



**Developer Guide for Mobile Workflow
Packages**

Sybase Unwired Platform 2.0

ESD #1

DOCUMENT ID: DC01218-01-0201-01

LAST REVISED: July 2011

Copyright © 2011 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.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. 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 Sun Microsystems, Inc. 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	5
Hybrid Web Container Architecture	5
Mobile Workflow Development Task Flow	7
Identify a Business Process for Workflow	
Development	8
Hybrid Web Container Patterns	10
Server Notification	11
Online Lookup	17
Cached Data	23
Mobile Workflow Development	31
Tools	31
The Mobile Workflow Forms Editor	31
Mobile Workflow Data Change Notification	57
Mobile Workflow Data Change Notification	
Development Life Cycle	60
Mobile Workflow DCN Request using HTTP	
Authentication	61
Non HTTP Authentication Workflow DCN	
Request	61
Implementing the Data Change Notification	
Filter Class	62
Mobile Workflow DCN Request Response	62
Security	63
Credentials	63
Configuring the Workflow Application to Use	
Credentials	66
Localization and Internationalization	72

Localization Limitations	72
Localizing a Mobile Workflow Package	73
Mobile Workflow Package Internationalization ...	79
Internationalization on the Device	81
Deploy a Mobile Workflow Package to Unwired Server	82
Generating the Files For a Mobile Workflow Package	82
Generated Mobile Workflow Files	84
Mobile Workflow Package Customization	86
Customizing a Mobile Workflow Package	86
Repackaging Mobile Workflow Package Files	87
Debugging Custom Code	87
Viewing an attachment as part of the Mobile Workflow Package	88
Manage a Mobile Workflow Package	89
Registering and Reregistering Messaging Devices	89
Enabling and Configuring the Notification Mailbox	90
Assigning and Unassigning Device Users	91
Activating the Workflow	92
Afaria Provisioning and Mobile Device Management	92
Install and Configure the Mobile Workflow Container On the Device	92
Preparing Android Devices for the Mobile Workflow Package	92
Preparing iOS Devices for the Mobile Workflow Package	93
Preparing Windows Mobile Devices for the Mobile Workflow Package	101
Preparing BlackBerry Devices for the Mobile Workflow Package	104

Installing Certificates and Testing on Simulators and Devices	106
Testing Mobile Workflow Packages	111
Testing Server Initiated Mobile Workflow Packages ...	111
Launching a Server-initiated Mobile Workflow on the Device	112
Reference	113
Custom.js File	113
Using Third-party JavaScript Files	115
Generated HTML Files	115
CSS Files	115
Workflow Client API	120
General Utility Functions	120
Mobile Workflow Utility Functions	121
Workflow UI Functions	123
Mobile Workflow Application Native Device Functions	127
Workflow Message Data Functions	133
Workflow Validation Functions	136
Resource Functions	140
Troubleshoot	141
HTTP Error Codes	141
Recovering from EIS Errors	142
Mapping of EIS Codes to Logical HTTP Error Codes .	143
Credentials Are Lost after User Successfully Passes Activation Screen	144
Mobile Workflow Exception Handling	144
Index	147

Contents

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

Documentation Roadmap for Unwired Platform

Learn more about Sybase® Unwired Platform documentation.

Table 1. Sybase Unwired Platform Documentation

Document	Description
<i>Sybase Unwired Platform Installation Guide</i>	<p>Describes how to install or upgrade Sybase Unwired Platform. Check the <i>Sybase Unwired Platform Release Bulletin</i> for additional information and corrections.</p> <p>Audience: IT installation team, training team, system administrators involved in planning, and any user installing the system.</p> <p>Use: during the planning and installation phase.</p>
<i>Sybase Unwired Platform Release Bulletin</i>	<p>Provides information about known issues, and updates. The document is updated periodically.</p> <p>Audience: IT installation team, training team, system administrators involved in planning, and any user who needs up-to-date information.</p> <p>Use: during the planning and installation phase, and throughout the product life cycle.</p>

Document	Description
<i>New Features</i>	<p>Describes new or updated features.</p> <p>Audience: all users.</p> <p>Use: any time to learn what is available.</p>
<i>Fundamentals</i>	<p>Describes basic mobility concepts and how Sybase Unwired Platform enables you to design mobility solutions.</p> <p>Audience: all users.</p> <p>Use: during the planning and installation phase, or any time for reference.</p>
<i>System Administration</i>	<p>Describes how to plan, configure, manage, and monitor Sybase Unwired Platform. Use with the <i>Sybase Control Center for Sybase Unwired Platform</i> online documentation.</p> <p>Audience: installation team, test team, system administrators responsible for managing and monitoring Sybase Unwired Platform, and for provisioning device clients.</p> <p>Use: during the installation phase, implementation phase, and for ongoing operation, maintenance, and administration of Sybase Unwired Platform.</p>
<i>Sybase Control Center for Sybase Unwired Platform</i>	<p>Describes how to use the Sybase Control Center administration console to configure, manage and monitor Sybase Unwired Platform. The online documentation is available when you launch the console (Start > Programs > Sybase > Sybase Control Center, and select the question mark symbol in the top right quadrant of the screen).</p> <p>Audience: system administrators responsible for managing and monitoring Sybase Unwired Platform, and system administrators responsible for provisioning device clients.</p> <p>Use: for ongoing operation, administration, and maintenance of the system.</p>

Document	Description
<i>Troubleshooting</i>	<p>Provides information for troubleshooting, solving, or reporting problems.</p> <p>Audience: IT staff responsible for keeping Sybase Unwired Platform running, developers, and system administrators.</p> <p>Use: during installation and implementation, development and deployment, and ongoing maintenance.</p>
Tutorials	<p>Tutorials for trying out basic development functionality.</p> <p>Audience: new developers, or any interested user.</p> <p>Use: after installation.</p> <ul style="list-style-type: none"> • Learn mobile business object (MBO) basics, and create a mobile device application: <ul style="list-style-type: none"> • <i>Tutorial: Mobile Business Object Development</i> • Create native mobile device applications: <ul style="list-style-type: none"> • <i>Tutorial: BlackBerry Application Development</i> • <i>Tutorial: iOS Application Development</i> • Create a mobile workflow package: <ul style="list-style-type: none"> • <i>Tutorial: Mobile Workflow Package Development</i>
<i>Sybase Unwired WorkSpace – Mobile Business Object Development</i>	<p>Online help for developing MBOs.</p> <p>Audience: new and experienced developers.</p> <p>Use: after system installation.</p>
<i>Sybase Unwired WorkSpace – Mobile Workflow Package Development</i>	<p>Online help for developing mobile workflow applications.</p> <p>Audience: new and experienced developers.</p> <p>Use: after system installation.</p>

Document	Description
<p>Developer guides for device application customization</p>	<p>Information for client-side custom coding using the Client Object API.</p> <p>Audience: experienced developers.</p> <p>Use: to custom code client-side applications.</p> <ul style="list-style-type: none"> • <i>Developer Guide for BlackBerry</i> • <i>Developer Guide for iOS</i> • <i>Developer Guide for Mobile Workflow Packages</i> • <i>Developer Guide for Windows and Windows Mobile</i>
<p>Developer guide for Unwired Server side customization – <i>Developer Guide for Unwired Server</i></p>	<p>Information for custom coding using the Server API.</p> <p>Audience: experienced developers.</p> <p>Use: to customize and automate server-side implementations for device applications, and administration, such as data handling.</p> <p>Dependencies: Use with <i>Fundamentals</i> and <i>Sybase Unwired WorkSpace – Mobile Business Object Development</i>.</p>
<p>Developer guide for system administration customization – <i>Developer Guide for Unwired Server Management API</i></p>	<p>Information for custom coding using administration APIs.</p> <p>Audience: experienced developers.</p> <p>Use: to customize and automate administration at a coding level.</p> <p>Dependencies: Use with <i>Fundamentals</i> and <i>System Administration</i>.</p>

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:

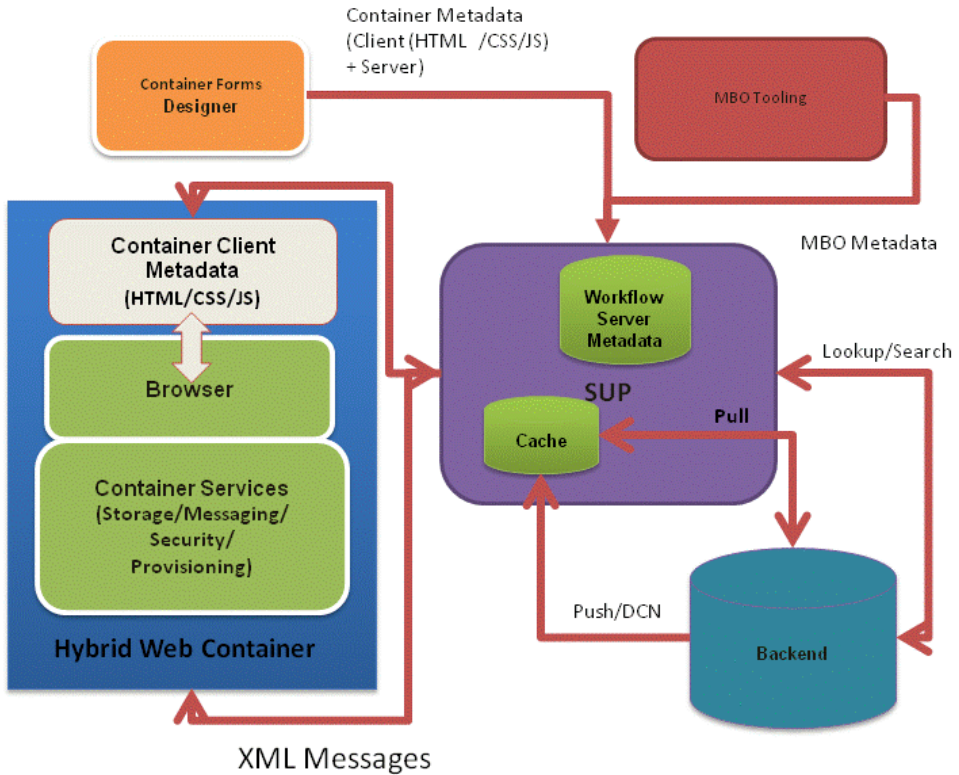
- The data aspects of the Mobile Workflow application are called mobile business objects (MBO), and “MBO development” refers to defining object data models with back-end enterprise information system (EIS) connections, attributes, operations, and relationships. Mobile Workflow applications can reference 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 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 file are documented in the *Reference* section of this guide.

Management

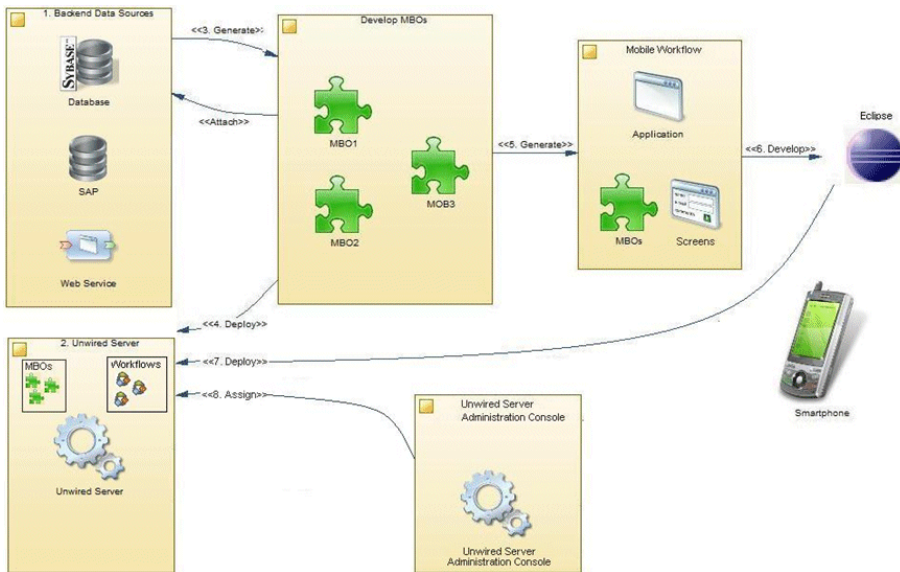
You can deploy Mobile Workflow Packages in Eclipse and manage them through the Sybase Control Center console. No device interaction is required from the administrator. Once a Mobile Workflow package is deployed into an existing installation, the administrator can configure the Mobile Workflow package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

Mobile Workflow Development Task Flow

Developing a Mobile Workflow package includes these basic tasks.

1. Open or import a mobile application project with predefined mobile business objects (MBOs).
2. Connect to Unwired Server.
3. Deploy the Mobile Application Project and select **Message-based** as the deployment mode.

4. Use the Mobile Workflow Forms editor to create a new Mobile Workflow package.
5. Generate screens by dragging and dropping MBOs and MBO operations from WorkSpace Navigator to the Flow Design page.
6. Manually create, delete, and edit screens, controls, menus, screen navigations, and so on.
7. Generate the Mobile Workflow package.
8. (Optional) Customize the generated `Custom.js` file.
9. (Optional) If you customized the Mobile Workflow package files, re-generate the Mobile Workflow package.
10. Deploy the Mobile Workflow package to Unwired Server.
11. Assign the Mobile Workflow package to the device user.
12. On the device, run, test and debug the Mobile Workflow package.



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

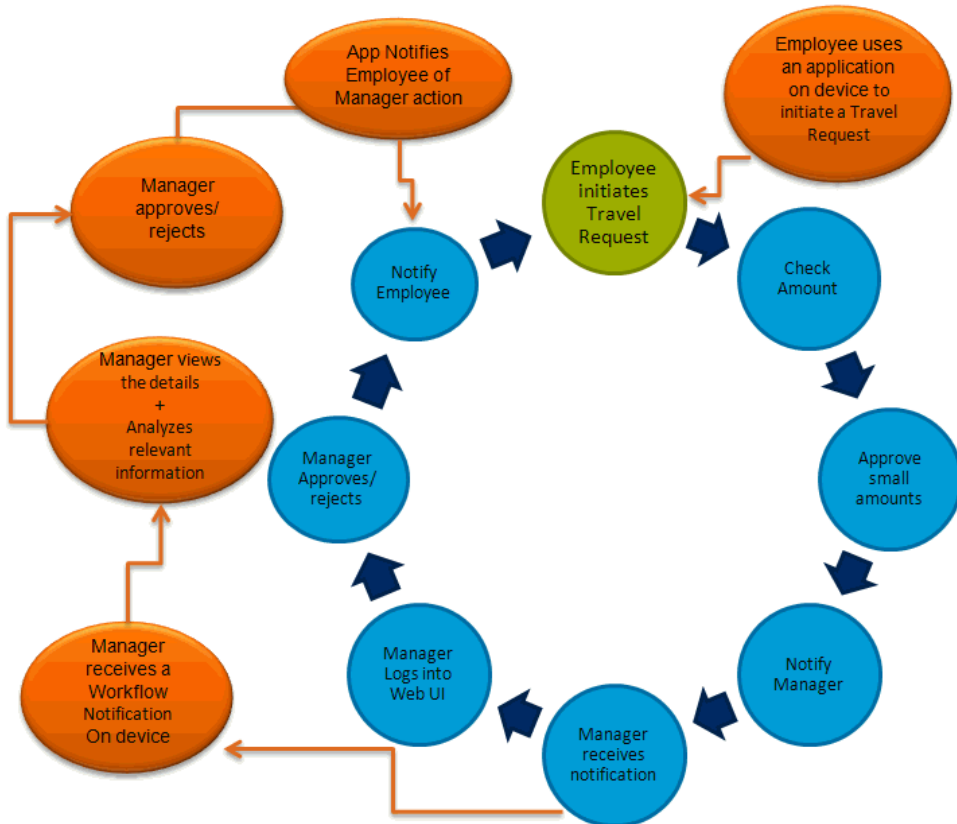
Identify a Business Process for Workflow Development

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.

