileDataKey = "Picture_create_fileData_paramKey";

    var messageValue =
getWorkflowMessage().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);
        getWorkflowMessage().getValues().add(fileDataKey,
messageValue);
    }

    getWorkflowMessage().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).
    getWorkflowMessage().getValues().add(fileKey, messageValue);

    getWorkflowMessage().setHasFileMessageValue(true); //
Explicitly tell Workflow about image
}
var options = { destinationType:
PictureOption.DestinationType.IMAGE_URI, sourceType:
PictureOption.SourceType.CAMERA};
getPicture( onGetPictureError, onGetPictureSuccess, options );

```

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 mobile workflow, ensure that the attribute set in the mobile business object is a **BigBinary** datatype, rather than `Binary`.

Certificate.js

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.

Function	Description
<code>CertificateStore.parseCertDate(value)</code>	Parses a certificate date.
<code>CertificateStore.createCert(value)</code>	Creates a certificate from the specified JSON string value.
<code>CertificateStore.prototype.certificateLabels(filterSubject, filterIssuer)</code>	Returns a list of all the certificate labels in this store (can be empty). Each certificate in this store has a unique label.
<code>CertificateStore.getDefault()</code>	Returns a certificate without the <code>signedCertificate</code> part set.
<code>CertificateStore.prototype.getPublicCertificate(label)</code>	<p>Returns a certificate without the <code>signedCertificate</code> part set.</p> <p>Supported platforms:</p> <ul style="list-style-type: none"> • Windows Mobile Professional • BlackBerry

Function	Description
<code>CertificateStore.prototype.getSignedCertificate(label, password)</code>	<p>Returns the certificate with the specified label, decrypting it if necessary using the specified password; or returns null if the certificate is encrypted and the password is incorrect.</p> <p>Supported platforms:</p> <ul style="list-style-type: none"> • Windows Mobile Professional • BlackBerry
<code>CertificateStore.prototype.listAvailableCertificatesFromFileSystem(folder, fileExtension)</code>	<p>Returns a list of full path names for the certificate files found in the file system for import.</p> <p>Android platforms only.</p>
<code>CertificateStore.prototype.getSignedCertificateFromFile(filePath, password)</code>	<p>Gets a certificate from a file.</p> <p>Android platforms only.</p>
<code>CertificateStore.prototype.getSignedCertificateFromServer(username, serverPassword, certPassword)</code>	<p>Gets a certificate from the server.</p> <p>iOS platforms only.</p>
<code>getSignedCertificateFromAfaria(commonName, challengeCode)</code>	<p>To retrieve an x509 certificate from Afaria, you must get a <code>CertificateStore</code> and then call getSignedCertificateFromAfaria. If Afaria is installed and configured on the device, this gets the Afaria seeding file from the Afaria server. If the seeding file is retrieved from the Afaria server, the user is prompted to update user specific information in the Settings screen. The parameters are:</p> <ul style="list-style-type: none"> • <code>commonName</code> – this parameter is required. • <code>challengeCode</code> – this parameter is optional, depending on how the Afaria server is configured. <p>For example:</p> <pre>var certStore = CertificateStore.getDefault(); afariaCert = certStore.getSignedCertificateFromAfaria(commonName, challengeCode);</pre>

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 Mobile Workflow package files, the `Custom.js` file is generated.

In subsequent file generations for the same Mobile Workflow 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 **custom Before** method is invoked and a **customAfter** method is invoked. The Before method returns a boolean. If it returns true, it continues to execute the default behaviour, 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:

Method	Description
<code>customBeforeWorkflowLoad()</code>	Invoked when the application is first launched, before any data is loaded, or screens are opened. Because workflow settings are not yet initialized at this point, you cannot call any <code>SharedStorage</code> functions here.
<code>customAfterWorkflowLoad()</code>	Invoked when the application is first launched, after data is loaded and screens are opened.
<code>customBeforeSubmit(screenKey, requestAction, workflowMessageToSend)</code>	Invoked before an operation or object query is about to be called as the result of the user clicking a <code>Submit</code> menuitem. You can set this to return false to prevent the default behaviour from occurring.
<code>customAfterSubmit(screenKey, requestAction)</code>	Invoked after an operation or object query is called as the result of the user clicking a <code>Submit</code> menuitem.
<code>customBeforeNavigateForward(screenKey, destScreenKey)</code>	Invoked when another screen is about to be opened. Set to false to prevent the screen from being opened.
<code>customBeforeNavigateBackward(screenKey, isCancelled)</code>	Invoked when another screen is about to be opened. You can set to false to prevent the screen from being opened.

Method	Description
<code>customAfterNavigateForward(screenKey, dest-ScreenKey)</code>	Invoked after another screen has been opened.
<code>customAfterNavigateBackward(screenKey, is-Cancelled)</code>	Invoked after a screen has been closed.
<code>customBeforeShowScreen(screenToShow, screenToHide)</code>	Invoked when a screen is about to be shown. User can return false to prevent the screen from being shown.
<code>customAfterShowScreen(screenToShow, screen-ToHide)</code>	Invoked after a screen is shown.
<code>customValidateScreen(screenKey, values)</code>	Invoked when the contents of a screen need to be validated. User can return false to indicate that the contents of the screen are not valid.
<code>customBeforeMenuItemClick(screen, menuItem)</code>	Invoked after a menuitem has been clicked. User can return false to prevent the default behavior (which might open a new screen, or perform a submit, and so on) from occurring.
<code>customAfterMenuItemClick(screen, menuItem)</code>	Invoked after a menuitem has been clicked and the default behavior has occurred.
<code>customBeforeSave(screen)</code>	Invoked before a screen's contents are persisted to the Mobile Workflow message. User can return false to prevent the default behaviour from occurring.
<code>customAfterSave(screen)</code>	Invoked after a screen's contents are persisted to the Mobile Workflow message through the default logic.

Method	Description
customConditionalNavigation(currentScreenKey, actionName, defaultNextScreen, conditionName, workflowMessage)	<p>For online request menu items and custom actions, this method is invoked to evaluate the given condition after a given action is executed. If the screen associated with the condition should be navigated to, the condition is true.</p> <p>For server-initiated starting points, the judgement condition is <code>if ((currentScreenKey === SERVERINITIATEDFLAG)</code>.</p> <p>This method is different from the others in two of its attributes:</p> <ul style="list-style-type: none"> • It returns true or false – the custom code used to implement this method can peer into the workflow message and execute logic. This routine generally does not modify the HTML or anything else. • There is no before or after behavior – this function is executed after the workflow message is received from the server, but before the screen is opened. Therefore, this is executed before the "customBefore-ShowScreen()" because this function is used to help decide what screen to show next. <p>Conditions set by the user in the designer are executed serially, and the first one that returns true determines what the start screen is. As soon as a true condition is found, evaluation stops and the screen is executed.</p>
