SYBASE®
An SAP® Company

Tutorial: iOS OData Application Development with REST Services

Sybase Unwired Platform 2.2 SP03

DOCUMENT ID: DC01976-01-0223-01
LAST REVISED: April 2013

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

Contents

Sybase Unwired Platform Tutorials

The Sybase® tutorials demonstrate how to develop, deploy, and test mobile business objects, device applications, online mobile applications (native OData and REST services based), and Hybrid App packages. You can also use the tutorials to demonstrate system functionality and train users.

- Learn mobile business object (MBO) basics, and use this tutorial as a foundation for the Object API application development tutorials:
  - *Tutorial: Mobile Business Object Development*

  Note: For all Object API tutorials, if you opt to use the Mobile Business Object example project instead of performing the Mobile Business Object Tutorial, you must deploy the mobile application project to Unwired Server as a prerequisite.
- Create native Object API mobile device applications:
  - *Tutorial: Android Object API Application Development*
  - *Tutorial: BlackBerry Object API Application Development*
  - *Tutorial: iOS Object API Application Development*
  - *Tutorial: Windows Object API Application Development*
  - *Tutorial: Windows Mobile Object API Application Development*
- Create a mobile business object, then develop a hybrid app package that uses it:
  - *Tutorial: Hybrid App Package Development*
- Create an OData mobile application with REST Services
  - *Tutorial: Android OData Application Development with REST Services*
  - *Tutorial: iOS OData Application Development with REST Services*

Getting Started with Unwired Platform (On-Premise)

Install and learn about Sybase Unwired Platform and its associated components.

Complete the following tasks for all tutorials, but you need to perform them only once.

1. *Installing Sybase Unwired Platform*

   (Applicable to On-Premise version) Install Sybase Mobile SDK and Unwired Platform Runtime.
2. *Starting Sybase Unwired Platform Services*

   Start Unwired Server, Sybase Control Center, the sample database, the cache database (CDB), and other essential services.
3. *Connecting to Sybase Control Center*

   Open Sybase Control Center to manage Unwired Server and its components.
4. *Creating a Security Configuration for a Domain*

   Create a security configuration using Sybase Control Center, then map it to the desired domain.
5. *Creating an Application ID and Whitelisting the Application Endpoint*

   Create a new application using Sybase Control Center

Installing Sybase Unwired Platform

(Applicable to On-Premise version) Install Sybase Mobile SDK and Unwired Platform Runtime.

Before starting this tutorial, install all the requisite Unwired Platform components. See the Sybase Unwired Platform documentation at *http://sybooks.sybase.com/sybooks/sybooks.xhtml?id=1289&c=firsttab&a=0&p=categories*:

- *Release Bulletin*
- *Installation Guide for Sybase Mobile SDK*
- *Installation Guide for Runtime*

1. Install these Unwired Platform Runtime components:
   - Data Tier (included with single-server installation)
   - Unwired Server
2. Install Sybase Mobile SDK.

Starting Sybase Unwired Platform Services

Start Unwired Server, Sybase Control Center, the sample database, the cache database (CDB), and other essential services.

The way in which you start Unwired Platform Services depends on the options you selected during installation. You may need to manually start Unwired Platform Services.
Select **Start > (All) Programs > Sybase > Unwired Platform > Start Unwired Platform Services**.
The following services will be started:

- Sybase Control Center *<Version>*
- Sybase Unwired Cache DB
- Sybase Unwired SampleDB
- Sybase Unwired Server

Unwired Platform Services enable you to access the Unwired Platform runtime components and resources.

Connecting to Sybase Control Center

Open Sybase Control Center to manage Unwired Server and its components.

From Sybase Control Center, you can:

- View servers and their status
- Start and stop a server
- View server logs
- Deploy a mobile application package
- Register application connections
- Set role mappings
- Assign/Unassign a hybrid application to a device

For information on configuring, managing, and monitoring Unwired Server, click **Help > Help Contents**.

1. Select **Start > (All) Programs > Sybase > Sybase Control Center**.

   Note: If Sybase Control Center does not launch, make sure that the Sybase Control Center service is started in the Windows Services dialog.

2. Log in by entering the credentials set during installation.

   Sybase Control Center gives you access to the Unwired Platform administration features that you are authorized to use.

Creating a Security Configuration for a Domain

Create a security configuration using Sybase Control Center, then map it to the desired domain.