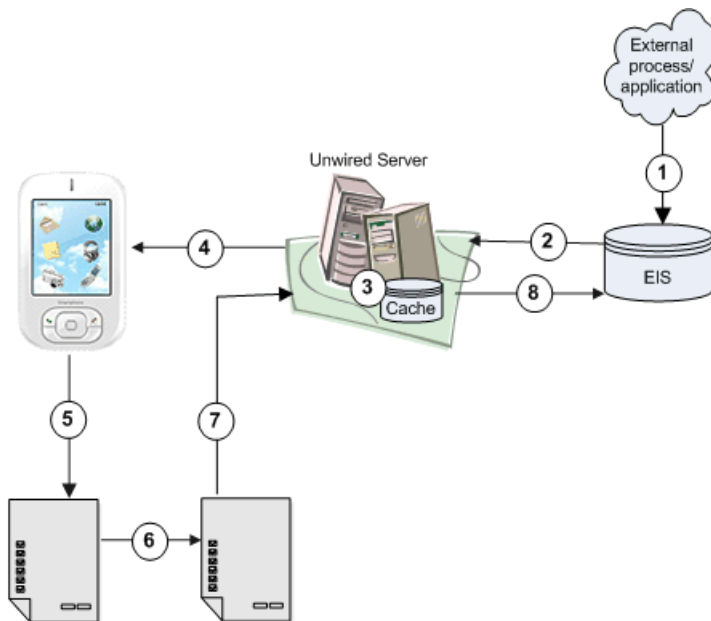


Hybrid Web Container Patterns

The Hybrid Web container allows you to create lightweight applications that implement various business solutions. The Hybrid Web container and the Unwired Platform uses three 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:

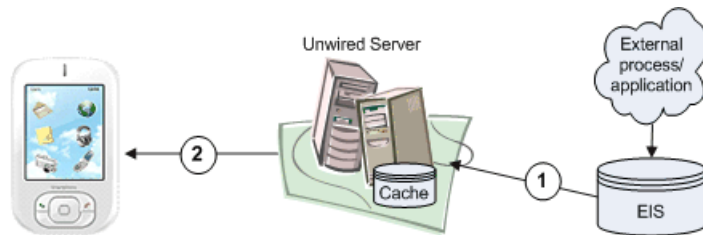


1. An external process or application updates EIS data.
2. The changed data triggers a data change notification (DCN), which is sent to Unwired Server, or a message from another workflow client updates mobile business object (MBO) data contained on Unwired Server.
3. The DCN could be programmed to update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

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 updated data are configured for the MBO/workflow package, Unwired Server sends 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

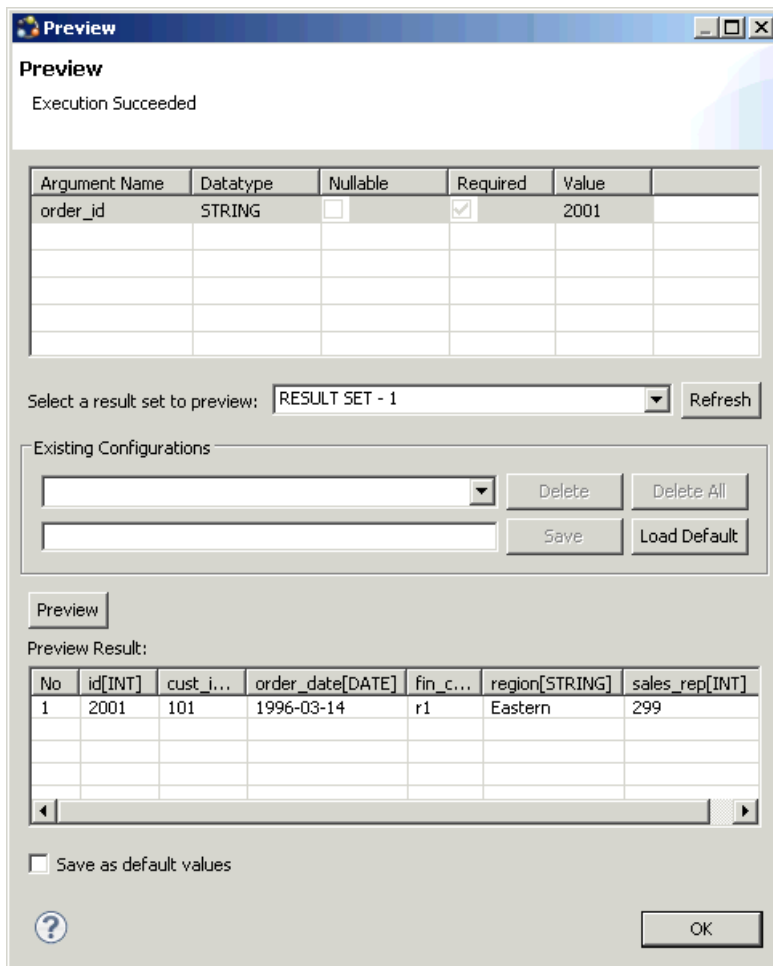
The server notification pattern supports any number of MBO definitions. For this example, create an MBO with one load parameter, map the parameter as a propagate-to attribute, then assign the MBO to a cache group that uses an Online policy.

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

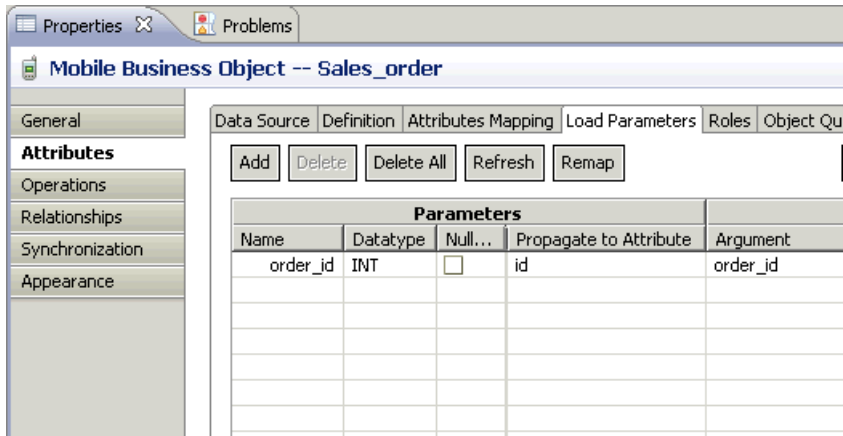
1. In Unwired WorkSpace, create an MBO from the sampledb database that has at least one load parameter. For example, you could define a Sales_order MBO as:

```
SELECT  id,
        cust_id,
        order_date,
        fin_code_id,
        region FROM sampledb.dba.sales_order
WHERE id = :order_id
```

2. Preview the MBO by selecting **Preview** from the Definition tab. Enter 2001 as the value. The preview returns one row from the sales_order table based on the id attribute (2001).



- In the MBO Properties view, click the **Load Parameters** tab, select the **id** attribute as the Propagate to attribute that maps to the order_id load parameter. Change the parameter and data source 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 parameter:

```
SELECT x.* FROM Sales_order x WHERE x.id = :order_id
```

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

Creating the Server-Driven Notification Starting Point

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

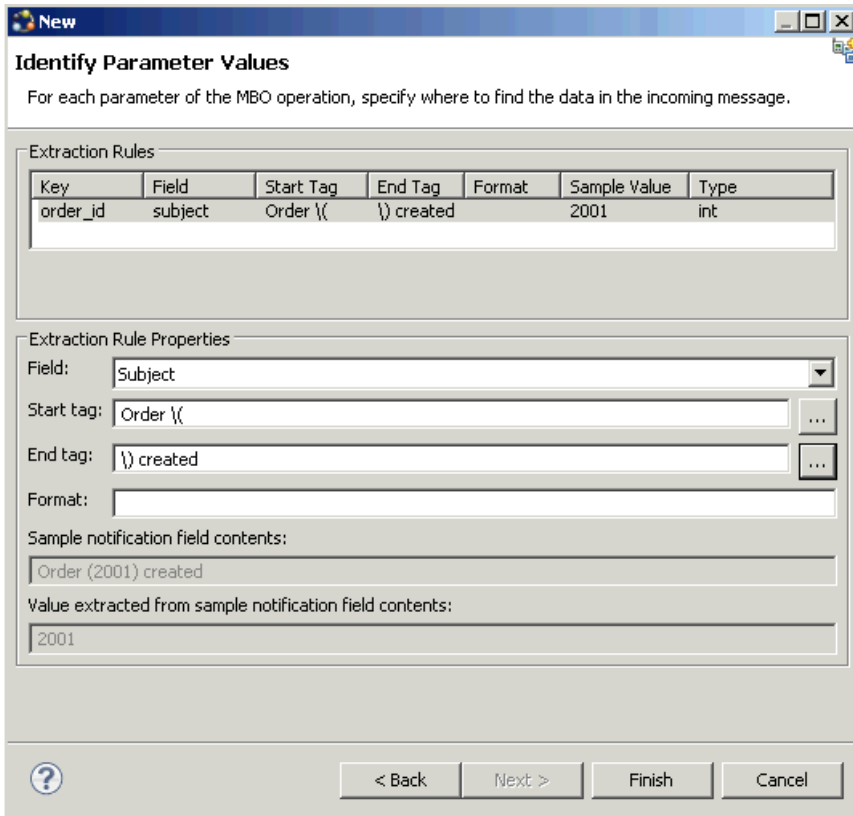
1. From Unwired WorkSpace, select **File > New > Mobile Workflow Forms Editor**.
2. Select the folder that contains the Sales_order MBO as the parent folder, name the file Sales_order.xbw, and click **Next**.
3. In the Starting Points screen, select **Responds to server-driven notifications**, and click **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**.

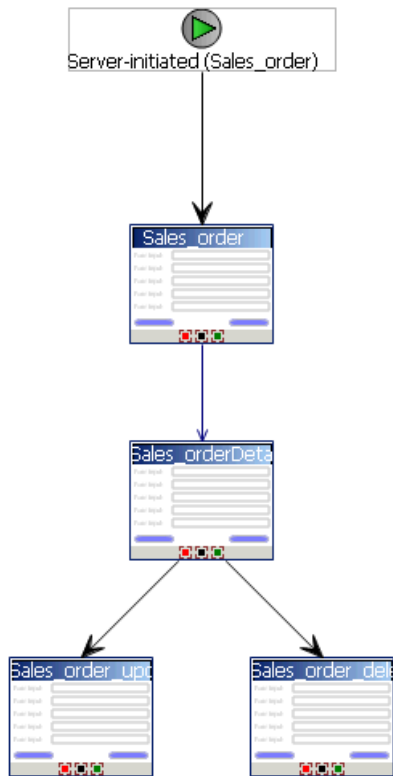
- d) Specify a sample notification. Enter Order (2001) created in the Subject line. Click **Next**.

- 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 form the client receives is populated with values returned by the findByParameter object query.



- 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:

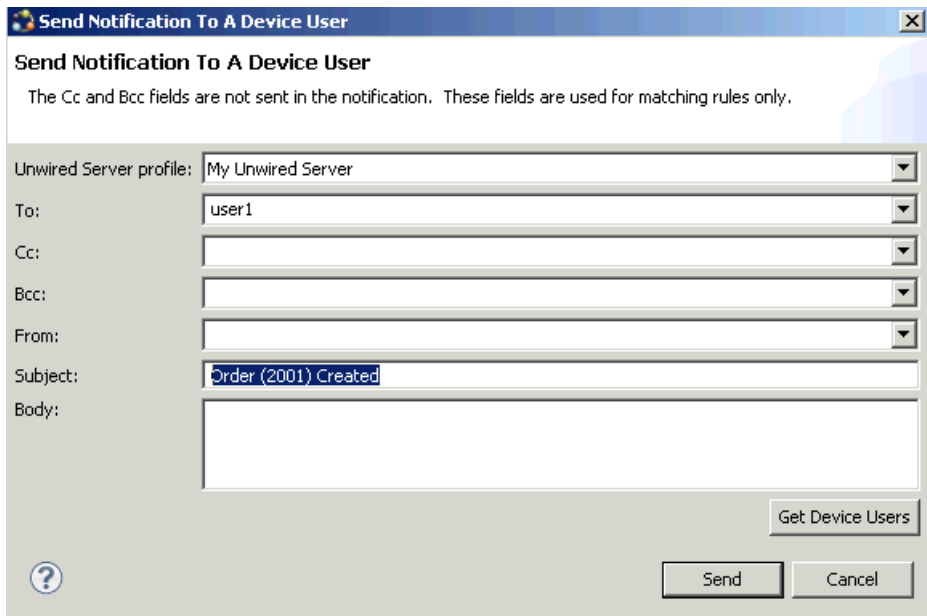
1. Register a device user and assign it to the workflow package in Sybase Control Center (SCC).
2. Download and configure the Sybase messaging client on the device or emulator to match those performed in SCC.

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



Send Notification To A Device User

The Cc and Bcc fields are not sent in the notification. These fields are used for matching rules only.

Unwired Server profile: My Unwired Server

To: user1

Cc:

Bcc:

From:

Subject: Order (2001) Created

Body:

Get Device Users

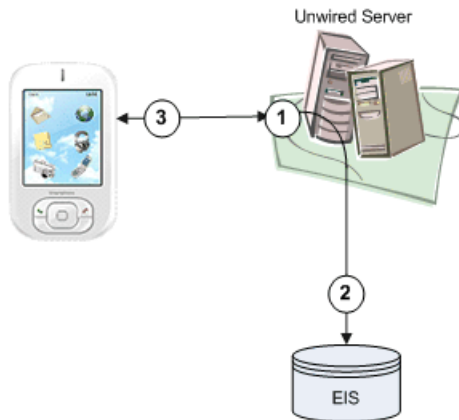
Send Cancel

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).