customBeforeReportErrorFromNative(errorString)	Invoked when a native error is reported on. Return false to prevent the default behavior from executing (bringing up an alert dialog)
customAfterReportErrorFromNative(errorString)	Invoked after a native error is reported.
customAfterDataReceived(incomingWorkflowMessage)	Invoked after data is received from the server. This allows you to view and manipulate the data.
customGetPictureURIFromCamera()	Use this method to get a picture URI from the camera for submission to the workflow message.

Method	Description
<code>customGetPictureURIFromLibrary()</code>	Use this method to get a picture URI from the photo library for submission to the workflow message.
<code>customGetPictureURISuccess(fileName, image-URI)</code>	Handles success from one of the URI calls.
<code>customGetPictureDataFromCamera()</code>	Use this method to get picture data from the camera for submission to the workflow message. Note: The picture data length should not be too long.
<code>customGetPictureDataFromLibrary()</code>	Use this method to get picture data from the photo library for submission to the workflow message. Note: The picture data length should not be too long.
<code>customGetPictureDataSuccess(fileName, imageData)</code>	Handles success from one of the "Data" calls.
<code>customGetPictureError(result)</code>	Invoked after an error is reported.
<code>getMimeType(fileName)</code>	A helper method that allows you to include the MIME data type in the workflow message.

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 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_attriKey").getV
alue();
            var orderId =
currItem.getData("PersonalWorkQueue_orderId_attriKey").getValue();
            var operationNumber =
currItem.getData("PersonalWorkQueue_operationNumber_attriKey").get
Value();
            var description =
currItem.getData("PersonalWorkQueue_description_attriKey").getValu
e();
            try {
                var promDate =
currItem.getData("PersonalWorkQueue_datePromised_attriKey").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.getKey() + '"
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.

Function	Description
<code>Resources(currentLocaleName)</code>	Creates a new resources object with the specified locale as the default locale.
<code>resources.prototype.hasLocale(localeName)</code>	Returns true if the locale supplied is included in the Mobile Workflow application; otherwise, returns false.
<code>resources.prototype.getStringFromLocale(key, localeName)</code>	Returns the localized string for the supplied key for the current locale.
<code>resources.prototype.getString(key)</code>	Returns the localized string for the supplied key for the specified locale.

ExternalResource.js

These functions allow you to access resources on external HTTP servers.

Function	Description
getExternalResource(url, options)	<p>Makes a request to access resources on an external HTTP server.</p> <ul style="list-style-type: none"> • options – a set of key/value pairs that configure the underlying request. Supported options are: <ul style="list-style-type: none"> • method – one of GET, PUT, DELETE, HEAD, OPTIONS, or POST. The default is GET. • HTTP and HTTPS 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. • complete – callback for when this method completes. <p>Note: An X.509 certificate blob (used in SSO) cannot be used as a client certificate for HTTPS communication. In other words, you can use only client certificates from trusted certificate authorities, which means that self-signed certificates do not work.</p>

Function	Description
complete(resultXHR)	<p>The complete(resultXHR) callback is sent when <code>getExternalResource</code> finishes. Its input is a <code>resultXHR</code> structure which closely mirrors the JavaScript <code>XmlHttpRequest</code> object.</p> <p>The fields/methods available on <code>resultXHR</code> are:</p> <ul style="list-style-type: none"> • <code>status</code> • <code>statusText</code> • <code>responsesText</code> • <code>getReponseHeader(key)</code> • <code>getAllResponsesHeaders()</code> <p>These fields and methods are not supported for resultXHR:</p> <ul style="list-style-type: none"> • <code>open()</code> • <code>setRequestHeader()</code> • <code>responseXML</code> • <code>send()</code> • <code>abort()</code>

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

Access the storage functions, which allow you to specify a cache that stores results from online requests.

These functions give you the ability to:

- Name the cached result sets
- Enumerate the cached result sets
- Read, delete, and modify cached contents individually for each cached result set

Cached result sets must be stored as strings (before deserialization to an `xmlWorkflowMessage` structure).

Method	Description
<code>SUPStorage(store)</code>	Provides encrypted storage of name value pairs. Results from online requests are one example. Strings stored in <code>SUPStorage</code> are encrypted and persisted to survive multiple invocations of the mobile workflow application.
<code>SharedStorage()</code>	Used to construct a new shared SUP storage. You can use the returned value to access the shared storage data with the existing <code>SUPStorage</code> interface, however, the operation only affects the items belonging to the specified shared storage key.
<code>getSharedStorageKey()</code>	Used to return the shared storage key defined for the mobile workflow application during design time. An empty string is returned if the shared storage function is disabled.
<code>isSharedStorageEnabled()</code>	Indicates whether the shared storage is enabled for the current workflow application.

Method	Description
<code>SUPStorage.prototype.length()</code>	Gets the number of available keys in this object. Retrieve the keys themselves by using key() . length – returns the number of key/value pairs currently present in the list
<code>SUPStorage.prototype.key(index)</code>	Returns the key at the supplied index. Keys are guaranteed to remain at the same index until a modification is made.
<code>SUPStorage.prototype.getItem(key)</code>	Retrieves the value associated with the specified key. Returns a copy of the current value associated with the given key. If the given key does not exist in the list, null is returned.
<code>SUPStorage.prototype.setItem(key, value)</code>	Sets the value associated with a specified key. This replaces the key's previous value, if any. If the given key does not exist in the list, a new key value is added to the list with the given key and with its value set to a copy of value .
<code>SUPStorage.prototype.clear()</code>	Removes all key/value pairs from this object.
<code>SUPStorageException(code, message)</code>	Constructs a new SUPStorageException object.

Calls to these methods do not trigger events.

- constructor