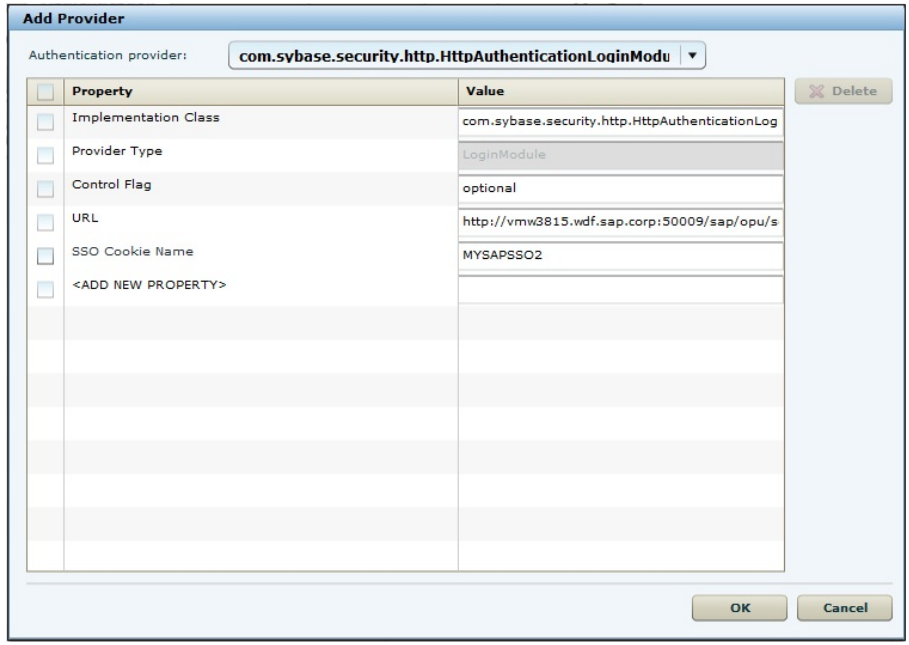
Prerequisites

Connect to Sybase Control Center.

Task

1. Log in to Sybase Control Center using the credentials you indicated during installation.
2. In the right pane, under **General** tab, click **New...**.
3. In **Create Security Configuration** dialog box, enter `SSO` as security configuration name.

4. Click **OK**.
   SSO is created as desired security configuration in left navigation pane under **Security** node.
5. In Sybase Control Center, select **View** > **Select** > **Mobile Server Cluster Management View**.
6. In the left navigation pane, select **Domains -> default -> Security** folder and click **Assign**.
7. Select **SSO**. In the right pane, under **Authentication** click **New...**.
8. In **Edit Provider** dialog box:
   a) Select the required loginModule in authentication provider from the drop-down list.
   b) Enter the authentication provider URL in **URL** field as `http://vmw3815.wdf.sap.corp:50009/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/`
   c) Click **Save**.



9. Under **Configuration authentication properties**, select and delete the default provider type: **NoSecLoginModule**.

   Similarly, under **Authorization** and **Attribution** tabs, delete the default provider types: **NoSecAuthorizer** and **NoSecAtributer** respectively.
10. Under **General** tab, click **Validate** to validate the configuration before applying the changes to the Unwired Server.
11. Click **Apply**.

Next

In Sybase Control Center, create the application ID.

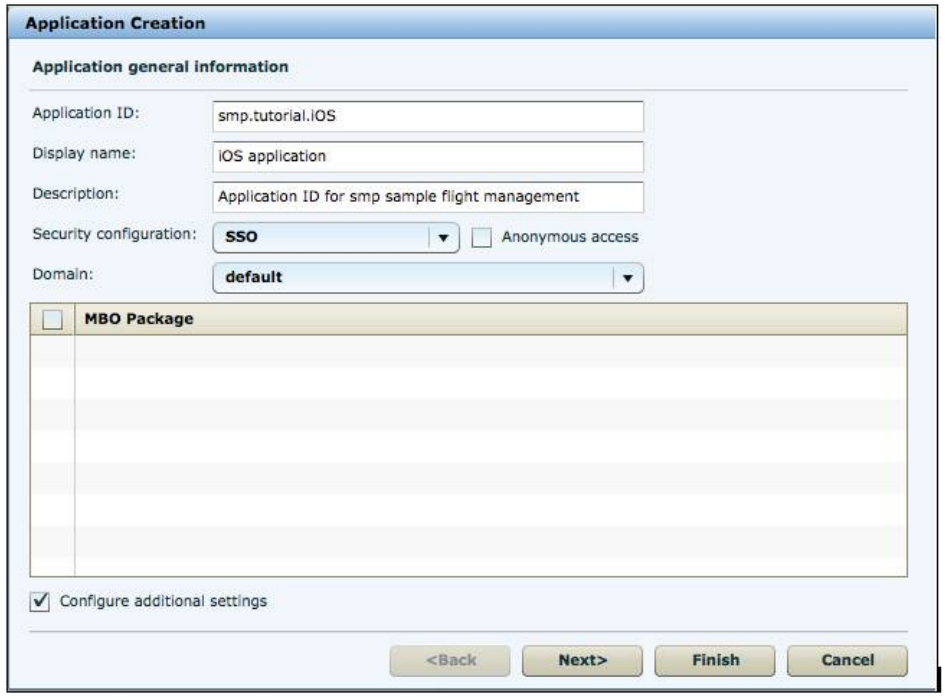
Creating an Application ID and Whitelisting the Application Endpoint