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 parameter 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 Parameters from Mapped Propagate to Attributes

Create an MBO with at least one load parameter, map parameters 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 parameter. For example, you could define an Emp MBO as:

```
SELECT id,
       empName,
```

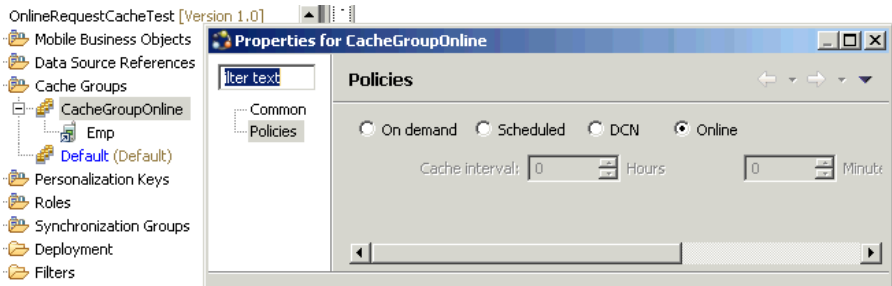
```
empDeptId FROM sampledb.dba.emp
WHERE empDeptId = :deptIdLP
```

2. In the MBO Properties view, select the **Load Parameters** tab, map each load parameter to be used as an operation load parameter for the Mobile Workflow package to a Propagate to Attribute. This example requires you to map the deptIdLP load parameter to the empDeptId attribute. You must also verify that data types are INT and the default value is a valid INT.

Parameters				
Name	Datatype	Null...	Propagate to Attribute	Argument
deptIdLP	INT	<input type="checkbox"/>	empDeptId	deptIdLP

Unmapped parameters are set to NULL, and get their value from the default value, if specified, or from the personalization key value to which they are mapped, if 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).

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 parameters that have propagate-to attributes:

Name	Return type	Create an index	Parameters	Query Definition
findByParameter	Multiple objects	<input checked="" type="checkbox"/>	deptIdLP	SELECT x.* FROM Emp x WHERE x.empDeptId = :deptIdLP

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

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.

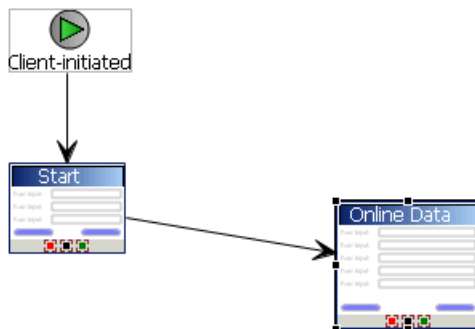
Prerequisites

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**.

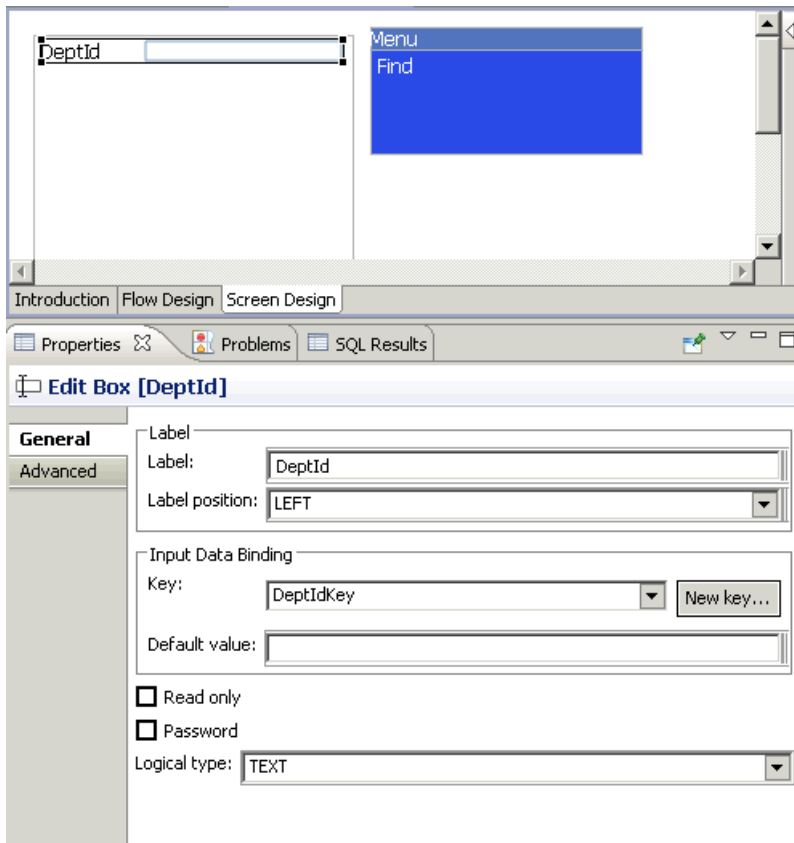
If you select the Parameter Mappings tab, you see all the load parameters defined for the MBO and used to generate the findByParameter object query. In addition to Key, you can map parameters to BackEndPassword, BackEndUser, DeviceId, DeviceName, DeviceType, UserName, MessageId, ModuleName, ModuleVersion, 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

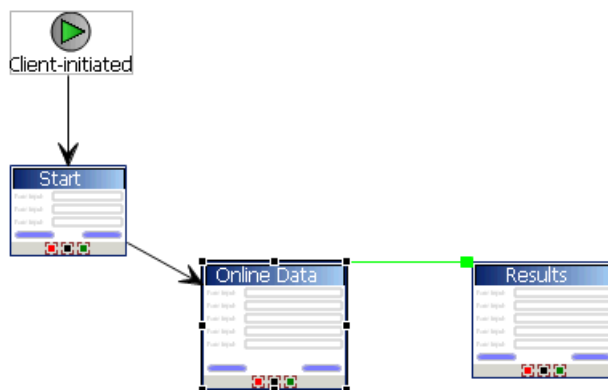
Add a control to pass the load parameter to Unwired Server. Define a screen that displays the results returned from the EIS.

1. Define a control that passes the load parameter 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.

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 parameters.

Task

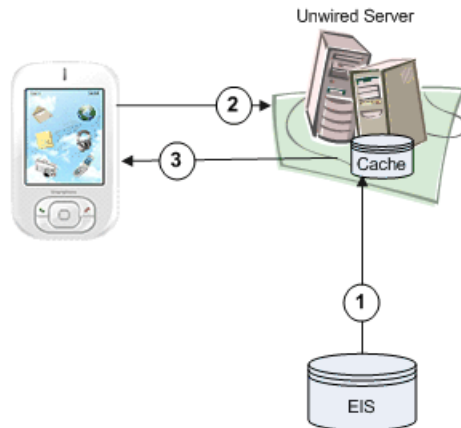
1. Launch the Mobile Workflow Forms Editor from Unwired WorkSpace and create a new Mobile Workflow form:
 - a) Select **File > New > Mobile Workflow Forms Editor**.
 - b) Select the parent folder that contains the MBO with a load parameter mapped to a transient personalization key. Name the file and 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 parameter 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 parameter values, and determine what data is cached in the Unwired Server cache (CDB) and returned to the client.

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.

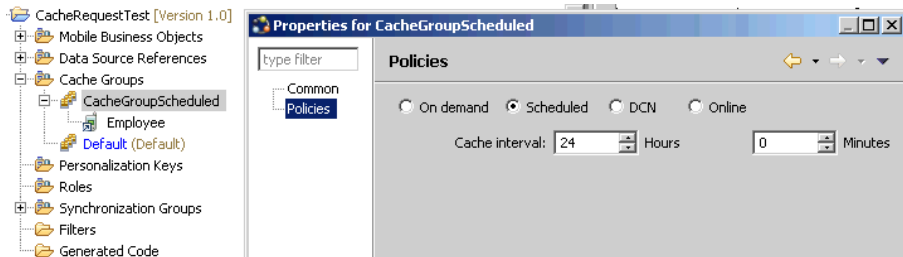
This example defines an MBO that retrieves employee benefit information for all employees of a given department based on the dept_id attribute using the findByDeptId object query.

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,
```

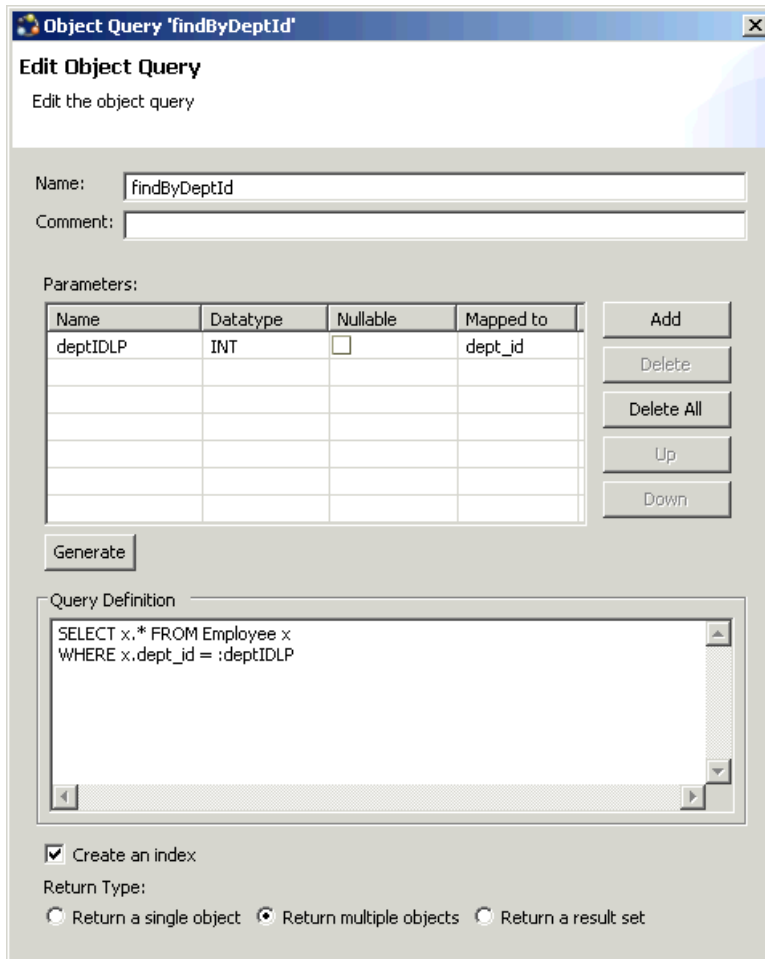
```
bene_life_ins ,
bene_day_care
FROM sampledb.dba.employee
```

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
```



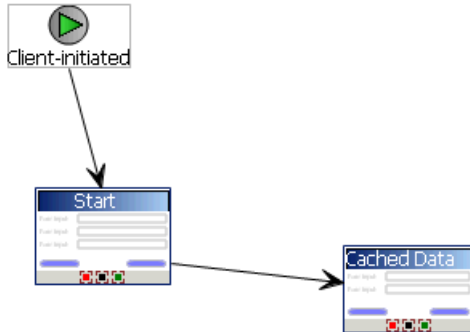
4. Deploy the project that contains the MBO to Unwired Server. Select **Message-based** in the deployment wizard.

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.



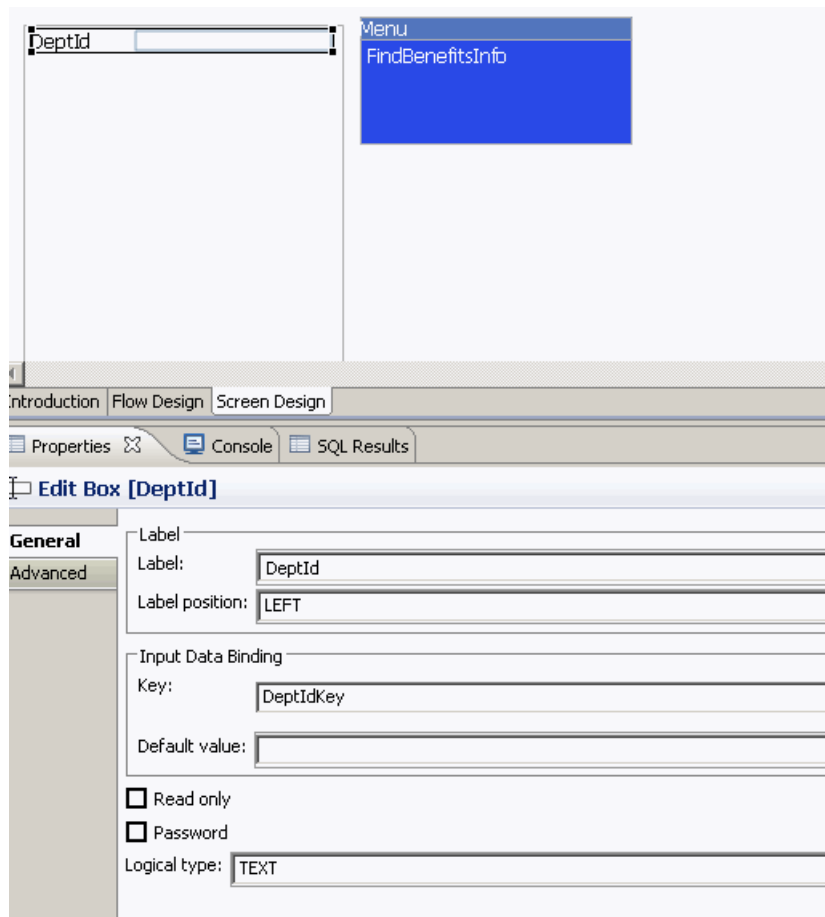
3. Highlight the menu item you want to map, or create a new menu item.
4. Define a Submit action named FindBenefitsInfo that invokes the findByDeptId object query:
 - a) In the Properties view, in the General properties for the selected menu item, select **Online Request** as the Type.
 - b) In the Details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

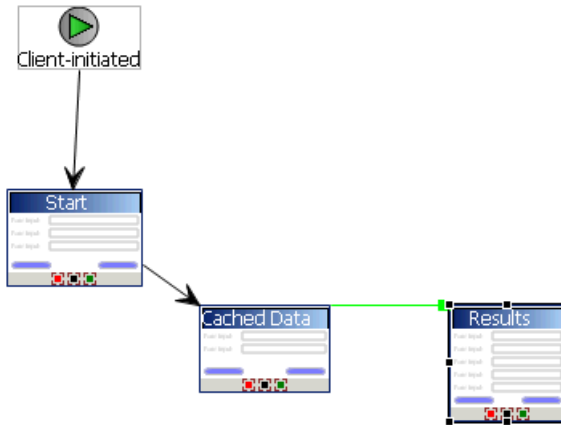
Defining the Control that Contains the findByDeptId Object Query Parameter

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.