```
// The following script creates a 2 local storage instances with
their own domain
var store1 = new SUPStorage("mydomain");
var store2 = new SUPStorage("myotherdomain");
```

- length

```
// The following script displays the current number of elements in
the storage
var store = new SUPStorage();
alert(store.length());
```

- key(index)

```
// The following script displays the value at the provided index in
the storage
var store = new SUPStorage();
alert(store.key(2));
```

- getItem(key)

```
// The following script displays the value for the provided key
```

```
var store = new SUPStorage();  
alert(store.getItem("mykey"));
```

- setItem(key, value)

```
// The following script sets a key/value pair  
var store = new SUPStorage();  
store.setItem("mykey", "myvalue");
```

- removeItem(key)

```
// The following script removes a key/value pair  
var store = new SUPStorage();  
store.removeItem("mykey");
```

- clear

```
// The following clears the storage  
var store = new SUPStorage();  
store.clear();
```

SUP Storage

The SUP 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 Mobile Workflow package. Values are also strings and values can be duplicated in the same Mobile Workflow package. Keys and values can contain multi-byte characters.

SUP Storage can span multiple screens in the Mobile Workflow application, 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 SUP Storage 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 PersistentStore
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/or the SQLite database size limit.

Limitations

- The amount of data that can be retrieved using the SUPStorage API, and returned to the JavaScript space, is limited to the JavaScript size limitation as established for each platform. See the topic *AttachmentViewer and Image Limitations*.
- 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 SUP storage is tied to a Mobile Workflow package. When the Mobile Workflow package is uninstalled, the corresponding SUP storage for the Mobile Workflow package is removed immediately.
- Items stored using the SUPStorage API are persisted, and therefore, survive soft device resets.
- SUP Storage persists through invocations of the Mobile Workflow application.
- The SUPStorage API does not restrict reading or writing of the storage data from different domains. For example, if a Mobile Workflow application 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 Mobile Workflow applications with a shared storage key assigned share the storage with other Mobile Workflows that have the same storage key assigned.

- When the last Mobile Workflow application 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 *AttachmentViewer 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 Workflow 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 Workflow application.

Function	Description
<code>getCurrentLocale()</code>	Returns the current locale for the device.
<code>getLocalizedDateTime(date)</code>	Returns the specified Date as a locale-specific display as a datetime using native functionality.
<code>getLocalizedDate(date)</code>	Returns the specified Date as a locale-specific display as a date using native functionality.
<code>getLocalizedTime(date)</code>	Returns the specified Date as a locale-specific display as a time using native functionality.
<code>convertUtcToLocalTime(date)</code>	Converts the specified Date to the device's local time.
<code>convertLocalTimeToUtc(date)</code>	Converts the specified Date to UTC time.
<code>getOffsetFromUTC(date)</code>	Given the specified Date, queries the device's OS to determine the total offset (difference) between the given "local" time and UTC, including any daylight savings time (DST) offsets if applicable. Returns a signed integer representing this GMT offset in minutes. For example, if the device was in the London timezone (GMT+1) and it is DST is currently in effect, the function would return "120"—60 minutes normal offset plus 60 minutes for its DST offset.
<code>isDstActiveAtGivenTime(date)</code>	Given the specified Date, queries the device's OS to determine if DST rules are in effect for the current timezone at the time specified in the Date object.
<code>getDstOffsetAtGivenTimeInMinutes(date)</code>	Given the specified Date, queries the device's OS to determine the DST offset for the current time-zone at the time specified in the Date object.

Function	Description
<code>getTimezoneId()</code>	Returns the current timezone's standard name.
<code>getUsesDST()</code>	Determines whether or not the device's current timezone resides in a timezone that practices daylight savings.

WorkflowMessage.js

Use these functions to access Workflow Message resources.

Function	Description
<code>MessageValue()</code>	Constructs a new MessageValue object.
<code>MessageValue.prototype.getKey()</code>	Returns the key of the given MessageValue object
<code>MessageValue.prototype.setKey(key)</code>	Sets the key of the given MessageValue object
<code>MessageValue.prototype.getValue()</code>	Returns the value of the given MessageValue object
<code>MessageValue.prototype.setValue(value)</code>	Sets the value of the given MessageValue object
<code>MessageValue.prototype.getType()</code>	Returns the type of the given MessageValue object
<code>MessageValue.prototype.setType(type)</code>	Sets the type of the given MessageValue object
<code>MessageValueCollection()</code>	Constructs a new MessageValueCollection object
<code>MessageValueCollection.prototype.getKey()</code>	Returns the key of the given MessageValueCollection object
<code>MessageValueCollection.prototype.setKey(key)</code>	Sets the key of the given MessageValueCollection object
<code>MessageValueCollection.prototype.getState()</code>	Returns the state of the given MessageValueCollection object.
<code>MessageValueCollection.prototype.setState(state)</code>	Sets the state of the given MessageValueCollection object

Function	Description