Create a new application using Sybase Control Center

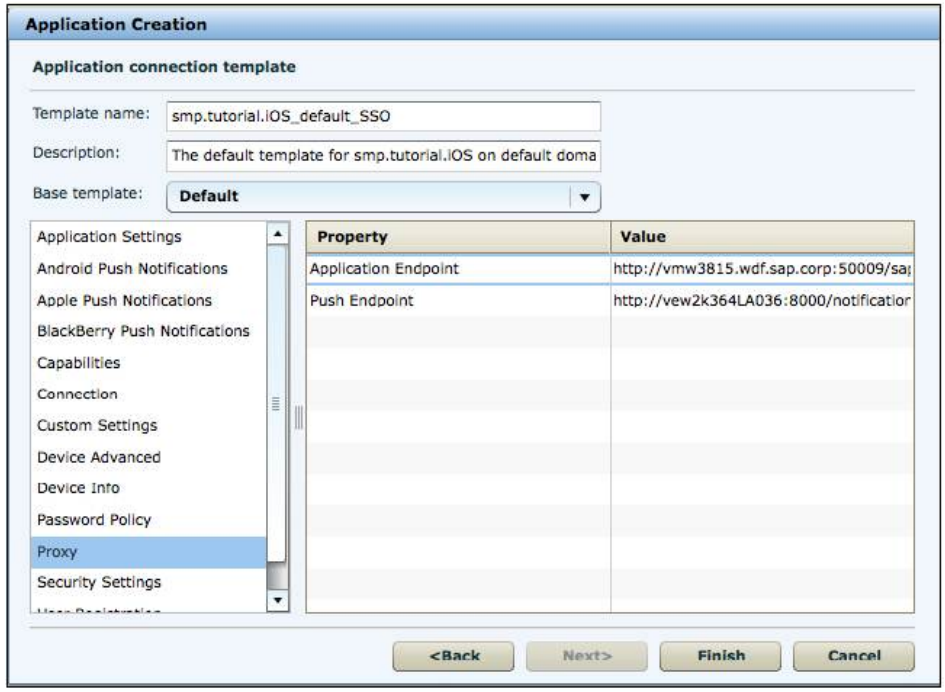
1. In the left navigation pane of Sybase Control Center, click the **Applications** node and select the **Applications** tab in the right administration pane.
2. Click **New...**.
3. In the **Application Creation** dialog box, enter the required information:
   - **Application ID** - `smp.tutorial.iOS`

- **Display name** - `iOS application`
- **Description** - `Application ID for SMP sample flight management application`
- Select **Security configuration** - `SSO`
- Select **Domain** - `default`

4. Enable **Configure additional settings** checkbox.



5. Click **Next**.
6. Under **Application connection template**, select **Proxy** from the list.
7. Enter the **Application Endpoint** as `http://vmw3815.wdf.sap.corp:50009/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/`



8. Click **Finish** to register the application with the configured settings.
   With the end of this procedure you have created the application ID and proxy connection (whitelisting of authentication endpoint URL).

Getting Started with SAP Mobile Platform Cloud

Install and learn about SAP® Mobile Platform and its associated components.

Complete the following tasks for all tutorials, but you need to perform them only once.

1. *Connecting to SAP Mobile Platform, enterprise edition, cloud version - Administration and Monitoring*

   Execute the tasks listed in this section prior to configuring mobile application using SAP Mobile Platform, enterprise edition, cloud version - Administration and Monitoring.
2. *Creating an Application*

   Create a new application using SAP Mobile Platform, enterprise edition, cloud version - Administration and Monitoring
3. *Creating Application Endpoint URL*

   Create the enterprise information system (EIS) or backend connection.
4. *Creating a Security Profile*

   Create a new security profile and store the application settings.