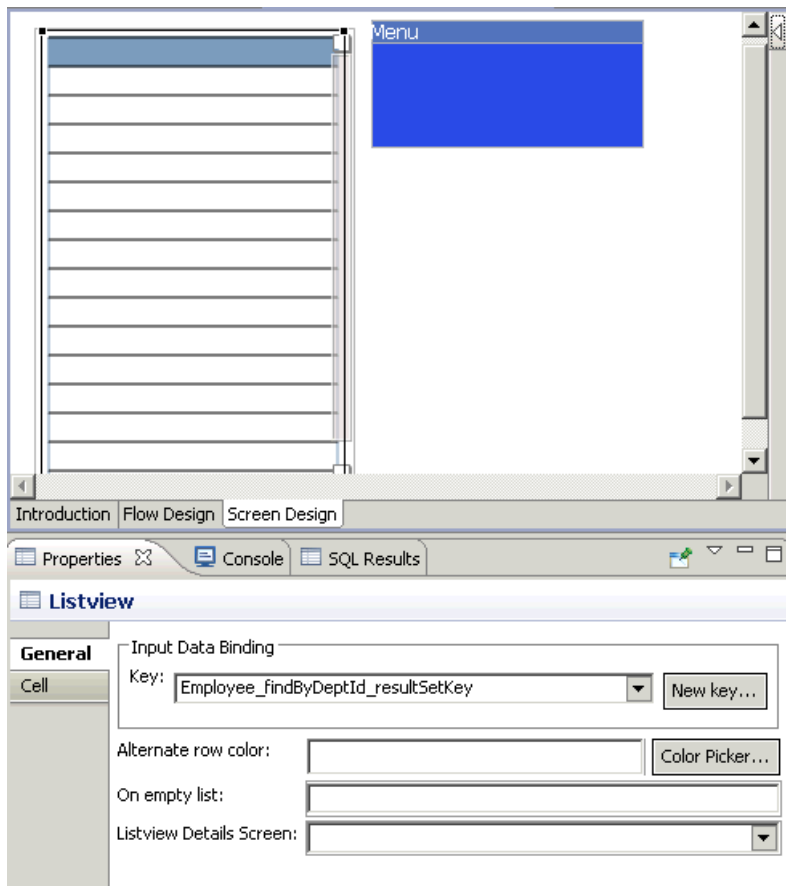
1. Define a control that passes the object query parameter to Unwired Server from the screen (named Cached Data) that contains the menu item (named FindBenefitsInfo) that invokes the findByDeptId object query:
 - a) Select an **EditBox** control and click in the control area.
 - b) Name the EditBox DeptId.
 - 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.



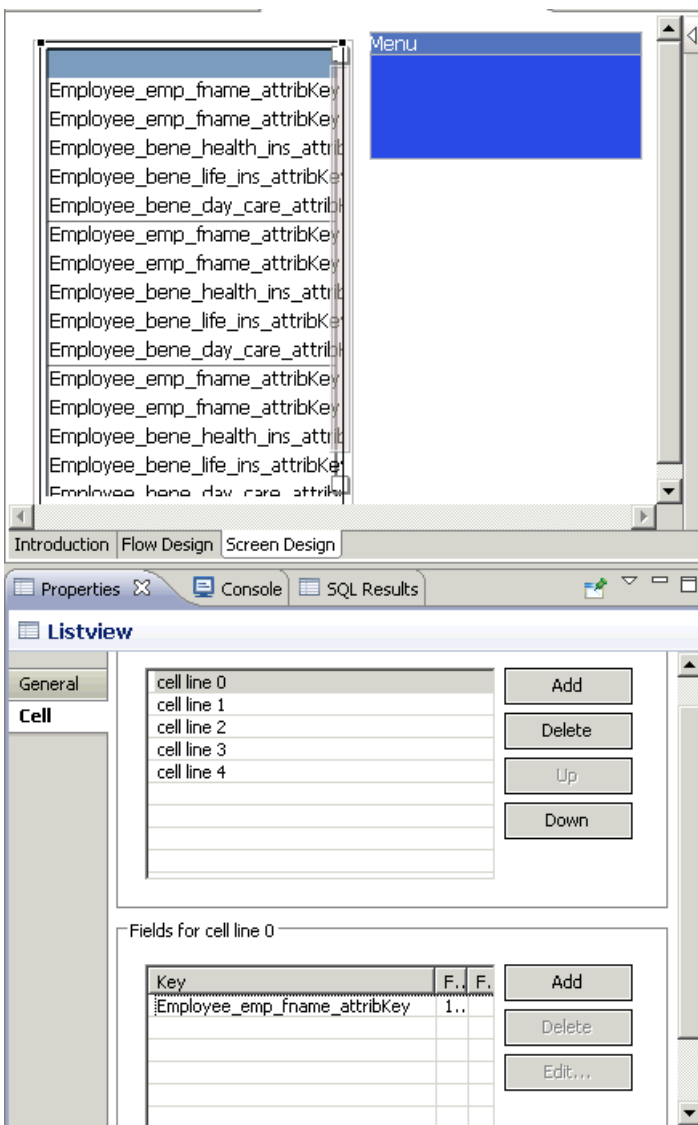
4. Configure the Results screen to display the results. In this example, the Employee MBO, contains seven attributes that identify the employee and their benefits. Create a Listview with a cell for each attribute to display the results returned from the cache as a list:
 - a) From the Flow Design window,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

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
- Android

See the *Sybase Unwired Platform Installation Guide* for supported version levels.

Tools

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 Sybase Mobile Workflow application installed and that are registered in Sybase Control Center.

The Mobile Workflow Forms Editor contains these pages:

Introduction

Create a new mobile workflow or double-click an existing one, to open the Introduction page of the Mobile Workflow Forms Editor. This page is the starting point for designing your mobile workflow form. You can access help from this page.

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 form. 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.

The Mobile Workflow Forms Editor supports two different types of look-and-feel. Select either:

- Optimize for performance – provides a more basic look-and-feel, which allows better performance. This means that 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 and JavaScript enhancements to customize the look-and-feel.
- Optimize for appearance – provides a more sophisticated, or native, look-and-feel but may affect performance.

Note: This option is not available on Windows Mobile platforms.

Preference settings determine the default behavior and style for the Flow Design and Screen Design pages and their objects. You can change the style and behavior of objects individually in the Properties view for a selected object. You can set these preferences for the Flow Design and Screen Design pages:

Preference	Description
Appearance	Set the color and font preferences for the designer.
Connections	<p>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.</p> <p>Select the default line style for new connections:</p> <ul style="list-style-type: none"> • Oblique (default) – the connection lines are not aligned with the horizontal or vertical axis and you can create break points 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.
Rulers and Grid	<p>Set the preferences for the ruler and grids:</p> <ul style="list-style-type: none"> • Ruler options: <ul style="list-style-type: none"> • Show rulers for new diagram • Ruler units • Grid options: <ul style="list-style-type: none"> • Show grid for new diagrams • Snap to grid for new diagrams – select if you want objects in the diagram to stick to grid lines when they are being moved. • Snap to shapes for new diagrams • Grid spacing (in inches)

Starting Points

Starting points specify which screen on the Mobile Workflow a 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 application. It does not activate on the second and subsequent times the mobile

workflow 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.

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. 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.
3. Add a connection between the starting point and the screen.
4. 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.
Link to file in the file system	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. Click Variables to define a new path variable. Path variables specify locations on the file system.

3. Click **Next**.
4. In the Starting Points page, select **Responds to server-driven notifications**.
5. In the E-mail Processing wizard, click **Search** to locate the mobile business object with which to associate the server-initiated starting point.
6. In the Search for Mobile Business Object wizard, click **Search**.
7. Select the mobile business object from the list, and click **OK**.
8. (Optional) Select an **Object query** and click **Next**.

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

9. Enter the contents for a sample notification message, and click **Next**.
10. Select the text that will serve as the matching rule by right-clicking and selecting **Select as Matching Rule**; then click **Next**.
11. From **Extraction Rules**, select a parameter, and specify from where the parameter receives data in the Extraction Rule Properties.

Note: The supported syntax for the regular expression is that used by the C# Match() methods(s) 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	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	The value of the field from the sample notification previously specified on the Specify Sample Notification page of the Notification Processing wizard.
Value extracted from sample notification field contents	The value the start tag and end tag matching rules would arrive at given the sample field value.

12. Click Finish.

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>

Keys

Click **Keys** in the left pane of the Properties view to see the keys that are associated with the server-initiated starting point.

Property	Description
Key name	The name of the associated key.
Type	The type of the associated key, for example, string.

Property	Description
<p>Data binding</p>	<p>Shows the data source to which the key is bound:</p> <ul style="list-style-type: none"> • Mobile business object attribute • Mobile business object relationship • MBO object query results <p>If you select an object query that returns multiple MBO instances, you can bind the result set to a Listview control using a key that is of the list type.</p> <ul style="list-style-type: none"> • User-defined • Extraction rule <p>If you create a server-initiated starting point that invokes an object query with a parameter, this value is extracted from the email message automatically.</p> <p>Click New to create a new key.</p> <p>Select any key and click Edit to edit the information for an existing key.</p> <p>Select a key and click Remove to remove an existing key.</p>

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.

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.

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\"(".
Tag after parameter	Enter the tag that indicates what to extract from the end of the field's contents, for example, "\)was sent".
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.
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.

Property	Description
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.
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
- 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 parameters 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 parameters. (Set the load parameters in the MBO Properties view, from the Load Parameters tab.)
2. Create a corresponding transient personalization key for each load parameter.
3. Map each load parameter 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.

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 <hr/> <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 <hr/> <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

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. It 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.

Adding a Menu Item

Add a menu item to the menu.

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

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

Menu Item Properties

Select the menu item on the Screen Design page to view and configure its properties using the Properties page.

General

Property	Description
Name	Enter a valid name for the menu item. <hr/> Note: To avoid wrapping of menu item names on Apple iOS devices, keep the name length to 10 or fewer characters. <hr/>

Property	Description
Key	Select the key with which to bind the menu item.
Default	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. It 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.

Property	Description
Details	You can configure menu item actions in the Details section. The properties available depend on the action type. See the topics for configuring the different menu item 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 item 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 form 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. <hr/> Note: To avoid wrapping of menu item names on Apple iOS devices, keep the name length to 10 or fewer characters. <hr/>
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	Select the type of action the menu item will perform when selected: <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 menu item of the first screen in a mobile workflow form must be associated with a mobile business object (MBO) operation or an object query.

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. From the Properties view, select the menu item for which to configure the properties.
3. In the Properties view, 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.
Invoke object query	Invoke the object query from the menu item.
Generate old value keys	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.
Submit confirmation message	(Optional) Enter text to show in a message box on the screen when the submit operation finishes successfully.

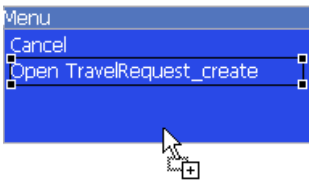
Property	Description
Resubmit confirmation message	(Optional) Enter text to show in a message box on the screen when the submit was not successful and thus must be resubmitted.

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	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 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.

Property	Description
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.
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.
Success screen	The screen to go to upon successful communication with the server.
Error screen	The screen to go to if there is an error when submitting the data to the server. Click Generate Error Screen to automatically generate error screens.

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.

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"> • Create 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.

Property	Description
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**.

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.

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 on the screen.
3. Configure the Attachment Viewer from the Properties view:
 - 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.

The AttachmentViewer control can be bound to data in one of two ways:

- Keys, or
- Object queries

Property	Description
Label	<ul style="list-style-type: none">• Label – enter the text that describes the control on the user interface.• Label position – the position of the label in relation to the control it describes.

Property	Description
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. Click New key to create a new key for the control. Default value – default value for the control.
Read-only	Select to make the control read-only.
Content	<p>The other way you can bind the AttachmentViewer 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 MIME type key – mime type for attachments that are supported, for example, *.doc, *.xls and so on File name – name of the file File size – size of the file <p>Note: Attachments requested on demand are limited to 5MB after Base64 encoding.</p>

- (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
- (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 Installing an attachment viewer like Documents To Go Premium Edition allows you to view additional file types, such as *.pdf, *.doc, *.xls, *.xlsx, *.ppt, and *.pptx.
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 Limitations

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

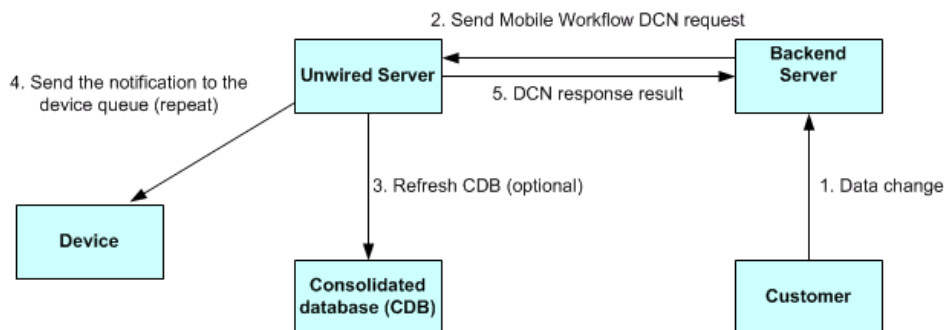
These limitations vary by platform.

Platform	Size limit
BlackBerry 5.0	The maximum size of a JavaScript variable is 500Kb, 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, and it can also run out of memory.
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.

Mobile Workflow Data Change Notification

Mobile Workflow data change notification (DCN) requests enable you to process the general DCN request and send notification to the device to keep the Mobile Workflow in sync with the back-end server when a change to the back-end server is made.

A typical scenario might look like this:



All DCN commands support both GET and POST methods. The EIS developer creates and sends a DCN to Unwired Server through HTTP GET or POST operations. The portion of the DCN command parameters that come after `http://host:8000/dcn/HttpAuthDCNServlet`, can all be in POST; any `var=name` can be in either the URL (GET) or in the POST. The HTTP POST method is more secure than HTTP GET methods; therefore, Sybase recommends that you include the `authenticate.password` parameter in the POST method, as well as any sensitive data provided for attributes and parameters.

Mobile Workflow DCN format

The Mobile workflow DCN request is a JSON string consisting of these fields:

1. Operation name(`op`) :**upsert** – the message must contain name/value pairs for every required attribute, and the name must exactly match the MBO attribute name.
:**delete** – provide only the name/value pairs for the primary key column(s).
These operations respectively insert or update, or delete a row in the CDB. Calling either of these operations does not trigger any other refresh action.
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 the Workflow DCN mechanism, the device user should first have established communication using the activation mechanism in Sybase Control Center.

Note: The "To" field must match the SUP user name—it is not the user name that is used for registering the device.

4. Subject (`subject`) – subject of the Mobile Workflow message.
5. Originator `<from>` – who the Mobile 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

When the Mobile Workflow DCN request is received, Unwired Server gets the **cmd** value from the request first. If the value of **cmd** is *wf* the DCN request is a Mobile Workflow DCN request. The flow of the Mobile Workflow DCN request is as follows:

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 Mobile Workflow DCN request message object.
3. The JSON format string is parsed into a Mobile Workflow DCN request object.
4. The DCN request in the Mobile Workflow message object is parsed and any of them 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 Mobile Workflow DCN request.
5. From the CDB, the server looks up all users assigned to the indicated Mobile Workflow package in the “to” attribute of the Mobile Workflow message, then matches them with the receiver list.

For every receiver, the server generates multiple Mobile Workflow messages (all Mobile 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 4, step 5 is not executed. If any errors occur in step 5, step 5 is not committed. If any errors occur in either of those steps, an HTTP 500 error is received.
7. Unwired Server invokes the `postProcessFilter`, if specified.

8. If no errors occur, the 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 Mobile Workflow DCN Result. Otherwise, the server returns an HTTP 500 error with the body of the JSON log records.