<code>MessageValueCollection.prototype.getParent()</code>	Returns the parent key of the given MessageValueCollection object
<code>MessageValueCollection.prototype.setParent(parent)</code>	Sets the parent key of the given MessageValueCollection object.
<code>MessageValueCollection.prototype.getParentValue()</code>	Returns the parent object of the given MessageValueCollection object
<code>MessageValueCollection.prototype.setParentValue(parentValue)</code>	Sets the parent object of the given MessageValueCollection object. Does not change the actual parenting.
<code>MessageValueCollection.prototype.add(key, value)</code>	Adds a new value to the given MessageValueCollection
<code>MessageValueCollection.prototype.clear()</code>	Removes all value from the given MessageValueCollection
<code>MessageValueCollection.prototype.getData(key)</code>	Returns the value corresponding to the specified key for the given MessageValueCollection
<code>MessageValueCollection.prototype.remove(key)</code>	Removes the value corresponding to the specified key for the given MessageValueCollection
<code>MessageValueCollection.prototype.getCount()</code>	Returns the number of values in the given MessageValueCollection
<code>MessageValueCollection.prototype.getKeys()</code>	Returns an array of the keys in the given MessageValueCollection.
<code>MessageValueCollection.prototype.getValues()</code>	Returns an array of the values in the given MessageValueCollection.
<code>WorkflowMessage(message)</code>	Constructs a new WorkflowMessage object with the specified contents (represented as a string).
<code>WorkflowMessage.prototype.getHeader()</code>	Returns the header of the given WorkflowMessage.
<code>WorkflowMessage.prototype.setHeader(header)</code>	Sets the header of the given WorkflowMessage.
<code>WorkflowMessage.prototype.getRequestAction()</code>	Returns the request action of the given WorkflowMessage.

Function	Description
WorkflowMessage.prototype.setRequestAction(requestAction)	Sets the request action of the given WorkflowMessage.
WorkflowMessage.prototype.getValues()	Gets the values in the given WorkflowMessage.
WorkflowMessage.prototype.getWorkflowScreen()	Returns the workflow screen of the given WorkflowMessage.
WorkflowMessage.prototype.setWorkflowScreen(workflowScreen)	Sets the workflow screen of the given WorkflowMessage.
WorkflowMessage.prototype.createFromString(messageAsString)	Updates the contents of the given WorkflowMessage object from the given string.
WorkflowMessage.prototype.serializeToString()	Returns a string representation of the given WorkflowMessage.
WorkflowMessage.prototype.updateValues(sourceValues, listViewValuesKey)	Updates the values of the given WorkflowMessage.

Using Third-Party JavaScript Files

To include your own files in the container application, copy them into the appropriate place in the Generated Workflows folder.

To load external JavaScript and CSS files dynamically, copy the relevant third-party JavaScript and CSS files to the Generated Workflow

\<Workflow_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.

These files will be included in the Mobile Workflow manifest.xml and .zip files automatically when the Mobile Workflow package is re-generated.

Repackaging Mobile Workflow Package Files

After modifying the Custom.js file, you must redeploy the Mobile Workflow package to Unwired Server.

1. Save and close the modified files after adding your custom code.
2. In Workspace Navigator, right-click the <mobile_workflow_name>.xbw file and select **Generate Mobile Workflow Package**.
3. In the Mobile Workflow Package Generation wizard, select **Deploy to an Unwired Server**, and select the Unwired Server connection profile.

4. In Deploy Mode, select either:

- New – generates and deploys the mobile workflow package and its files for the first time.
- Update – updates any pre-existing Mobile Workflow package in-place, preserving associated assignments.
- Replace – removes any pre-existing Mobile Workflow package and notifications before deploying.

5. Click **Finish**.

Common Customizations

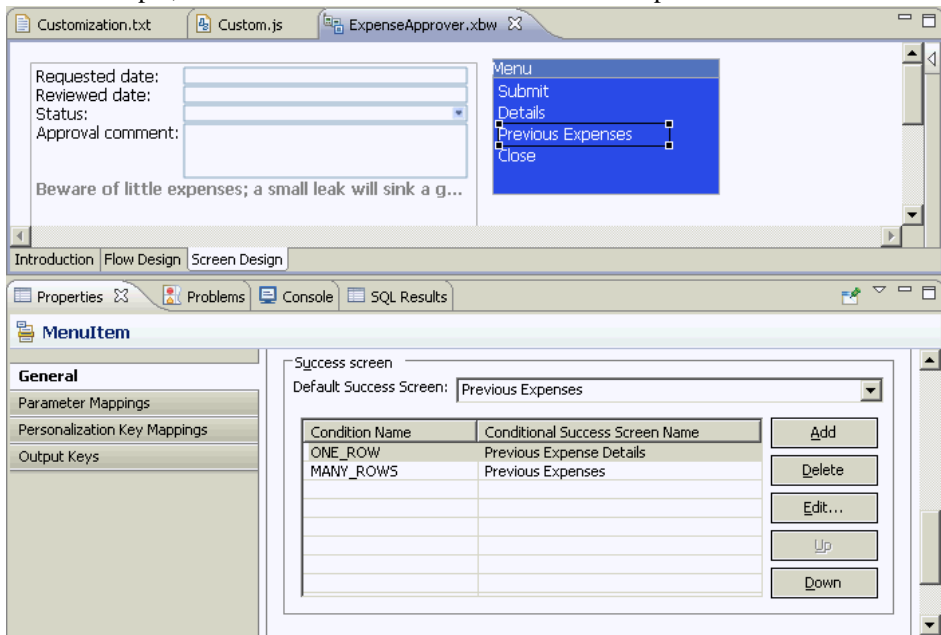
Implementing Conditional Navigation

Conditional navigation allows you to implement a custom function that allows you to override navigation behavior between screens.

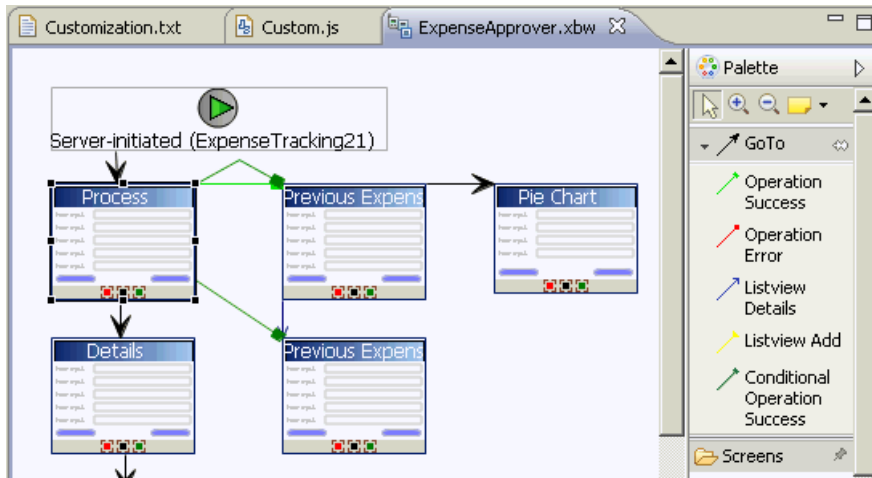
This procedure gives an example of how you can use conditional navigation to skip a screen.

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 Mobile Workflow Package Generation wizard to re-generate the Mobile Workflow Package with a new workflow_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 grey to differentiate it from the default GoTo line.

The screenshot shows the Sybase Unwired Platform Flow Design page. The main canvas displays a workflow diagram with a 'Server-initiated (ATable)' starting point. A default black line connects it to the 'ATableDetail' screen. Two grey lines represent conditional start paths: one to 'Screen Start One' and another to 'Screen Start Two'. From 'ATableDetail', black lines lead to 'Success Detail 1', 'Success Detail 2', 'ATable_update', and 'ATable_delete'. A red line leads from 'ATableDetail' to 'ErrorList', which then leads to 'ErrorDetail'. The right-hand 'Palette' contains categories like 'GoTo', 'Screens', 'Starting Points', and 'Activate'. The bottom 'Properties' view is open for the 'Notification [Server-initiated (ATable)]'.

Notification [Server-initiated (ATable)]

General

Keys

Parameter Mapping

Personalization Key Mapping

Start Screen(s)

Default Screen: ATableDetail

Condition Name	Conditional Success Screen Name
Wilma_first_ss1	Screen Start One
Fred_second_screen	Screen Start Two

Buttons: Add, Delete, Edit...

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 Mobile Workflow package.

When you regenerate the Mobile Workflow package, the Workflow.js file is regenerated. The conditional start screen method is shown in the Workflow.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;
}