Connecting to SAP Mobile Platform, enterprise edition, cloud version - Administration and Monitoring

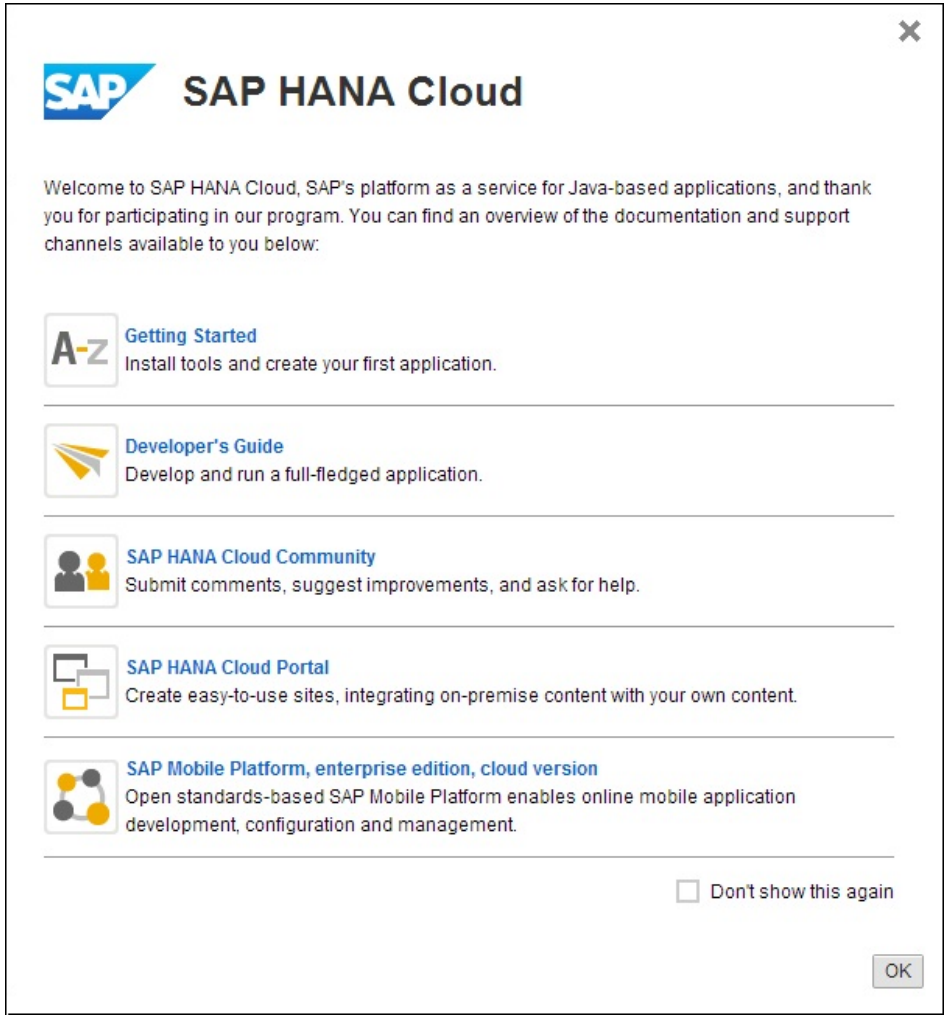
Execute the tasks listed in this section prior to configuring mobile application using SAP Mobile Platform, enterprise edition, cloud version - Administration and Monitoring.

1. Get the SAP HANA Cloud account. See *Signing Up for an Account*.

   Note:
   - If you are a user with an SAP HANA Cloud trial (developer) account, you are automatically subscribed and authorized to access **SAP Mobile Platform, enterprise edition, cloud version** and you can skip Step 2. Proceed with *Getting Started*.
   - If you are a user with a SAP HANA Cloud productive account, you must manually subscribe to **SAP Mobile Platform, enterprise edition, cloud version** and get authorization access using Step 2. After getting authorization, proceed with *Getting Started*.
2. Get administration, authentication, and authorization for SAP Mobile Platform, enterprise edition, cloud version, Administration and Monitoring portal.
   a. Go to the SAP HANA Cloud Account page: *https://account.hana.ondemand.com*.

   b. To assign users to roles, select the **AUTHORIZATIONS** tab.
   c. On the **Roles** subtab, enter user ID.
   d. Select the **Application** and **Role** from drop-down lists and click **Show**.
   e. Under **Assigned role <Role> for users:**, select **Users** or **Groups** from drop-down.
   f. Click **Assign**.
   g. In **Assign role <Role> for user** dialog, enter the **User ID** and click **Assign**.
   For more information on various roles and associated tasks, see *Platform Administration Roles and Tasks*.