Mobile Workflow Data Change Notification Development Life Cycle

To use a DCN with the Mobile Workflow package:

1. Develop the Sybase Unwired Platform package and deploy it to Unwired Server.
2. Develop the Mobile Workflow form associated with the back-end database.
3. Log into Sybase Control Center and deploy the Mobile Workflow Package to Unwired Server.
4. Assign the Mobile Workflow package to a device.
5. Run the Mobile Workflow package on the device and subscribe.
6. Write the DCN request to update the back-end database.
7. Generate a Mobile Workflow request in the JavaScript Object Notation (JSON) data format. This request can include one, or many, general DCN requests.
8. Invoke the Mobile Workflow DCN servlet in one of two ways—HTTP authentication or non HTTP authentication.
9. Process the HTTP response. If the response code is 200, the Mobile Workflow DCN result can be read from the HTTP response, for example:

```
int returnCode = con.getResponseCode();
    if (returnCode != 200)
    {
        String rspErrorMsg = "Error getting response from the
server (error code "+ returnCode + ")";
        throw new AdminException(AdminException.Type.SERVER_ERROR,
            rspErrorMsg);
    }
    else
    {
        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);
    }
}
```

10. Verify the Mobile Workflow DCN request:
 - a) Check the CDB to verify it has been refreshed.
 - b) If the operation of the Mobile Workflow DCN request is "upsert," check the device to see if there is a new notification, or if the notification has been updated.

- c) If the operation of the request is “:delete,” verify on the device that the notification was deleted.

Mobile Workflow DCN Request using HTTP Authentication

If you do not want to use URL parameters to send login and password information, you can send the information to the HTTP server as part of the HTTP request header.

The DCN request messages can also be sent as HTTP POST data. The URL of an HTTP authentication Mobile Workflow DCN request is:

```
http://host:8000/dcn/HttpAuthDCNServlet?
cmd=wf&security=admin&domain=default&dcn_filter=aa.bb
[post data]
```

The *[post data]* is the content of the Mobile Workflow request.

Mobile Workflow DCN request using HTTP authentication:

```
URL url = new URL("http://<host>:8000/dcn/HttpAuthDCNServlet?
cmd=wf&security=default");
    HttpURLConnection huc = (HttpURLConnection)
url.openConnection();
    huc.setDoOutput(true);
    huc.setRequestMethod("POST");
    final String login = "supAdmin";
    final String pwd = "s3pAdmin";
    Authenticator.setDefault(new Authenticator()
    {
        protected PasswordAuthentication
getPasswordAuthentication()
        {
            return new PasswordAuthentication(login,
pwd.toCharArray());
        }
    });

    StringBuffer sb = new StringBuffer();
    sb.append(wfdcn_request);
    OutputStream os = huc.getOutputStream();
    os.write(sb.toString().getBytes());
    os.close();
```

Non HTTP Authentication Workflow DCN Request

You can also sent the Mobile Workflow DCN request in a non HTTP authentication way.

The URL is:

```
http://host:8000/dcn/DCNServlet?
cmd=wf&security=admin&domain=default&username=supAdmin&password=s3p
Admin&dcn_filter=aa.bb&dcn_request=<wfrequestdata>
```

Implementing the Data Change Notification Filter Class

Write and deploy preprocess and postprocess Mobile Workflow data change notification (DCN) filters to Unwired Server.

A system classloader is used to load the DCN filter class for the Mobile Workflow. To use a DCN filter class, jar the class file and put the jar file under the following folder.

1. Package your DCN filter class in a JAR file.
2. Place the JAR file in <UnwiredPlatform_InstallDir>\Unwired Platform\Servers\Unwired Server\lib\ext.
3. At a command prompt, run: <UnwiredPlatform_InstallDir>\Unwired Platform\Servers\Unwired Server\bin\configure-mms.bat <hostname>.

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"
}
]
}

```

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 to – the date the certificate is valid to
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.
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 *System Administration > Security Administration* documentation.

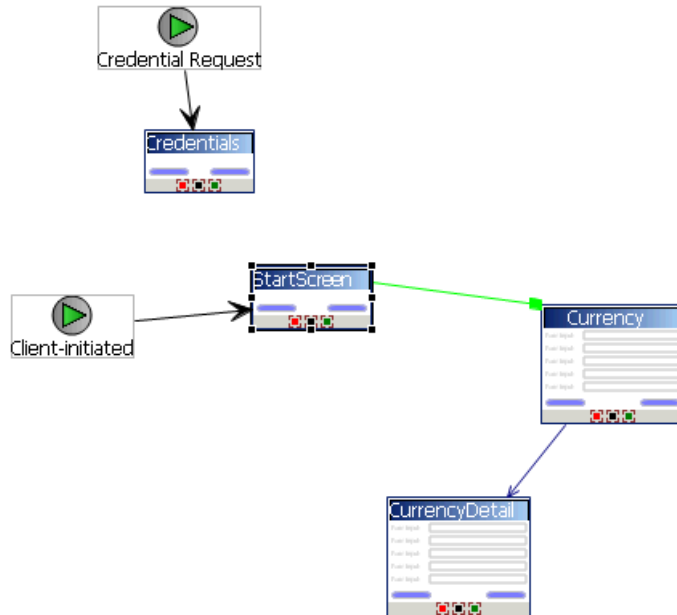
Configuring the Workflow Application to Use X.509 Credentials

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 username and password properties to system-defined "username" 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 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) 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 username and password portions for the selected server connection is replaced by the username and password propagated from the device application.

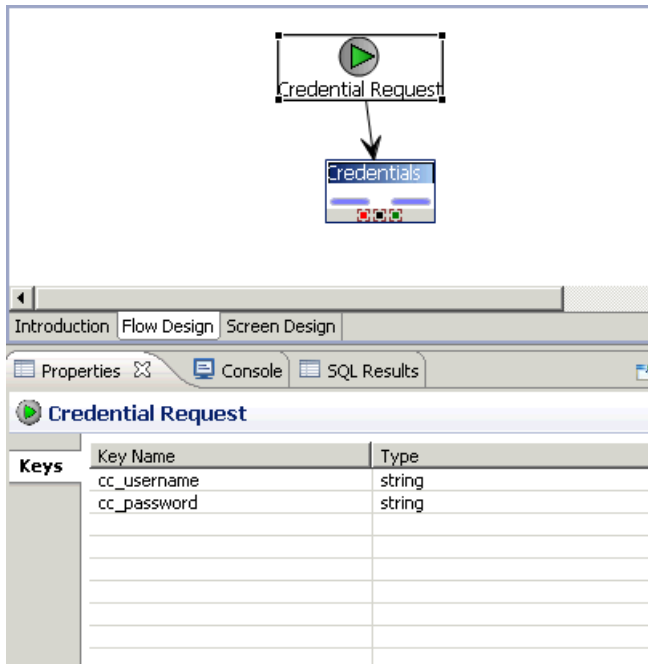
Note:

- 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).
-

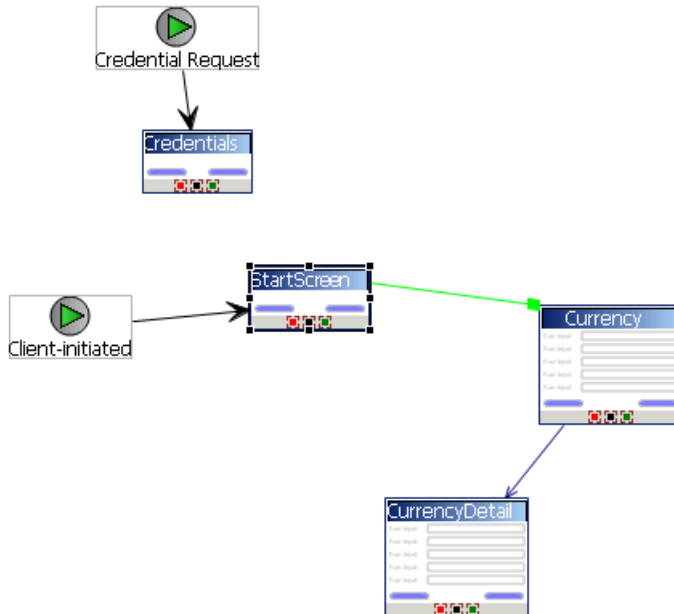
Configuring a Workflow Application to Use SSO2 Tokens

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.



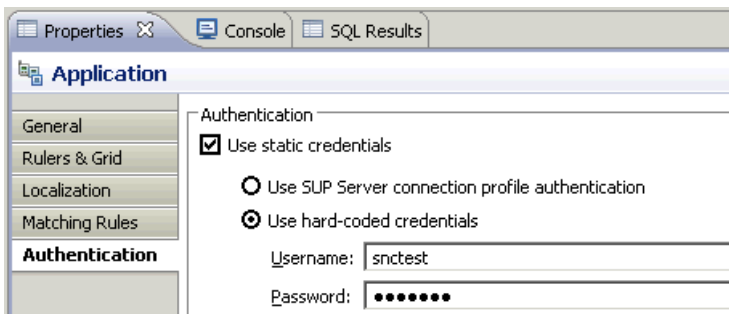
4. Double-click **Credentials** to open it in the Screen Design. Add a **Save screen** menu item to the Menu, and two edit boxes (`Username` and `Password`).
The `Save screen` saves the `Username` and `Password` entered by the workflow application. You could also add a **Submit workflow** menu item instead 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");
```

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.

Localization Limitations

Some restrictions for the locale properties files apply:

- 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.
- 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).

Creating and Validating a New Locale Properties File

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

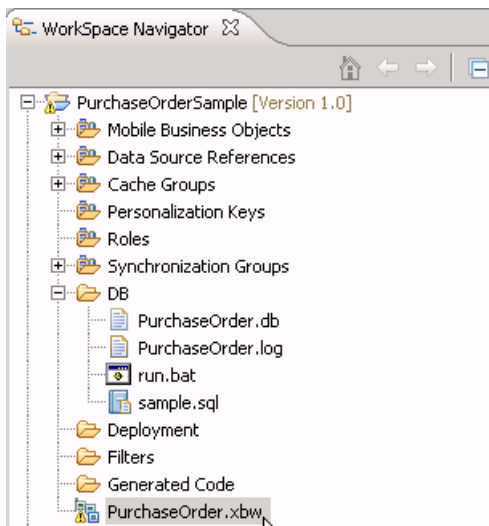
Prerequisites

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.

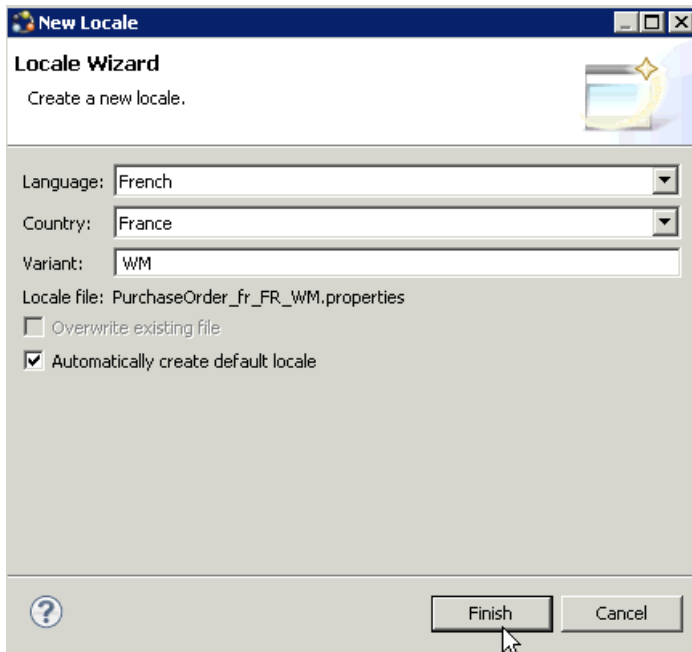


2. Click the **Flow Design** tab.
3. Right-click in a blank area on the Flow Design page, and select **Show Properties View**.
4. In the Properties view, on the left, click the **Localization** tab.
5. In the right pane, click **New**.
6. Select or enter the information for the new locale, select **Automatically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

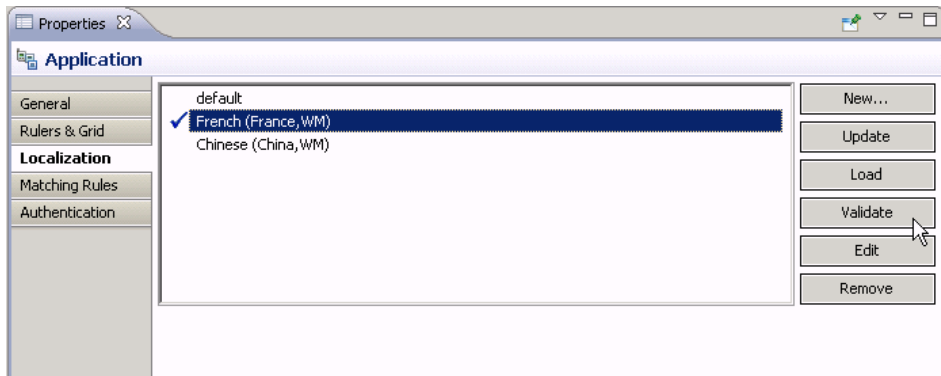
For example:

- 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.

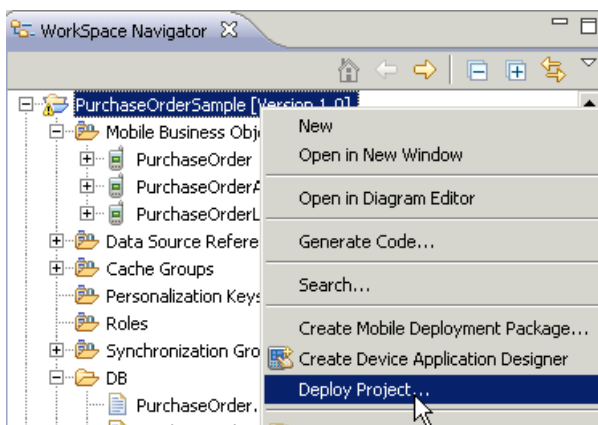
Prerequisites

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.

Configuring the Device for Localization

Goal: Configure the device.

Keep in mind that the device you are testing may be different from the devices available to consumers in other geographies, and the locales available on your device may be different from those available on other devices.

1. For Windows Mobile:
 - a) Go to **Settings > System > Regional Settings**.
 - b) Click **Region**.
 - c) Change the region to the desired locale, for example, French (France) or Chinese (China).



Note: The locales that appear in the Regional Settings depend on what you have installed on your device.

2. For iPhone devices:

- a) Go to **Settings > General > International > Language**.
- b) Change the Language setting to the desired language, for example, Français, and click **OK**.



- c) From **Settings > General > International > Language > Region Format**, select the corresponding Region Format and click **OK**.

3. For Android devices:

- a) Go to **Settings > Locale and text > Select locale**.
- b) Change the region to the desired locale, for example, English (Australia).

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.

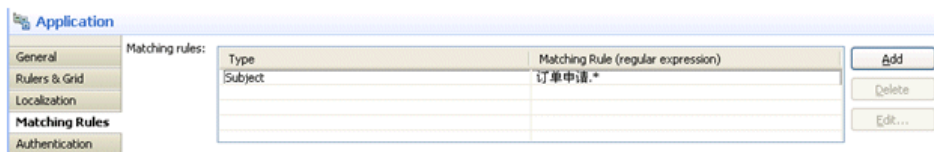
1. In the Screen Design page Properties view, click **Localization**.
2. Click **Update**.