```

Create a Mobile Workflow Package Manually

While using the Mobile Workflow Forms Editor is the easiest and fastest way to develop and customize mobile workflow applications, it is also possible to develop a mobile workflow package outside of the constraints of the Mobile Workflow Forms Editor.

Developing a mobile workflow application this way allows you to use of a greater variety of application designs, from using different HTML formatting to using different Web application frameworks, and beyond. It should be emphasized, however, that the Mobile Workflow Forms Editor does a lot of the work to seamlessly support multiple platforms, which you must duplicate if you choose not to use it.

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

```
function processWorkflowMessage(incomingWorkflowMessage)
```

The Workflow container needs to call this function when online request processing is complete. The incoming workflow message is an XML-formatted string.

Mobile Workflow URL Parameters

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

An example of how to use these URL parameters can be found in the `onWorkflowLoad()` function in the `Utils.js` file.

URL parameter	Description
loglevel	Current device log level.
screenToShow	Name of the screen which should be displayed.
supusername	Username of the current Workflow (if available).
lang	Current language of the device.
isalreadyprocessed	Indicates whether or not the Workflow 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.

URL parameter	Description
loadtransformdata	Indicates that the JavaScript should request the transform data (contents of the e-mail message) from the Container using the loadtransformdata querytype. For information about the query types, see the topic <i>Calling the Hybrid Web Container</i> .
ignoretransformsreen	Indicates that the JavaScript should ignore the RequestScreen tag in the transform data (contents of the e-mail message). This is set to true when the screen that needs to be shown is either the Activation or Credentials screen.

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 that the Mobile Workflow Forms editor generates.

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 Web Container.

close

Closes the native Web Container (Windows Mobile only).

addMenuItem

Adds a single menu item to the Web Container.

removeallmenuitems

Removes all the menu items from the Web Container.

clearrequestcache

Clears the entire Online Request cache for the current Mobile Workflow application.

clearrequestcacheitem

Clears a single Online Request cache entry for the current Mobile Workflow application.

logtoworkflow

Logs a message to the AMPHostLog.txt (mocalog.txt for iOS) on the device. This log file can be retrieved remotely from Sybase 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 Workflow 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 Web Container.

showlocalattachment

Using third party file viewers, this function displays an attachment that was included as part of the Workflow .zip package, in a separate window.

Note: On iOS, the attachment is shown within the Web Container.

rmi

This function executes an online request to the Unwired 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 Unwired 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 Unwired Server for processing by the server plug-in. This operation occurs asynchronously. If a network 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 Web Container for the transform data (the contents of the e-mail message) for the current Workflow message.

addallmenuitems

Instructs the Web Container to add the supplied list of menu items.

formredirect

Notifies the Container that a screen navigation is occurring, and to update credentials in the credentials cache, if required.

Mobile Workflow Package Files

To build a mobile workflow package manually, you should first familiarize yourself with its contents.

This section describes the contents of the Mobile Workflow 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 Mobile Workflow package contains Web application files.

When developing a Mobile Workflow package manually:

- Include HTML files that follow the same general pattern as the files generated when using the Mobile Workflow Forms editor to generate the Mobile Workflow 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
- Use `WorkflowMessage.js` to view and manipulate the workflow messages

HTML Format

This is the basic 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="onWorkflowLoad();">
    <div id="Screen1KeyScreenDiv" sup_screen_title="Screen1Title"
style="display: none"
sup_menuitems="NativeMenu1Key,NativeMenu1DisplayName,NativeMenu2Key
,NativeMenu2DisplayName" sup_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 Mobile Workflow package .zip file are organized.

This file must reside at the root of the Mobile Workflow .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>

  <TransformPlugin>
    <File>WorkflowClient.dll</File>
    <Class>Sybase.UnwiredPlatform.WorkflowClient.Transformer</Class>
  </TransformPlugin>

  <ResponsePlugin>
    <File>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>
  </ClientWorkflows>
</Manifest>
```

```

    <BlackBerry6>
      <HTMLWorkflow>
        <File>...</File>
        <HtmlFiles>
          <HtmlFile>...</HtmlFile>
          <HtmlFile>...</HtmlFile>
        </HtmlFiles>
      </HTMLWorkflow>
    </BlackBerry6>
  </Android>
  <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>

<MatchRules>
  <SubjectRegExp>...</SubjectRegExp>
  <ToRegExp>...</ToRegExp>
  <FromRegExp>...</FromRegExp>
  <CCRegExp>...</CCRegExp>
  <BodyRegExp>...</BodyRegExp>
</MatchRules>
</Manifest>

```

ModuleName

```
<ModuleName>SampleActivitiesModule</ModuleName>
```

The *ModuleName* defines the name of the Mobile Workflow package.

ModuleVersion

```
<ModuleVersion>2</ModuleVersion>
```

The `ModuleVersion` defines the version of the Mobile Workflow package.

ModuleDesc

```
<ModuleDesc>AMP Sample - Activities Collection</ModuleDesc>
```

The `ModuleDesc` provides a short description of the Mobile Workflow package.

ModuleDisplayName

```
<ModuleDisplayName>Activities Sample</ModuleDisplayName>
```

The name of the Mobile Workflow package that is displayed to the user in the Workflow list on the device for Mobile Workflows that are client-invoked. When the Mobile Workflow 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 Mobile Workflow 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 Mobile Workflow 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 Mobile Workflow can be used without an associated e-mail. 1 = true, 0 = false. If 1 is specified, the Mobile Workflow is shown in the Workflow list on the device and can be used without the context of an e-mail message.

PersistAppDomain

```
<PersistAppDomain>1</PersistAppDomain>
```

States whether this Mobile Workflow 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 Workflow message shows a visual indication in the Inbox after it has been processed (1 = true, 0 = false).

Note: When a Workflow 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 Workflow 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 Workflow messages associated with this Workflow 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 Workflow requires credentials (1 = true, 0 = false). Different Workflows can specify different credentials keys. Workflows with the same credentials key share that set of credentials. In the case of shared credentials, they are requested only once by the Workflow that is launched first.

RequiresActivation

```
<RequiresActivation key="shared_credentials_key">1</
RequiresActivation>
```

Specifies whether a workflow requires activation (1 = true, 0 = false). If set to true, the screen defined in the `ActivationScreen` tag is displayed the very first time the workflow is launched, before the default screen is displayed.

If the Activation Screen contains credentials controls (and the workflow requires credentials), the values are updated to the Credentials Cache automatically, without further prompting, with the specified Credentials Screen.

Different workflows can specify different activation keys. Workflows with the same activation key share their activation status. For example, if Workflow A and Workflow B both specify an activation key of AB (using the `key` attribute on the `RequiresActivation` tag), when Workflow A gets activated, it also activates Workflow B so that when Workflow 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>
```

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 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 workflows 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 workflow will use its own version of that DLL in the Workflow'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>
```

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 Mobile Workflow .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 workflows 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 workflow will use its own version of that DLL in the Workflow'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>
<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 Mobile Workflow 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/myAndroidWorkflow.html" default="Start_Screen">
    <screen key="html/myAndroidWorkflow.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 Mobile Workflow .zip package.