3. Open your SAP HANA Cloud trial (developer) account page using `https://account.hanatrial.ondemand.com/` and SAP HANA Cloud productive account using `https://account.hana.ondemand.com/`.
   The SAP HANA Cloud account welcome page is displayed with a link to the **SAP Mobile Platform, enterprise edition, cloud version**.



4. Click **SAP Mobile Platform, enterprise edition, cloud version**.
   The SAP Mobile Platform Administration and Monitoring portal is displayed in a new window. The direct URL to access SAP Mobile Platform Administration and Monitoring portal for a trial (developer) account is `https://smp-<account_name>.hanatrial.ondemand.com/Admin` and productive account is `https://smp-<account_name>.hana.ondemand.com/Admin`.

Figure 1: SAP Mobile Platform Administration and Monitoring Portal

Note: If you do not have authorization to access **SAP Mobile Platform, enterprise edition, cloud version**, an error is displayed. To get authorization, see *https://help.hana.ondemand.com/mobile/frameset.htm?doc/html/mdw1361529553461.html*

Next

Go to **APPLICATIONS** tab in SAP Mobile Platform, enterprise edition, cloud version - Administration and Monitoring to configure a mobile application.

For more information, see SAP Mobile Platform, enterprise edition, cloud version - Administration and Monitoring *Administration*.

Creating an Application

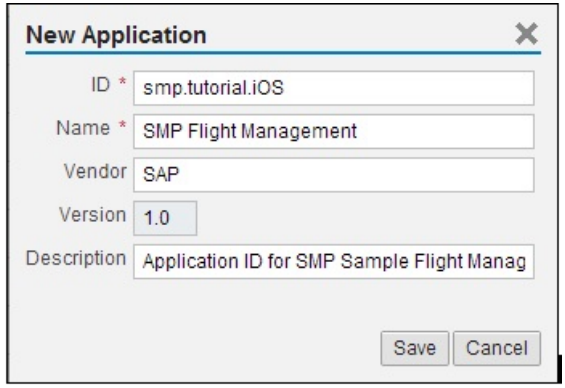
Create a new application using SAP Mobile Platform, enterprise edition, cloud version - Administration and Monitoring

Prerequisites

Connect to SAP Mobile Platform, enterprise edition, cloud version - Administration and Monitoring.

Task

1. Under **Applications** tab, click the **New**.
2. In **New Application** dialog box, enter the values:
   - **ID** - `smp.tutorial.iOS`
   - **Name** - `SMP Flight Management`
   - **Vendor** - `SAP`
   - **Description** - `Application ID for SMP Sample flight management`



3. Click **Save**.

Next

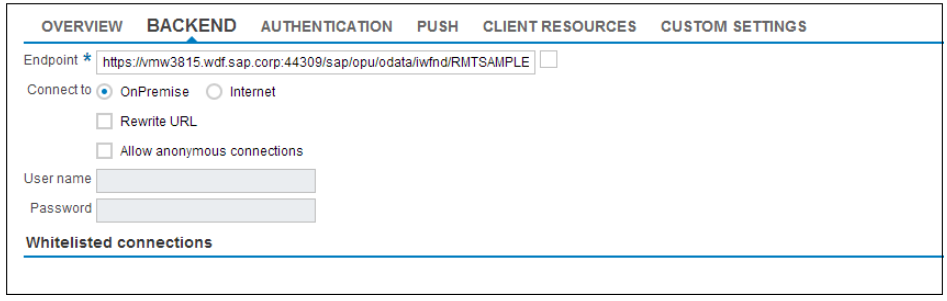
On successful creation of new application, you will be automatically taken to **BACKEND** tab, where you should configure the application endpoint.

Creating Application Endpoint URL

Create the enterprise information system (EIS) or backend connection.

1. Under **BACKEND** tab, enter **EndPoint** as `http://vmw3815.wdf.sap.corp:50009/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/`.
2. In **Connect to** field, select **OnPremise**.

Note: You can retain the default values for other fields.



Next

Navigate to **AUTHENTICATION** tab.

Creating a Security Profile

Create a new security profile and store the application settings.

1. Under **AUTHENTICATION** tab, select **New Profile**:
   - Enter **Security Configuration Name** - `SSO`
   - Retain the default values under **General Settings**.
   - Select **Authentication Type** - `Basic Authentication`.
   - Enter **Authentication URL** - `http://vmw3815.wdf.sap.corp:50009/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/`
2. Click **Save**.
   The **Confirm action** dialog box opens.
3. Click **OK**.
   The application status turns to green and cloud destinations are created.