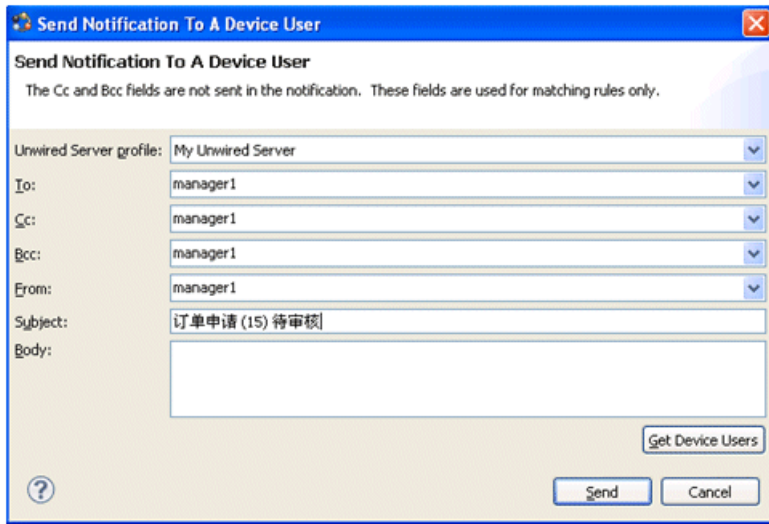
Mobile Workflow Package Internationalization

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

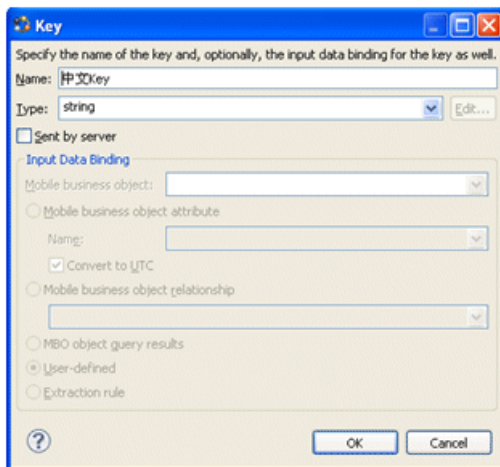
In the Mobile Workflow Forms Editor, you can use international data in:

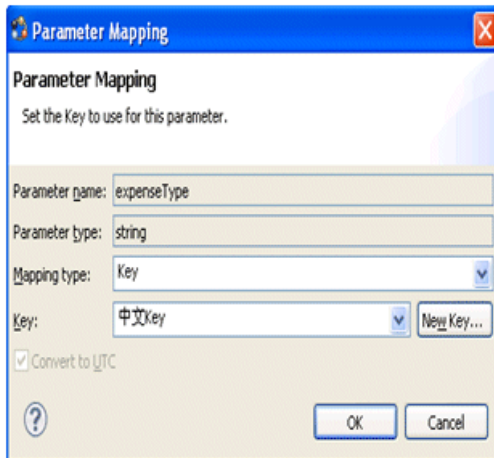
- Matching rules for notifications.





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





- 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:






Deploy a 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 form, optionally deploy the generated workflow package files to the server, and assign it 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 Workflow Package**, 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
Generate	Select to generate the mobile workflow package and its files. When this option is unchecked, the Mobile Workflow package files are not regenerated, so that modifications made to files that are normally regenerated will not be overwritten. This also means, however, that changes made in the Mobile Workflow editor will not be reflected in the generated files..
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 application and its generated files into a location outside of the current project. Click Browse to select the alternate location.
Unwired Server Profile	Select the Unwired Server profile with which to associate the mobile workflow application and extract the username and password credentials if you are using static authentication.
Deploy to an Unwired Server	Select to deploy the mobile workflow application to an Unwired Server.
Deploy Mode	<p>The deploy mode is automatically set and cannot be changed.</p> <ul style="list-style-type: none"> • 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.
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. It can be assigned 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.</p> <hr/> <p>Note: You must have registered device users in Sybase Control Center.</p>

Option	Description
Validate controls as soon as the user tries to change focus away from them	If 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.

3. Click Finish.

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

Generated Mobile Workflow Files

When you generate the 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
- <workflow_name>.zip

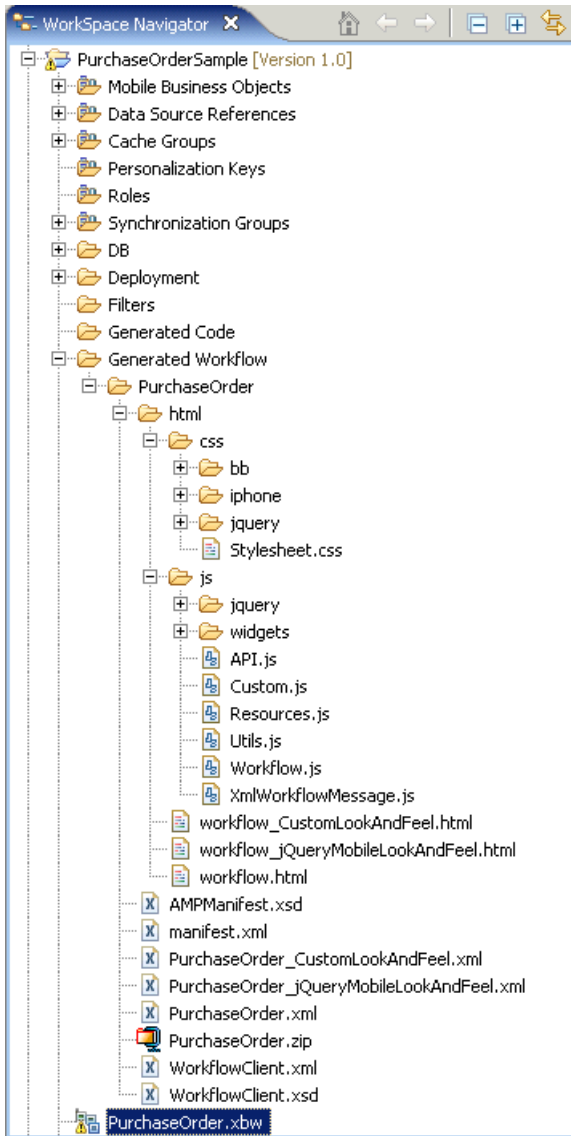
These files are regenerated only if the **Generate** option is selected in the Mobile Workflow package generation wizard:

- <workflow_name>.html
- <workflow_name>_CustomLookAndFeel.html
- <workflow_name>_jQueryMobileLookAndFeel.html
- WorkflowClient.xml
- <workflow_name>.xml
- Resources.js
- Workflow.js

These files are regenerated only if the **Generate** option is selected and the files do not exist:

- API.js
- Custom.js
- Utils.js
- WorkflowMessage.js
- all *.css files

When you choose to 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. They 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** option for the 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 re-generate the Mobile Workflow package. Examples of ways you can customize the Mobile Workflow application include the ability to:
 - Manipulate HTML elements.
 - Write code that gets called before or after generated behavior is invoked for menu items.
 - Implement custom validation logic.
- `<project_name>.zip` – contains the mobile workflow application and its files, including the images, user interface, and controls

Mobile Workflow Package Customization

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

Customizing a Mobile Workflow Package

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.

See the *Developer Guide for Mobile Workflow Packages* for information about the types of customizations that are supported and the available API methods.

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.

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**.

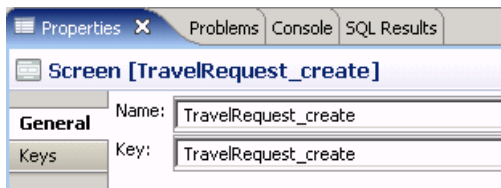
Debugging Custom Code

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. Open the browser to use for debugging and open the Java Console.
2. 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.
3. From WorkSpace Navigator, drag and drop the `workflow.html` file for the Mobile Workflow application to debug onto the browser window.
4. Find the name of the screen to debug:
 - a) In Flow Design, click the screen to debug.
 - b) In the Properties view, click **General** in the left pane.

The screen name is shown in Name, in this example, that is `TravelRequest_create`.



5. 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
```

6. 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.
7. Set up client tracing for the device user in Sybase Control Center. This is the location where the messages from the method `logToWorkflow` are written.
 - a) In Sybase Control Center, set the device debug trace level to 4 (debug).
See *Sybase Control Center for Unwired Platform* documentation for information about setting the device trace level.
 - b) Redeploy the Mobile Workflow package.
 - c) In Sybase Control Center, select **Get Trace** after using the Mobile Workflow.
The log file is written to `<UnwiredPlatform_InstallDir>\logs\<machine_name-server.log>`.

Viewing an attachment as part of the Mobile Workflow Package

You can view an attachment such as an image, a Word document, a .pdf file, and so on as part of the Mobile Workflow package.

This procedure uses an image file as an example.

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 `customBeforeMenuItemActivate`:

```

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.

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 Messaging Devices

Use Sybase Control Center to trigger the registration and device activation process, which allows messaging mobile business objects (MBOs) to handle messages belonging to different data sources.

Note: When using a Windows Mobile emulator or BlackBerry simulator to register a device 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 device from Unwired Server. Then, reregister the device. 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 emulator and BlackBerry simulator devices.

1. In the left navigation pane, click the **Device Users** node.
2. In the right administration pane, click the **Devices** tab.
3. Click **Register** to register a new device, or **Reregister** to update the device used of an existing device user.
4. In the **Register Device** or the **Reregister Device** dialog:

- a) For new device registrations only, type the name of the user that will activate and register the device. For reregistrations or clones, the same name is used and cannot be changed.
 - b) Select the name of the template for initial device registration. If you have not created any templates, only **Default** appears in the list.
The template you choose supplies initial values in the subsequent device activation fields.
5. Change the default activation field values for the template you have chosen. If you are using the default template, you must provide the server name, which is empty.
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 a device, this value cannot be changed.
 - **Activation expiration** – the number of hours the activation code is valid.
6. (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 device can be retrieved from the **Connections** group of the **Device Properties** dialog
7. 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 Device Users

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 > MyWorkFlow**.
2. In the right administration pane, click the **Devices** 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) Click **OK**.

4. To unassign a mobile workflow package, select the Activation User Name 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.

Afaria Provisioning and Mobile Device Management

Afaria® extends Unwired Platform functionality by providing additional device client management features for remote and mobile computing devices, including laptops, desktops, and handheld devices.

For information on setting up an Afaria environment, see *System Administration > Device Provisioning > Afaria Provisioning and Mobile Device Management*.

Install and Configure the Mobile Workflow Container On the Device

To enable deploying Workflow packages to a device, you must download, install, and configure a 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 Mobile Workflow container on devices for enterprise deployment. For information on setting up an Afaria environment, see *System Administration > Device Provisioning > Afaria Provisioning and Mobile Device Management*.

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\ClientAPI\Workflow\Android`.

Configuring the Android Simulator

Configure the Android simulator.

Prerequisites

Install the Android SDK and run the SDK Manager to install SDK Platform Android version 2.2 or higher, Android SDK Platform-tools, and Android SDK Tools.

Task

1. Run the Android SDK Manager and select **Virtual devices**. Click **New**, provide a name, and select 2.2 or higher for the target.
2. In the Android Simulator, start the newly created virtual device.
3. (Optional) Select **Wipe User Data**.
4. Run `platform-tools\adb` and install `SybaseDataProvider.apk` and `Workflow.apk`, which are located in `\UnwiredPlatform\ClientAPI\Workflow\Android`.

The Sybase Mobile Workflow application should now appear in the simulator.

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 *Sybase Unwired Platform System Administration > Security Administration*.

- Register each device using Sybase Control Center – Device registration pairs a user and a messaging-based synchronization (MBS) device. See *Messaging Devices* in Sybase Control Center online 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 > Systems Administration > Device and User Management > Device Provisioning > Apple Provisioning for iPhone*.

Note: APNS cannot be used on either a simulator or iTouch.

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 row being inserted, updated, or deleted in the backend and identified by a scheduled refresh of the cache
 - A data change notification request inserting, updating, or deleting rows in the cache
 - Device does not receive notifications when it is online.
 - When the device is online, the new messages are sent immediately to the device through the connection open by device.
 - When the device is offline, notifications about new messages are sent to the device through APNS.
- The device is offline (not accessible).

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

- Use the `MobileWorkflowPushDistCert.p12` located in `<Unwired_Platform_InstallDir>\UnwiredPlatform\Servers\MessagingServer\bin\` with the pre-built Workflow application that is available from the App Store, or
- 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\ClientAPI\Workflow\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\ClientAPI
\Workflow\ios\MobileWorkflow-2.0.0.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 developer documentation.
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. See the *Apple Push Notification service Programming Guide* for details.
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.
This creates a certificate request and saves it 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. Create an XCode project for the Mobile Workflow application.

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. You can change it to something unique.

Note the product name. This is used to configure the mobile workflow in Sybase Control Center. By default, the product name is Workflow.

10. 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.
Make note of the product name. This corresponds to the Application Name property in SCC.

Configuring Apple Push Settings for the Mobile Workflow Application

In Sybase Control Center, create a new Apple Push Notification Service (APNs) configuration that specifies the application, security certificate, and ports that the service uses.

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.

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**. This name corresponds to the Product Name you specified in Xcode.
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, `MobileWorkflowPushDistCert.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 – Free** version of the Workflow from the App Store, enter the password: `M0b1lEw0rkfl0w;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.

Messaging Device Apple Push Notification Properties

Apple push notification properties allow iPhone users to install messaging 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 iPhone OS 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 iPhone 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 iPhone OS system-sound facility, they must be in one of the supported audio data formats. See the iPhone 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 iPhone OS 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.

Note: If you have an existing version of the Mobile Workflow application on your device, delete it before installing a newer version.

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.

Installing 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 iOS SDK 4.3 installed.
- Xcode 3.2.6 or higher.

Task

1. From your Mac, connect to the Microsoft Windows machine where Sybase Unwired Platform is installed:
 - a) From the Apple menu, select **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-2.0.1.tar.gz` from your Sybase Unwired Platform installation `<UnwiredPlatform_InstallDir>\UnwiredPlatform\ClientAPI\Workflow\ios\` to a location on your Mac:
3. Unarchive the `MobileWorkflow-2.0.1.tar.gz`.
This creates a `Workflow` folder.
4. Copy over the libraries from `<UnwiredPlatform_InstallDir>\UnwiredPlatform\ClientAPI\MBS\ObjectiveC\libs` into the `Workflow` directory on your Mac.
5. In the `Workflow` folder, double-click **WorkFlow.xcodeproj** so that it opens in the XCode IDE.
6. If necessary, add these frameworks from the SDK to the project by selecting **Project > Edit Active Target <ProjectName> > General**:

- `Security.framework`
- `AddressBook.framework`
- `QuartzCore.framework`
- `CoreFoundation.framework`
- `libcucore.A.dylib`
- `libz.1.2.3.dylib`
- `libstdc++.dylib`

7. In XCode, select **Build > Build**.

The project builds.

Installing the Mobile Workflow Package from the Apple App Store

Install the Mobile Workflow container from the Apple App Store.