```
<HtmlFile>...</HtmlFile>
```

Indicates that the named file (html/js/API.js, html/myAndroidWorkflow.html) will be used on the specified platform. The referenced file must be present in the Mobile Workflow .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 Mobile Workflow 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 Mobile Workflow package, they have the ability to override the value of the context variable that is specified here.

It is good practice for developers of Mobile Workflows 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 Mobile Workflow, 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, Sybase 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 Sybase Control Center console.

MatchRules

<MatchRules>...</MatchRules>

Describes the collection of match rules that will be used to determine if an e-mail message should be sent to a TransformPlugin server module for processing. When Administrators deploy a Mobile Workflow, they have the ability to Add, Delete and /or 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 an e-mail message.

<ToRegExp>...</ToRegExp> The value to test for against the "To" line of an e-mail message.

<FromRegExp>...</FromRegExp> The value to test for against the "From" line of an e-mail message.

<CCRegExp>...</CCRegExp> The value to test for against the "CC" line of an e-mail message.

<BodyRegExp>...</BodyRegExp> The value to test for against the <Body> text of an e-mail message.

WorkflowClient.dll File

The `WorkflowClient.dll` file is used by the Unwired Platform messaging server to transform the data that is sent to the device as notifications, and to respond to online request and submit workflow actions from the device.

The supplied `WorkflowClient.dll` file loads the metadata in the `WorkflowClient.xml` file to determine how to map the data in the workflow message to and from calls to Mobile Business Object operations and object queries.

The `WorkflowClient.dll` is shared by all Mobile Workflows. It is installed only once, into the `UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\MessagingServer\bin` folder. Your `Manifest.xml` file must refer to the `WorkflowClient.dll` as a shared file. It does not need to be included in the Mobile Workflow .zip file.

WorkflowClient.xml File

The `WorkflowClient.xml` file contains metadata that specifies how to map the data in the workflow 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"
targetsScreen="...">
        <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>
</Triggers>
</Workflow>

```

Globals

```

<Globals> <DefaultScreens activation="Introduction"
credentials="Authentication" /> </Globals>

```

Describes the global information for the Mobile Workflow 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 workflow 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.
sourcescreen	The screen from where the action was invoked.
targetscreens	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.

Attribute	Description
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 workflow 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).
package	The Mobile Workflow package name and version of the MBO, separated by a colon, for example, <code><package_name>:<mbo_version></code> .

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 workflow 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 workflow action is received from the client.

Value

```
<Value sourceType="Key"
workflowKey="Reports_type_id_attrbKey" attrbName="id"
mboType="int" />
```

Describes how to obtain the parameter value or attribute value from the workflow message.

Table 4. Attributes

Attribute	Description
sourceType	<p>The source of the data. Must be one of these types:</p> <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 workflow 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.
relationshipName	The name of the relationship, if any.

Attribute	Description
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 workflow 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 workflow message.

Mapping

```
<Mapping workflowKey="Department_dept_id_attrbKey"
workflowType="number" attrbName="dept_id" mboType="int" />
```

Describes how to fill a key's value in the workflow message from the results of the object query.

Table 6. Attributes

Attribute	Description
workflowKey	The name of the key in the workflow message to fill with the results of the object query.
workflowType	The type of the data in the workflow message. Must be one of these types: <ul style="list-style-type: none"> • text • number • boolean • datetime • date • time • list • choice
attrbName	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 workflow message.

Notification

```
<Notification type="onEmailTriggered" targetscreen="dept"> ...
</Notification>
```

Describes how to formulate the workflow message for the given notification type and which screen to open on the device when that workflow 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 workflow message.

`<Methods> ... </Methods>` Contains the description for one or more object queries to be executed when this online request or submit workflow 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 workflow 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	The name of the key in the workflow message to fill with the value extracted from the server notification.
workflowType	<p>The type of the data in the workflow message. Must be one of these data types:</p> <ul style="list-style-type: none"> • text • number • boolean • datetime • date • time • list • 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.
beforeMatch	A regular expression used to indicate where the value starts.
afterMatch	A regular expression used to indicate where the value ends.
format	If the workflowType is datetime or time, the C# formatting string to be passed to DateTime.ParseExact when converting the value to a datetime.

The Look and Feel XML Files

Each device platform (WindowsMobileProfessional, BlackBerry, BlackBerry6, iOS, and Android) provides a `<File>...</File>` tag, which refers to an .xml file in the Mobile Workflow .zip package.

The contents are similar to this:

```
<?xml version="1.0" encoding="utf-8"?>
<widget>
  <screens src="html/myAndroidWorkflow.html" default="Start_Screen">
    <screen key="html/myAndroidWorkflow.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 Mobile Workflow package is generated using the Mobile Workflow Forms editor, the with the **Optimized for appearance** option selected in Preferences, three look and feel .xml files are generated: workflow.xml, workflow_CustomLookAndFeel.xml, and workflow_jQueryMobileLookAndFeel.xml.

Using Third-party Files

To load external JavaScript and CSS files dynamically when creating a Mobile Workflow 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>
```

Troubleshoot

Use troubleshooting tips to isolate and resolve common issues.

See *Troubleshooting Sybase Unwired Platform* for information about troubleshooting issues for Workflow package and other Sybase Unwired Platform components.

HTTP Error Codes

Unwired Server examines the EIS code received in a server response message and maps it to a logical HTTP error code, if a corresponding error code exists. If no corresponding code exists, the 500 code is assigned to signify either a Sybase Unwired Platform internal error, or an unrecognized EIS error. The EIS code and HTTP error code values are stored in log records.

These tables list recoverable and unrecoverable error codes. All error codes that are not explicitly considered recoverable are considered unrecoverable.

Table 9. Recoverable Error Codes

Error Code	Probable Cause
409	Backend EIS is deadlocked.
503	Backend EIS is down, or the connection is terminated.

Table 10. Unrecoverable Error Codes

Error Code	Probable Cause	Manual Recovery Action
401	Backend EIS credentials wrong.	Change the connection information, or backend user password.
403	User authorization failed on Unwired Server due to role constraints (applicable only for MBS).	N/A
404	Resource (table/Web service/BA-PI) not found on backend EIS.	Restore the EIS configuration.
405	Invalid license for the client (applicable only for MBS).	N/A
412	Backend EIS threw a constraint exception.	Delete the conflicting entry in the EIS.

Error Code	Probable Cause	Manual Recovery Action
500	Sybase Unwired Platform internal error in modifying the CDB cache.	N/A

Error code 401 is not treated as a simple recoverable error. If the `SupThrowCredentialRequestOn401Error` context variable is set to true (the default), error code 401 throws a `CredentialRequestException`, which sends a credential request notification to the user's inbox. You can change this behavior by modifying the value of the `SupThrowCredentialRequestOn401Error` context variable in Sybase Control Center. If `SupThrowCredentialRequestOn401Error` is set to false, error code 401 is treated as a normal recoverable exception.

Recovering from EIS Errors

After sending a JSON request to Unwired Server, if you receive in the response log message an EIS code which is recoverable, the mobile workflow client throws a `TransformRetryException` or `ResponseRetryException`, as is appropriate.

A retry attempt is made after a retry time interval, which is set by default to 15 minutes for recoverable errors, and by default to 3 days for unrecoverable errors. You can configure the retry time interval by setting the `SupRecoverableErrorRetryTimeout` (default: 15 minutes) and `SupUnrecoverableErrorRetryTimeout` context variables through the Sybase Control Center admin console.