Next

Create the user interface and application logic.

Developing an iOS Application

Generate OData API code for the iOS platform, develop a universal iOS device application with code, and test its functionality. The device application communicates with the Unwired Server.

Prerequisites

Before starting with the application development, ensure that the following requirements are met:

- You should have basic knowledge of Open Data Protocol (OData). For more information, see *Open Data Protocol* .
- OData service document (Sample Flight)is available at `http://vmw3815.wdf.sap.corp:50009/sap/opu/sdata/iwfnd/RMTSAMPLEFLIGHT/` .
- Unwired Server is available with application referring to OData service document.
  - On-Premise - Create application ID and refer to OData service document, see *Creating an Application ID and Whitelisting the Application Endpoint* on page 6.
  - Cloud - Create application endpoint URL and refer to OData service document, see *Creating Application Endpoint URL* on page 13.

Note: This tutorial has been developed using Sybase Unwired Platform 2.2 SP03, Mac OS X 10.7.5, iOS SDK 6.0, and Xcode 4.5.1 Development Environment, and executed on an iOS Simulator v 6.0 (358.4). If you use a different version, some steps may vary. For more information on Xcode, refer to the Apple Developer Connection: *http://developer.apple.com/technologies/tools/whats-new.html*.

1. Complete the tasks in *Getting Started with Mobile Platform*.
2. Download and deploy the SMPFlightManagement example project (complete project files) from the SAP® Community Network: *http://scn.sap.com/docs/DOC-8803*

   Note: If you upgrade SAP Mobile SDK after completing the tutorial, you can convert the project to the current SDK by importing the earlier project into the Sybase Unwired WorkSpace and then accepting the confirmation prompt.
3. (Optional) To use as a reference and copy source code when completing this tutorial, download the iOS SMPFlightManagement example project (source code only) and extract to your Mac from the SAP® Community Network: *http://scn.sap.com/docs/DOC-8803*

Task

1. *Installing the iOS Development Environment*

Install the iOS development environment, and prepare iOS devices for authentication.

2. *Creating an iOS Project*

   Set up and create an iOS client application in the Xcode IDE.

3. *Creating the User Interface*

   Use Interface Builder to create and configure the user interface for the SMPFlightManagement application.

4. *Defining the Application Logic*

   Define the application logic using REST SDK.

5. *Deploying the Device Application on iPhone Simulator*

   Deploy the SMPFlightManagement application to the iPhone simulator for testing.

Installing the iOS Development Environment

Install the iOS development environment, and prepare iOS devices for authentication.

See also

- *Creating an iOS Project* on page 17

Downloading Older Versions of the Xcode IDE

If you do not have the supported version of Xcode and the iOS SDK, you need to download it from the Downloads for Apple Developers Web site.

See *Supported Hardware and Software* for the most current version information for mobile device platforms and third-party development environments. If necessary, you can download older versions.

1. Go to *http://developer.apple.com/downloads/*.

   You must be a paying member of the iOS Developer Program. Free members do not have access to the supported version.

2. Log in using your Apple Developer credentials.
3. (Optional) Deselect all Categories except Developer Tools to narrow the search scope.
4. Download the supported Xcode and SDK combination.

Downloading the Xcode IDE

Download and install Xcode.

1. Download Xcode from the Apple Web site: *http://developer.apple.com/xcode/*.
2. Complete the Xcode installation following the instructions in the installer.

Creating an iOS Project

Set up and create an iOS client application in the Xcode IDE.