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

1. On the iOS home page, tap **App Store**.
2. Search for **Sybase**.
3. When the **Sybase Mobile Workflow – Free** container appears, tap **Free**.
4. Tap **Free** again on the Workflow information page.
5. Tap **Install** to download the application.
6. In **Settings > Sybase**, for Connection Info, enter:
 - a) In Connection Info, enter:
 - Server Name – varies depending on your environment:
 - In a single node environment, use the name of the Unwired Server.
 - In a production environment with relay servers, use the name of a deployed relay server host.
 - In a production environment with load balanced relay servers, use the name of the load balance controller. For example, if you set up an environment using the Microsoft Web Platform Installer, this would be the name of the Application Request Router server.
 - Server Port – the port used by Unwired Server, or in a relay server-enabled environment, the port of the relay server. You can use either HTTP or HTTPS ports for these connections. Contact the platform administrator for production values you may need. For example, the default relay server HTTP port is 80 and the default HTTPS port is 443.
 - Farm ID – the company ID, or Farm ID registered in the relay server. If you are not using relay server, use a value of 0.
 - User name – the user name you entered for the device in Sybase Control Center.
 - Activation Code – the activation code for the device.

7. Scroll to the page that contains the **Sybase** icon, then tap to launch.
8. 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.
 - (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.

9. Click the **Messages** tab bar, then tap **Messages** to view the Workflows.
10. (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**.

Configuring iOS Device Connection Settings

Configure the connection settings for the Mobile Workflow application.


1. Go to your device Settings 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.
 - Company\Farm ID – the company or farm ID you entered when you registered the device in Sybase Control Center, in this case, 0 (zero).
 - User Name – the user you registered in Sybase Control Center.
 - Activation Code – the activation code for the user, for example, 123.
3. Start the Mobile Workflow application, then tap **Settings > General > Connection Information** to see if the connection is active.

The Settings log indicates if you are successfully connected to the server.

Preparing Windows Mobile Devices for the Mobile Workflow Package

Install and configure the container required to prepare a Windows Mobile device to run Mobile Workflow packages.

1. Navigate to <UnwiredPlatform_InstallDir>\UnwiredPlatform\ClientAPI\Workflow\WM.
2. Copy the Windows Mobile Professional device file, SUPMessaging_Pro.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 .cab file from the Windows Mobile device. This installs the container.
6. In the 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 – varies depending on your environment:
 - In a single node environment, use the name of the Unwired Server.
 - In a production environment with relay servers, use the name of a deployed relay server host.
 - In a production environment with load balanced relay servers, use the name of the load balance controller. For example, if you set up an environment using the Microsoft Web Platform Installer, this would be the name of the Application Request Router server.
 - Server Port – the port used by Unwired Server, or in a relay server-enabled environment, the port of the relay server. You can use either HTTP or HTTPS ports for these connections. Contact the platform administrator for production values you may need. For example, the default relay server HTTP port is 80 and the default HTTPS port is 443.
 - Farm ID –the company ID, or Farm ID registered in the relay server. If you are not using relay server, use a value of 0.
 - User Name – the user name you entered for the device in Sybase Control Center.
 - Activation Code – the activation code for the device.

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

7. Access your Microsoft Outlook inbox on the device. For example:
 - On Windows Mobile Touch Screen devices, tap **Start > Messaging**, and then scroll to and tap Outlook Inbox.
 - On Windows Mobile Non-touch Screen devices, click **Start > Messaging** on the Home screen. Scroll to a Outlook Inbox and press **ENTER**.



8. Select **Menu > Workflows**.



9. Select a 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.

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 container files, <UnwiredPlatform_InstallDir> \UnwiredPlatform\ClientAPI\Workflow\BB.
6. Select the files and click **Open**.
The application is listed on the Application Loader wizard.
7. Click **Next**.
8. Click **Finish**.
9. Restart your BlackBerry device.

Installing the 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.

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. Navigate to <UnwiredPlatform_InstallDir>\UnwiredPlatform \ClientAPI\Workflow\BB.
2. Copy the required .cod files to the BlackBerry Simulator directory. These include:
 - CommonClient.cod – shared code that can be used by native Sybase Unwired Platform BlackBerry applications.
 - MessagingClientApp.cod – the main Mobile Workflow application.
 - MessagingClientSettings.cod – Sybase Settings options screen where the user enters the server connection information.
 - MocaClient.cod – messaging library.

- `WorkflowResources.cod` – image resources for the Mobile Workflow application, for example, the Mobile Workflow icons.

3. Start the BlackBerry Simulator.

Installing Certificates and Testing on Simulators and Devices

Install and test certificates on various types of simulators and devices.

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

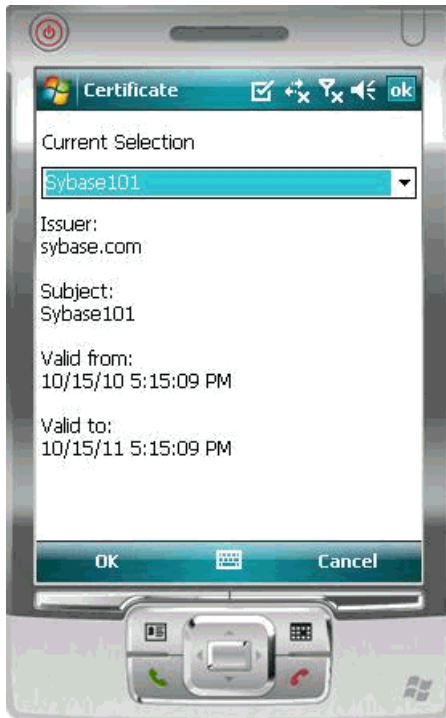
Select an X.509 certificate to use for Mobile Workflow application user authentication.

Prerequisites

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 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

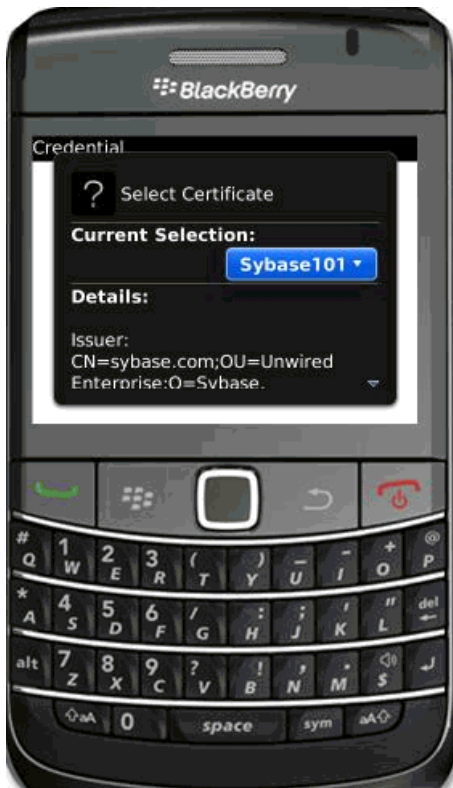
Select an X.509 certificate to use for workflow application user authentication.

Prerequisites

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 and Testing X.509 Certificates on iOS Clients

Install generated X.509 certificates and test them in your iOS clients.

Adding an Authentication Screen for iPhone Workflow Clients using an X.509 Certificate

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.
6. During device application development, define and add a screen that has a Certificate Picker menu item.
7. Generate and deploy the application to the iPhone client.
8. Select **Certificate Picker** from the iPhone client.
9. Enter Windows credentials and certificate password in the dialog and click **Done**. Make sure the format is `domain\user`.
10. Submit credentials to Unwired Server.

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.

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**.

Testing Mobile Workflow Packages

To test a Mobile Workflow package:

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.

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.

Reference

This section describes the generated files and the Workflow client API.

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. 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.
<code>customAfterWorkflowLoad()</code>	Invoked when the application is first launched, after data is loaded and screens are opened.
<code>customBeforeSubmit(screenKey, actionName, workflowMessageToSend)</code>	Invoked before an operation or object query is about to be called as the result of the user clicking a Submit menuitem. User can return false to prevent the default behaviour from occurring.
<code>customAfterSubmit(screenKey, actionName)</code>	Invoked after an operation or object query is called as the result of the user clicking a Submit menuitem.
<code>customBeforeNavigateBackward(screenKey, isCancelled)</code>	Invoked when another screen is about to be opened. User can return false to prevent the screen from being opened.
<code>customAfterNavigateForward(screenKey, destScreenKey)</code>	Invoked after another screen has been opened.
<code>customBeforeNavigateBackward(screenKey, isCancelled)</code>	Invoked when a screen is about to be closed. User can return false to prevent the screen from being closed.

Reference

Method	Description
customAfterNavigateBackward(screenKey, isCancelled)	Invoked after a screen has been closed.
customBeforeShowScreen(screenToShow, screenToHide)	Invoked when a screen is about to be shown. User can return false to prevent the screen from being shown.
customAfterShowScreen(screenToShow, screenToHide)	Invoked after a screen has been shown.
customValidateScreen(screenKey, values)	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.
customBeforeMenuItemClick(screen, menuItem)	Invoked after a menuitem has been clicked. User can return false to prevent the default behaviour (which might open a new screen, or perform a submit, and so on) from occurring.
customAfterMenuItemClick(screen, menuItem)	Invoked after a menuitem has been clicked and the default behavior has occurred.
customBeforeSave(screenKey)	Invoked before a screen's contents are about to be persisted to the Mobile Workflow message. User can return false to prevent the default behaviour from occurring.
customAfterSave(screenKey)	Invoked after a screen's contents have been persisted to the Mobile Workflow message through the default logic.

```
//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;
}

```

Using Third-party JavaScript Files

To load external JavaScript files dynamically:

Copy the relevant third-party JavaScript files to the Generated Workflow \<Workflow_package_name>\html and js folders. They will be included in the Mobile Workflow manifest and .zip file automatically.

Generated HTML Files

The Mobile Workflow Forms Editor generates these HTML files:

- 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** option for the 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.

CSS Files

On BlackBerry 6.0 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.

CSS files include:

- jquery.mobile-1.0a3.css – located in Generated Workflow \<mobile_workflow_name>\html\css\jquery folder and used on BlackBerry 6.0 and iOS platforms. By default, pages are generated using the A 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 modify .
- master.css – located in Generated Workflow \<mobile_workflow_name>\html\css\bb and used on the BlackBerry 5.0 platform. 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. To modify the background colour 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 as follows:

- 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.

```
<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 menuitem:

```
<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.

```
<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_Dept_name_paramKey_help"
class="help"></span>
      </div>
```

The first <div> is 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 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 will be 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>
    </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

Reference

divs in the form will 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>.

- Each screen has a block, contained in a <div> element, and each <div> has a form, where the controls are hosted.
- The first <div> is 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 will 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 look-and-feel is simple and can be used on all platforms, including Windows 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 as follows:

- Each screen has a block, a <div> element. Each of those <div>s 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, <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 put aside for use 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 will 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, <label> and the second holding 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">
      <ul id="Department_createScreenDivMenu" class="menu">
        <li><a class="nav" name="Create"
onclick="menuItemCallbackDepartment_createSubmit_Workflow();" >Creat
e</a></li>
          <li><a class="nav" name="Cancel "
onclick="menuItemCallbackDepartment_createCancel();" >Cancel</a></
li>
        </ul>
        <form name="Department_createForm"
id="Department_createForm">
          <table class="screen">
            <tr>
              <td colspan="2"><span id="Department_createForm_help"
class="help"></span></td>
            </tr>
            <tr>
              <td colspan="2" id="topOfDepartment_createForm"></td>
            </tr>
            <tr>
              <td class="left"><label
for="Department_create_dept_name_paramKey">Dept name:</label></td>
              <td class="right"><input class="right" type="text"
id="Department_create_dept_name_paramKey"/><span
id="Department_create_Department_create_dept_name_paramKey_help"
class="help"></span></td>
            </tr>
            <tr><td colspan="2" id="bottomOfDepartment_createForm"></
td></tr></table>
          </form>
        </div>
      </body>
</html>
```

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.

General Utility Functions

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

All of general utility functions are synchronous.

Method	Description
guid()	Generates a unique string.
trimSpaces(str)	Removes spaces from the specified string. str – the specified string.
escapeValue(val)	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;'; val – the specified value. Returns the escaped value.
unescapeValue(val)	Replaces all instances in the specified string: <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 "'. val – the specified value. Returns the escaped value.
isIOS()	Returns true if the Mobile Workflow application is running on an iOS platform.
isBlackBerry()	Returns true if the Mobile Workflow application is running on a BlackBerry platform.

Method	Description
isWindowsMobile()	Returns true if the Mobile Workflow application is running on a Windows Mobile platform.
isWindows()	Returns true if the Mobile Workflow application is running on a Windows platform.
getAttribute(elem, attribName)	Reliably returns the specified attribute value for the specified HTML element. <ul style="list-style-type: none"> elem – the specified HTML element. attribName – the attribute name.
getElementsByTagName(El, 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"> el – the specified HTML element. tagName – the specified tag name.
getFormElementById(formEl, elemID)	Returns the form element with the specified ID. <ul style="list-style-type: none"> formEl – the form element. elemId – 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
getDateString(date)	Returns a string representation of the specified date (currently in yyyy-mm-dd format only). date – the specified date.

Reference

Method	Description
getDateTimeStringToDisplay(datetime, precision)	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.
getDateTimeStringForXMLMessage(datetime)	Returns a string representation of the specified date in yyyy-mm-ddThh:mm:ss format. datetime – the specified datetime.
getDateFromExpression(toolingStr)	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. toolingStr – the date as specified in the Mobile Workflow Forms editor.
parseBoolean(value)	Returns true if the specified string is equal, in a case-insensitive way, to true.
parseDateTime(value)	Returns a date that corresponds to the specified string.
convertToSUPType(htmlElement, typeAttribute)	Returns the XmlWorkflowMessage type for the given HTML element. <ul style="list-style-type: none"> • htmlElement – the specified HTML element. • typeAttribute – the type of the specified HTML element.
getHTMLValue(htmlElement, typeAttribute)	Returns a string representation of the specified HTML element’s value. <ul style="list-style-type: none"> • htmlElement – the specified HTML element. • typeAttribute – the type of the specified HTML element.

Method	Description
setHTMLValue(htmlElement, value, screenName)	<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.

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.
getPreviousScreen()	Returns the key of the screen that was open previous to the current screen being opened, if applicable.
navigateForward(toScreen, listViewKey)	<p>Navigates from the current (open) screen to a new screen with the specified key.</p> <ul style="list-style-type: none"> toScreen – the screen to open. (optional) listViewKey – the listview row for which the details screen is being opened.
navigateBack(isCancelled)	<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>isCancelled – true for a Cancel action, false for a Save action.</p>

Reference

Method	Description
<p>updateUIFromMessageValueCollection(screenKeyName, values)</p>	<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>screenKeyName</code> – the screen. • <code>values</code> – the message value collection.
<p>updateMessageValueCollectionFromUI(values, screenName, keys, keyTypes)</p>	<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.
<p>saveScreen(currMVC, currScreen, needValidation)</p>	<p>Saves the contents of the specified screen to the specified <code>MessageValueCollection</code>. This function differs from <code>updateMessageValueCollection</code> 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"> • <code>currMVC</code> – the current message value collection • <code>currScreen</code> – the current screen. • (optional) <code>needsValidation</code> – 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.

Method	Description
saveScreens()	Saves the contents of all open screens if they are successfully validated.

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'>";
                    }
                }
            }
        }
    }
}
```