Only certain error codes are considered to be recoverable.

Table 11. Recoverable Error Codes

Error Code	Probable Cause
409	Backend EIS is deadlocked.
503	Backend EIS down or the connection is terminated.

Note: If the problem with the EIS is not corrected, the retry process can continue indefinitely. Ensure that you set an appropriate retry time interval.

Other error codes are considered to be non-recoverable. A retry attempt is made after a retry time interval, which is set to three days by default.

Table 12. Non-recoverable Error Codes

Error Code	Probable Cause	Manual Recovery Action
401	Backend EIS credentials wrong.	Change the connection information, or backend user password.
403	User authorization failed on Unwired Server due to role constraints (applicable only for MBS).	N/A
404	Resource (table/webservice/BA-PI) not found on Backend EIS.	Restore the EIS configuration.
405	Invalid license for the client (applicable only for MBS).	N/A
412	Backend EIS threw a constraint exception.	Delete the conflicting entry in the EIS.
500	SUP internal error in modifying the CDB cache.	N/A

Mapping of EIS Codes to Logical HTTP Error Codes

A list of SAP® error codes mapped to HTTP error codes. By default, SAP error codes that are not listed map to HTTP error code 500.

Note: These JCO error codes are not applicable for DOE-based applications.

Table 13. Mapping of SAP Error Codes to HTTP Error Codes

Constant	Description	HTTP Error Code
JCO_ERROR_COMMUNICATION	Exception caused by network problems, such as connection breakdowns, gateway problems, or unavailability of the remote SAP system.	503
JCO_ERROR_LOGON_FAILURE	Authorization failures during login. Usually caused by unknown user name, wrong password, or invalid certificates.	401

Constant	Description	HTTP Error Code
JCO_ERROR_RESOURCE	Indicates that JCO has run out of resources such as connections in a connection pool.	503
JCO_ERROR_STATE_BUSY	The remote SAP system is busy. Try again later.	503

Credentials Are Lost after User Successfully Passes Activation Screen

User logs in successfully on Activation screen, but is no longer logged in at some point after that.

This happens when you do not execute a Save from the Activation screen, and then execute a Cancel on a subsequent screen, before a Save is executed.

Always execute a Save immediately after credentials are successfully validated on the Activation screen.

Mobile Workflow Exception Handling

Describes how to handle a blocked mobile workflow.

If a mobile workflow is not received or processed on a device, this may indicate the mobile workflow is blocked in the message queue. By default, Unwired Server retries actions that threw recoverable exceptions every 15 minutes and it retries actions that threw unrecoverable exceptions every 3 days. Both types will continue to retry indefinitely, unless the administrator intervenes, either by fixing the error or by unblocking the mobile workflow in the Sybase Control Center queue.

This typically indicates that a workflow operation failed with a recoverable or unrecoverable error. To resolve the situation:

1. Check the Unwired Server and mobile workflow client trace logs for information.
 - Unwired Server logs:
 - Aggregated Unwired Server logs – `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\logs`
 - Messaging service log details – `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\MessagingServer\Data\Log`
`<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\MessagingServer\Trace`

- Workflow client trace logs:
 - Single node – <UnwiredPlatform_InstallerDir>\UnwiredPlatform\Servers\MessagingServer\Data\ClientTrace
 - Cluster CDB node – <UnwiredPlatform_InstallerDir>\Data\Messaging\ClientTrace
- 2. Use information in the logs to resolve the problem.
- 3. Use Sybase Control Center to check message queue status, and to suspend, resume, unblock, or remove items in the queue. See:
Sybase Control Center for Sybase Unwired Platform > Manage > Managing Unwired Platform > Routine System Maintenance Tasks > Checking System Status > Checking Mobile Workflow Users and Queues

Unable to Deploy Workflow

Problem: When generating the Mobile Workflow package, you get an error that the Mobile Workflow package cannot be deployed similar to this:

```
=====
Deployment to Unwired Server
=====
Deploying the workflow
Unable to deploy workflow:
System.Web.Services.Protocols.SoapException: Could not find a
part of the path 'C:\Sybase\UnwiredPlatform\Servers\MessagingServer
\Data\Mobile Workflow\117_1'.
at Admin.ReplaceWorkflow(Byte[] baZippedPackage)
```

Explanation: Some software, such as Microsoft Security Essentials, locks the temp folder, or files in the temp folder when they are written by the Workflow installation routine, thus preventing the Directory.Move from succeeding.

Solution: Disable Microsoft Security Essentials.

Index

.p12 certificates 97

A

activating devices 103
 Advanced Encryption Standard 118
 AES
 See also Advanced Encryption Standard
 AES-128 121
 AES-256 119
 Alert Message property 86
 Alerts property 86
 Android 77
 API.js 147
 APNS 81
 APNS Device Token property 86
 App Store 89
 Apple push notification properties 86
 Apple push notification, configuring 85
 AttachmentViewer 70, 72
 configuring 69
 limitations 71

B

Badges property 86
 BlackBerry 92
 BlackBerry 5.0 144
 BlackBerry Desktop Manager 92

C

cached data lookup pattern
 data flow diagram 19
 overview 19
 Callbacks.js 166
 CallbackSet 166
 certificate picker 109
 Certificate.js 173
 certificates
 for context variables 106
 client trace logs 226
 ClientIconIndex 203
 conditional navigation 194

conditional start 196
 connection settings
 configuring 93
 device 93
 Hybrid Web Container 93
 content security 118
 Android 119
 BlackBerry 118
 iOS 121
 content type preference, changing 125
 context variables 108
 configuring 106
 controls 56
 convertToSUType() 149
 creating keys 47
 credential functions 166
 Credentials
 dynamic 108
 static 108
 CredentialsCache 203
 custom actions
 adding 59
 configuring 69
 properties 60
 types 60
 Custom.js 140
 custom.js file
 methods 176
 customAfterNavigateForward 176
 customAfterReportErrorFromNative 182
 customAfterShowScreen 176
 customAfterSubmit 176
 customAfterWorkflowLoad 176
 customBeforeMenuItemActivate 176
 customBeforeNavigateBackward 176
 customBeforeNavigateForward 176
 customBeforeReportErrorFromNative 182
 customBeforeShowScreen 176
 customBeforeSubmit 176
 customBeforeWorkflowLoad 176
 customValidateScreen 176

D

data change notification 30
 GET 28

Index

- JSON format 28
- POST 28
- request response 31
- DCN 31
- debugging 138
- default locale, creating 126
- defining an MBO
 - for cached data lookup 20
 - for real-time data lookup 10
- DeleteProcessedMessages 203
- Delivery Threshold property 86
- deploy 74
- deploy mode 129
- deploying
 - mobile application project 129
- deploying the workflow 227
- deployment mode
 - replace 76
 - update 76
- device platforms 37
- device users
 - assigning mobile workflow packages 105
- devices
 - Apple push notification properties 86
- documentation roadmap 1
- Dynamic authentication 110