Prerequisites

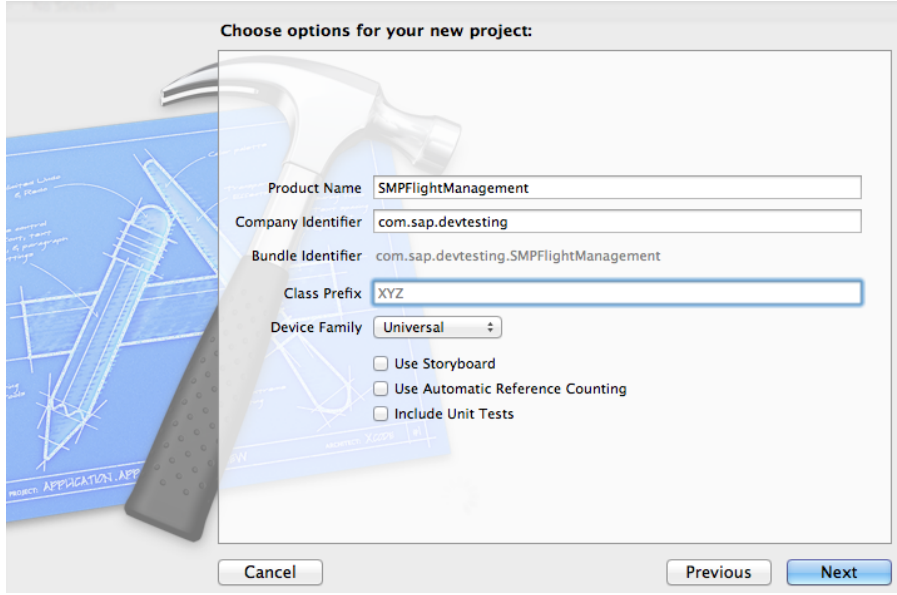
- Verify that Sybase Unwired Platform is installed in a shared directory so you can access it from your Mac.
- To help create your project, and to later build the interface, download and import the iOS SMPFlightManagement (2.2 SP03) example project from the SAP Community Network (SCN) at *http://scn.sap.com/docs/DOC-8803*.
- Copy the SMPFlightManagement iOS OData example project to your Mac machine and extract it into a folder.

Task

1. On your Mac, start Xcode and select **Create a new Xcode project**.
2. Select **iOS Application** and **Single View Application** as the project template, and then click **Next**.
3. Specify these values and click **Next**.
   a) Enter `SMPFlightManagement` as the product name.
   b) Enter **`com.<MyCorporation>.<BundleID>`** (or another value as needed) as the company identifier.

      Note: You may enter the class prefix, as needed. It is not mandatory to enter the class prefix.

   c) Select **Universal** as the device family product.
   d) Unselect **Use Storyboard**.
   e) Unselect **Use Automatic Reference Counting**.
   f) Unselect **Include Unit Tests**.



4. Select a location in which to save the project and click **Create** to open it.

   Xcode creates a folder, `SMPFlightManagement`, to contain the project file, `SMPFlightManagement.xcodeproj`, and another `SMPFlightManagement` folder, which contains a number of automatically generated files and a build folder. By default, `ViewController.h` and `ViewController.m` classes are added in the `SMPFlightManagement` folder along with `AppDelegate.h` and `AppDelegate.m` files.

5. Verify that the SDK and deployment targets are correct:
   a) Select `SMPFlightManagement` in Project Navigator and then select **Build Settings**.
   b) Under **Project**, select `SMPFlightManagement`.
   c) Verify that Base SDK under Architectures is set to `Latest iOS (iOS 5.0)`.
   d) Select **Info** and set the iOS Deployment Target to `iOS 5.0`.
   e) Select **Targets** > SMPFlightManagement and verify that those values are also set.

Note: For deploying on iOS simulator, accept the default values under **Code Signing**. For deploying on a device, set the required provisioning profile. See the *iOS Provisioning Portal documentation* on the Apple Developer Website: *https://developer.apple.com/devcenter/ios/index.action* .

Next

Add libraries, resources, and source code to the `SMPFlightManagement` Xcode project.

See also

- *Installing the iOS Development Environment* on page 16
- *Creating the User Interface* on page 21

Adding Source Code Files, Libraries, and Resources to the Xcode Project

Once you set up the initial project in Xcode, add files from the Sybase Unwired Platform folders.

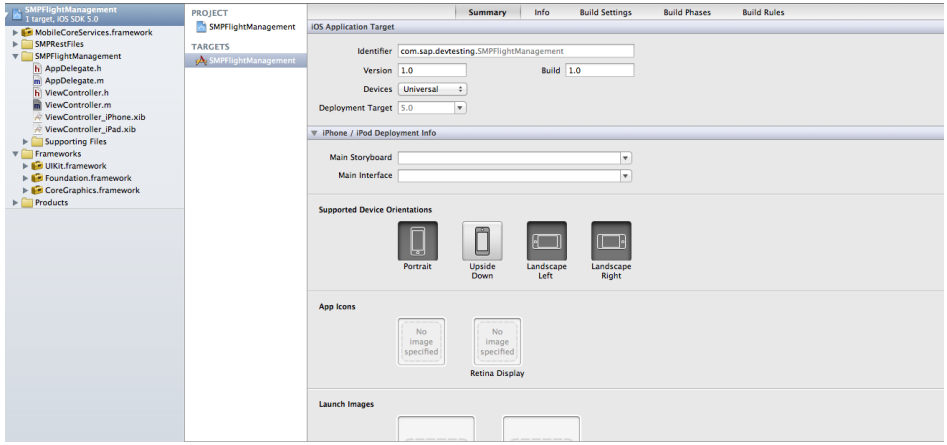
1. Copy REST SDK connectivity libraries from `<buildlocation>` to `SMPRESTSDK` folder on your Mac.
2. In the Xcode Project Navigator, Ctrl-click the **SMPFlightManagement** project, then select **Add Files to "SMPFlightManagement"**.