Reference

```
        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 Application Native Device Functions

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

Method	Description	Synchronous or Asynchronous
setScreenTitle(screenKeyName, screenTitle)	<p>Sets the specified screen's title based on its sup_screen_title attribute value.</p> <ul style="list-style-type: none"> screenKeyName – the screen. (optional) screenTitle – an explicit screen title to use rather than the sup_screen_title attribute value. 	<p>Asynchronous on iOS</p> <p>Synchronous on BlackBerry and Windows Mobile</p>
closeWorkflow()	Closes the Mobile Workflow application.	<p>Asynchronous on iOS</p> <p>Synchronous on BlackBerry and Windows Mobile</p>
addMenuItem(menuItemName, functionName, subMenuName)	<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. 	Synchronous
removeAllMenuItems()	Removes all native menu items.	<p>Asynchronous on iOS</p> <p>Synchronous on BlackBerry and Windows Mobile</p>

Reference

Method	Description	Synchronous or Asynchronous
clearCache()	Allows the user to clear the contents of the on-device request result cache for the current workflow.	Synchronous
clearCacheItem(cacheKey)	<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>cacheKey – the key for the item to be removed. See the <code>Workflow.js</code> file for details on how to construct this.</p>	Synchronous
logToWorkflow(message, level, notifyUser)	<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"> • message – message to log. • level – level at which to log (ERROR, WARN, INFO, or DEBUG). • notifyUser – if true, an alert dialog is shown before logging commences. 	Synchronous
showCertificatePicker()	Opens a form on the device that allows the user to specify the credentials through the use of certificate-based authentication.	<p>Asynchronous on iOS</p> <p>Synchronous on BlackBerry and Windows Mobile</p>

Method	Description	Synchronous or Asynchronous
showUrlInBrowser(url)	<p>Open the supplied URL in the browser. To have a hyperlink in the default value for the HtmlView control, or for doing customization in Javascript, follow showUrlInBrowser without using standard HTML.</p> <p>For example, to add html in the default value for the HtmlView control, you can use something similar to:</p> <pre data-bbox="565 591 999 1100"> <html> <body> Welcome to Sybase Mobile Workflow

Your activation was successful, the newly created Workflow requests will automatically be pushed to you.

For more information contact your administrator or visit us at:

 The Unwired Enterprise </body> </html> </pre>	<p>Asynchronous on iOS</p> <p>Synchronous on BlackBerry and Windows Mobile</p>
showAttachmentContents(contents, mimeType, fileName)	<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 data-bbox="565 1350 999 1541" style="list-style-type: none"> • contents – the Base64-encoded version of the binary content of the attachment to show. • mimeType – (optional) the MIME type of the file. • fileName – (optional) the name of the file. 	<p>Synchronous on Windows Mobile</p> <p>Asynchronous on iOS and BlackBerry</p>

Reference

Method	Description	Synchronous or Asynchronous
showAttachmentFromCache(uniqueKey, mimeType, fileName)	<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"> • uniqueKey – the unique key for the attachment. • (optional) mimeType – the MIME type of the file. • (optional) fileName – the name of the file. 	<p>Synchronous on Windows Mobile</p> <p>Asynchronous on iOS and BlackBerry</p>
showLocalAttachment(key)	<p>Shows a local attachment.</p> <p>key – the key.</p>	<p>Asynchronous on iOS</p> <p>Synchronous on BlackBerry and Windows Mobile</p>

Method	Description	Synchronous or Asynchronous
doOnlineRequest(screenKey, requestAction, timeout, cacheTimeout, errorMessage, errorCallback, workflowMessageToSend, cacheKey)	<p>Allows the user to cause an operation/object query to be invoked.</p> <ul style="list-style-type: none"> • screenKey – the specified screen on which the submit is occurring. • requestAction – the specified action for the submit. • timeout – specifies the time, in seconds, to wait for a response. • cacheTimeout – 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. • errorMessage – specifies the string to show if an online request fails. • errorCallback – name of the function to be called if an online request fails. • workflowMessageToSend – Mobile Workflow message that is sent as the input in an online request. • cacheKey – string used as the key for this request in the on-device request result cache. 	<p>Synchronous on Windows Mobile</p> <p>Asynchronous on iOS and BlackBerry</p>

Reference

Method	Description	Synchronous or Asynchronous
doAttachmentDownload(screenKey, requestAction, workflowMessageToSend, attachmentKey, cacheGUID, cacheCallback)	<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. • cacheGUID – 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. • cacheCallback – 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. 	Asynchronous
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. 	Synchronous

Method	Description	Synchronous or Asynchronous
showAlertDialog	Brings up a message dialog with the specified message, or, optionally, on iOS only, the specified title.	Asynchronous

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()</code>	<p>Recommended for listviews so that the user gets the appropriate part, or values, of the message, for the current screen.</p> <p>Each row in the listview corresponds to a <code>Values</code> 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.</p>
<code>getWorkflowMessage()</code>	<p>Allows the user access to the Mobile Workflow message.</p> <hr/> <p>Note: Sybase recommends this method, generally, over <code>getCurrentMessageValueCollection()</code>.</p>

Reference

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")
                                .getKey()
                                    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;
}

```

WorkflowMessage.js File

You have full access to the workflow message being sent to and from the client, through customization.

The workflow message is presented to the unit in an object-based format. At the top level, you have access to a `WorkflowMessage` object, constructed with an XML-encoded string.

WorkflowMessage object properties and methods

The `WorkflowMessage` object has these properties and methods:

- Properties
 - header (string)
 - requestAction (string)
 - workflowScreen (string)
 - Values (MessageValueCollection)
- Methods

Reference

- `createFromstring` (`workflowMessageAsString`)
- `parseMessageValueCollection` (`valuesNode`, `messageValueCollection`)
- `serializeToString`()
- `serializeValues` (`values`, `prefix`)

MessageValueCollection properties and methods

The `MessageValueCollection` object has these properties and methods:

- Properties
 - `key` (string)
 - `state` (string)
- Methods
 - `add` (`key`, `value`)
 - `clear` ()
 - `getData` (`key`)
 - `remove` (`key`), where the `key` parameter is a string and the `value` parameter is either a `MessageValueCollection` or a `MessageValue`

MessageValue object properties and methods

The `MessageValue` object has these properties:

- `key` (string)
- `type` (string) – one of `BOOLEAN`, `NUMBER`, `DATETIME`, `TEXT`, or `LIST`
- `value` (string)

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>validateRegularExpression(val, rExp, userSuppliedMsg, helpElement)</code>	Returns true if the specified value matches the specified regular expression. <ul style="list-style-type: none">• <code>val</code> – the specified value.• <code>rExp</code> – the specified regular expression.• (Optional) <code>userSuppliedMsg</code> – the message to use if the specified value is invalid.• (Optional) <code>helpElement</code> – the help element.

Method	Description
validateDate(val, required, minValue, maxValue, userSuppliedMsg, helpElement)	<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"> • val – 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.
validateDateTime(val, required, minValue, maxValue, userSuppliedMsg, helpElement)	<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"> • val – 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
<p>validateTime(val, required, minValue, maxValue, userSuppliedMsg, helpElement)</p>	<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"> • val – 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.
<p>validateNumber(val, required, minValue, maxValue, numofDecimals, maxLength, userSuppliedMsg, helpElement)</p>	<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"> • val – 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.

Method	Description
<code>validateText(val, required, maxLength, userSuppliedMsg, helpElement)</code>	<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"> • <code>val</code> – the specified value. • (Optional) <code>required</code> – true if the value must be specified, otherwise, false. • (Optional) <code>maxLength</code> – the maximum number of characters. • (Optional) <code>userSuppliedMsg</code> – the message to use if the specified value is invalid. • (Optional) <code>helpElement</code> – the help element.
<code>validateEmail(val, 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>val</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)</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>validateAllScreens()</code>	<p>Validates all open screens.</p>

Resource Functions

The resource functions allow you to access localized string resources.

Method	Description
<code>resources.hasLocale(localeName)</code>	Returns true if the locale supplied is included in the Mobile Workflow application; otherwise, returns false.
<code>resources.getString(key)</code>	Returns the localized string for the supplied key for the current locale.
<code>resources.getString(key, localeName)</code>	Returns the localized string for the supplied key for the specified locale.

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.

The following is a list of recoverable and non-recoverable error codes. Beginning with Unwired Platform version 1.5.5, all error codes that are not explicitly considered recoverable are now considered unrecoverable.

Table 2. Recoverable Error Codes

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

Table 3. 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.

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

Beginning with Unwired Platform version 1.5.5, error code 401 is no longer treated as a simple recoverable error. If the `SupThrowCredentialRequestOn401Error` context variable is set to true (which is the default), error code 401 throws a `CredentialRequestException`, which sends a credential request notification to the user's inbox. You can change this default 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. 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 4. 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. No retry exception is thrown for these errors.

Table 5. 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 Un-wired 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

The following is a list of SAP® error codes mapped to HTTP error codes. SAP error codes which are not listed map by default to HTTP error code 500.

Table 6. 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 inavailability of the remote SAP system.	503
JCO_ERROR_LOGON_FAILURE	Authorization failures during the logon phase usually caused by unknown username, wrong password, or invalid certificates.	401
JCO_ERROR_RESOURCE	Indicates that JCO has run out of resources such as connections in a connection pool	503

Constant	Description	HTTP Error Code
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

Index

A

- activating devices 89
- Alert Message property 97
- Alerts property 97
- Android 93
- APNS Device Token property 97
- App Store 99
- Apple push notification properties 97
- Apple push notification, configuring 96
- AttachmentViewer 56
- AttachmentViewer limitations 56

B

- Badges property 97
- BlackBerry 105
- BlackBerry Desktop Manager 104

C

- cached data lookup pattern
 - data flow diagram 23
 - overview 23
- certificate picker 64
- client trace logs 144
- controls 42
- convertToSUPTYPE() 121
- creating keys 40
- Credentials
 - dynamic 63
 - static 63
- Custom.js 86
- custom.js file
 - methods 113
- customAfterNavigateForward 113
- customAfterShowScreen 113
- customAfterSubmit 113
- customAfterWorkflowLoad 113
- customBeforeMenuItemActivate 113
- customBeforeNavigateBackward 113
- customBeforeNavigateForward 113
- customBeforeShowScreen 113
- customBeforeSubmit 113
- customBeforeWorkflowLoad 113

- customValidateScreen 113

D

- data change notification 61
 - filter class 62
 - GET 57
 - HTTP authentication 61
 - JSON format 57
 - notification development life cycle 60
 - POST 57
 - request response 62
- DCN 60, 62
- default locale, creating 73
- defining an MBO
 - for cached data lookup 23
 - for real-time data lookup 17
- Delivery Threshold property 97
- deploy mode 76
- deploying
 - mobile application project 76
- device platforms 31
- device users
 - assigning mobile workflow packages 91
- documentation roadmap
 - document descriptions 1
- Dynamic authentication 65

E

- editing
 - locale properties file 76
- EIS error codes 141–143
- Enable property 97
- error codes
 - EIS 141–143
 - HTTP 141–143
 - mapping of SAP error codes 143
 - non-recoverable 141, 142
 - recoverable 141, 142

F

- findByParameter
 - binding to a Workflow menu item 19

Index

findByParameter object query 11

functions

 general utility 120

 resource 140

 workflow UI 123

G

generate old key values 52

generated files 84

 workflow_customlookandfeel.html 115

 workflow_jquerymobilelookandfeel.html 115

 workflow.html 115

getCurrentMessageValueCollection() 133

getDateFromExpression() 121

getDateString() 121

getTimeStringForXMLMessage() 121

getTimeStringToDisplay() 121

getHTMLValue() 121

getWorkflowMessage() 133

H

HTTP error codes 141–143

I

installing 97, 99

internationalization

 Mobile Workflow Forms editor 79

 on the device 81

iPhone push notification properties 97

iTunes 100

J

jquery.mobile-1.0a3.css 115

K

key collection menu item 52

Keys 41

keys, creating 40

L

locale

 editing 76

 properties file 76

locale properties file

 creating 73

 validating 73

localization 72

 creating a new locale 73

 device settings 77

 limitations 72

 Mobile Workflow package 73

 task flow 73

Localization

 current locale 79

 updating the current locale 79

M

manage 89

manually adding a starting point 34

master.css 115

matching rules

 specifying 13

matching rules, adding 40

menu item 43

 properties 44

 types 44

menu item, adding 44

menu item, creating 47

menu items

 online request 50

message-based application 76

messaging device registration 89

messaging device setup 89

messaging devices

 Apple push notification properties 97

methods

 utility 121

 validation 136

mobile application project, deploying 76

mobile workflow

 client trace logs 144

 exception handling 144

 unblocking 144

Mobile Workflow Application Generation wizard

 82

Mobile Workflow Forms Editor 31

 preferences 32

 preferences, Mobile Workflow Forms Editor

 32

 setting preferences 32

- mobile workflow package
 - generated files 84
- mobile workflow packages
 - assigning device users 91
 - configuring notification mailbox 90

N

- native device functions 127
- non HTTP authentication request 61
- notification extraction rules 38
- notification extraction rules, editing 39
- notification mailbox 90
- notification, matching rules 40

O

- object queries
 - binding to a workflow menu item 25
- object query parameters
 - defining a control that passes 26
- online request menu item 50
- optimized for performance 118
- OTA 105
- over the air 105

P

- parameter mappings 53
- parseBoolean() 121
- parseDateTime() 121
- properties
 - push notification for iPhones 97
 - server-initiated 36
- PurchaseOrderSample 73
- push notification properties for iPhones 97

R

- real-time lookup pattern
 - data flow diagram 17
 - overview 17
- regional settings 77
- registering messaging devices 89
- resource functions 140

S

- sending server notification to a device 15

- server notification pattern
 - creating an MBO for 11
 - data flow diagram 11
 - overview 11
- server-driven notification
 - creating 13
- server-initiated properties 36
- server-initiated starting point 34
- setHTMLValue() 121
- settings
 - device 77
 - iPhone 77
 - regional 77
 - Windows Mobile 77
- Simulator 93
- single sign-on
 - using credentials 66
 - using SSO2 tokens 68
 - using static SSO2 tokens 70
 - using static X.509 certificates 67
- Sounds property 97
- starting point
 - adding manually 34
- starting points
 - Activate 33
 - Client-initiated 33
 - Credential request 33
 - editing 40
 - server-initiated 34, 40
 - Server-initiated 33
- static authentication 64
- stylesheet.css 115
- submit menu item 48
- SUPMobileWorkflow.replaceMobileWorkflowCertificate() 71
- supported attachment types 56
- Sybase Mobile Workflow 93

T

- tutorial
 - configuring the Android simulator 93

U

- Unwired Server logs 144

Index

V

- validateAllScreens 136
- validateControl 136
- validateDate 136
- validateDateTime 136
- validateEmail 136
- validateNumber 136
- validateRegularExpression 136
- validateScreen 136
- validateText 136
- validateTime 136

W

- workflow client
 - using credentials 66

- workflow clients
 - and static SSO2 tokens 70
 - and static X.509 certificates 67
 - using credentials in 66
 - using SSO2 tokens in 68
- Workflow control
 - that passes object query parameters 20
- Workflow screen
 - that displays results 20
- workflow.html 116
- workflow.html file 115