E

- editing
 - locale properties file 129
- EIS error codes 223–225
- Enable property 86
- encoding type
 - changing 125
 - default 125
 - ISO-8859-1 125
 - non-ASCII 125
 - UTF-8 125
- encryption key length 118
- encryption policy 188
- error codes
 - EIS 223–225
 - HTTP 223–225
 - mapping of SAP error codes 225
 - non-recoverable 223, 224
 - recoverable 223, 224

F

- file associations 125

- findByParameter
 - binding to a Workflow menu item 10
- findByParameter object query 14
- Flow Design
 - general properties 50
 - localization 50
 - matching rules 50
 - page properties 50
 - rulers and grid 50
- functions
 - general utility 147
 - resource 183
 - workflow UI 152

G

- general utility functions 147
- generate old key values 67
- generated files 140
- getCurrentMessageValueCollection() 160
- getHTMLValue() 149
- getISODateString() 149
- getISODateTimeStringToDisplay 149
- getLocaleDateString() 149
- getPicture 167
- getWorkflowMessage() 160

H

- hard coded credentials 108
- HTML format 201
- HTTP error codes 223–225
- Hybrid Web Container
 - architecture 3
 - customization 3
 - management 3
 - offline capabilities 3

I

- image 71
- installing 87, 89
- internationalization
 - Mobile Workflow Forms editor 131
 - on the device 132
- InvokeOnClient 203
- iOS 80
- iOS push notification properties 86
- iTunes 90

J

jquery.mobile-1.0.css 142

K

key collection menu item 67

Keys 48

keys, creating 47

L

listview, generated keys 50

locale

 editing 129

 properties file 129

locale properties file

 creating 126

 validating 126

localization 123

 creating a new locale 126

 limitations 124

 Mobile Workflow package 125

 task flow 125

Localization 50

 current locale 130

 updating the current locale 130

look and feel 222

look and feel files 142

M

manage 103

manifest.xml 203

manually adding a starting point 40

MarkProcessedMessages 203

master.css 142

matching rules

 specifying 16

matching rules, adding 47

menu item 58

 cancel 62

 close 62

 open 62

 properties 60

 save 62

 types 60

menu item, creating 62

menu items

 adding 59

 custom 69

 online request 65

message-based application 129

messaging device registration 103

messaging device setup 103

methods

 utility 149

 validation 163

mobile application project, deploying 129

mobile workflow

 client trace logs 226

 exception handling 226

 unblocking 226

Mobile Workflow Forms Editor 37

 preferences 38

 preferences, Mobile Workflow Forms Editor
 38

 setting preferences 38

Mobile Workflow Generation wizard 74

mobile workflow package

 generated files 140

mobile workflow packages

 assigning device users 105

 configuring notification mailbox 104

ModuleDesc 203

ModuleDisplayName 203

ModuleName 203

ModuleVersion 203

N

native device functions 156

non HTTP authentication request 30

notification extraction rules 46

notification extraction rules, editing 46

notification mailbox 104

notification, matching rules 47

O

object queries

 binding to a workflow menu item 21

object query parameters

 defining a control that passes 22

offline capabilities 3

online request menu item 65

optimized for performance 145

Index

OTA 92
over the air 92

P

parameter mappings 49
parseBoolean() 149
parseDateTime() 149
PersistAppDomain 203
PersistentContent 118
PersistentContentListener 118
PersistentStore 118
preferences
 appearance 125
 content types 125
 general 125
ProcessUpdates 203
properties
 push notification for iOS 86
 server-initiated 43
Properties
 Flow Design page 50
PurchaseOrderSample 126
push notification properties for iOS 86

Q

query types
 addAllmenuitems 199
 addMenuItem 199
 alert 199
 clearrequestcache 199
 clearrequestcacheitem 199
 close 199
 downloadattachment 199
 formredirect 199
 loadtransformdata 199
 logtoworkflow 199
 removeallmenuitems 199
 rmi 199
 setscreentitle 199
 showattachment 199
 showcertpicker 199
 showInBrowser 199
 showlocalattachment 199
 submit 199

R

real-time lookup pattern
 data flow diagram 9

 overview 9
 registering messaging devices 103
 RequiresActivation 203
 resource functions 183
 rmi.xml 138
 RSA algorithm 97

S

Screen Design
 page properties 54
send a notification 134
sending server notification to a device 18
server notification pattern
 creating an MBO for 14
 data flow diagram 14
 overview 14
server-driven notification
 creating 16
server-initiated properties 43
server-initiated starting point 41
setHTMLValue() 149
shared storage 189
SharedStorage 189
showErrorFromNative 182
single sign-on
 using credentials 112
 using SSO2 tokens 114
 using static SSO2 tokens 116
 using static X.509 certificates 113
single sign-on task flow 28
Sounds property 86
SQLite Encryption Extensions (AES-128) 121
starting point
 adding manually 40
starting points
 Activate 40
 Client-initiated 40
 Credential request 40
 editing 47
 server-initiated 41, 47
 Server-initiated 40
static authentication 109
stylesheet.css 142
submit menu item 63
SupCertificateIssuer 108
SupCertificateNotAfter 108
SupCertificateNotBefore 108
SupCertificateSubject 108

SUPMobileWorkflow.replaceMobileWorkflowCertificate() 117
 SupPassword 108
 for context variables 106
 supported attachment types 70
 SUPStorage 188
 SUPStorage.js 186
 SupUser 108
 for context variables 106

T

task flow 6
 testing
 X.509 certificates 98, 102
 tutorial
 configuring the Android emulator 77

U

Unwired Server logs 226
 URL parameters 198
 UTF-8 encoding 125

V

validateAllScreens 163
 validateControl 163
 validateDate 163
 validateDateTime 163
 validateEmail 163
 validateNumber 163

validateRegularExpression 163
 validateScreen 163
 validateText 163
 validateTime 163
 variables, context
 configuring 106
 viewing workflow messages
 Android 134
 BlackBerry 134
 iOS 134
 Windows Mobile 134

W

workflow client
 using credentials 112
 workflow clients
 and static SSO2 tokens 116
 and static X.509 certificates 113
 using credentials in 112
 using SSO2 tokens in 114
 Workflow control
 that passes object query parameters 11
 Workflow screen
 that displays results 11
 workflow_CustomLookAndFeel.xml 222
 workflow_jQueryMobileLookAndFeel.xml 222
 workflow.html 142
 workflow.xml 222
 WorkflowClient.dll 211
 WorkflowClient.xml 211