   Select the `SMPRESTSDK` folder (contains **includes** and **libraries** folder by default), and click **Add**.

   The `SMPRESTSDK` folder is added to the project in the Project Navigator. The libraries are added to the project in the Project Navigator.
3. Copy the `SMPFlightManagement` folder from the `SMPFlightManagement` REST SDK tutorial zip file to the `SMPFlightManagement` project folder on your Mac.
4. Add the source code files that you copied from the SMPFlightManagement iOS example project.
   a) In Xcode, Ctrl-click the SMPFlightManagement project and select **Add Files to "SMPFlightManagement"**.

      Select the `SMPFlightManagement` folder and click **Add**.

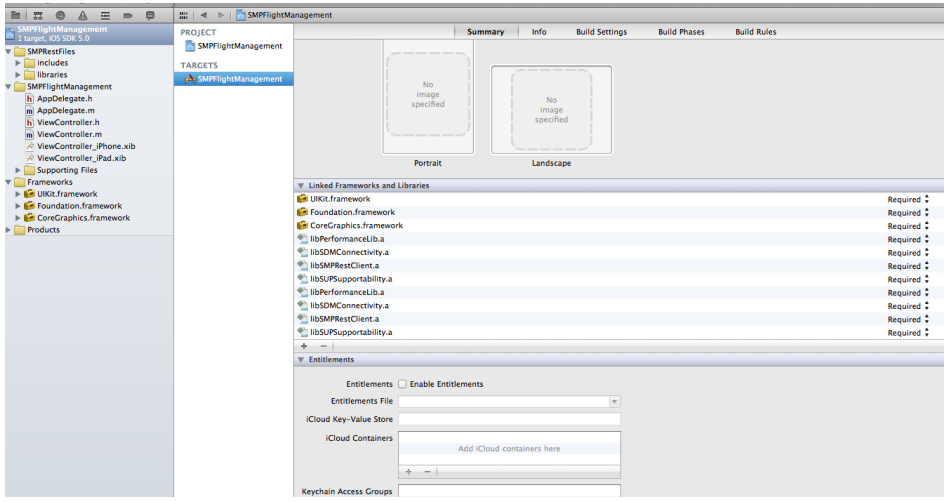
   The project now looks like this:



Configuring the Build Settings

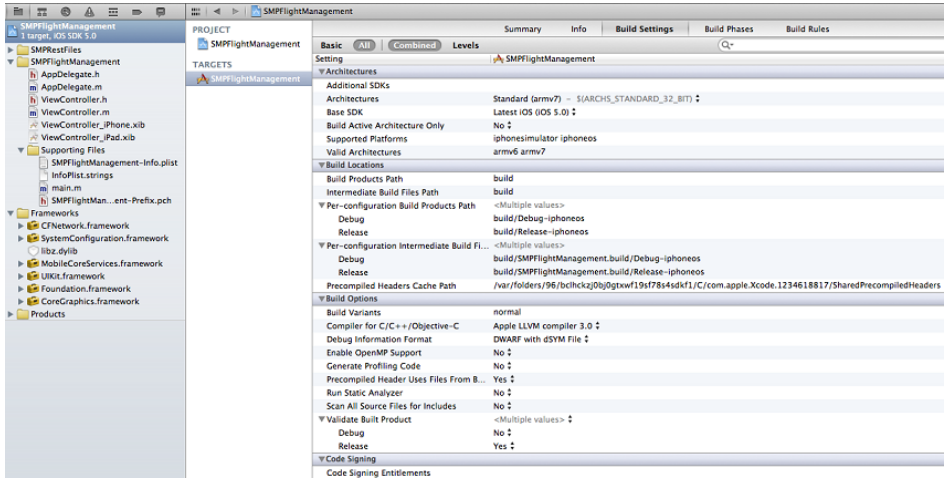
Configure the build settings for the Xcode project, then build the project.

1. In the Project Navigator, select the `SMPFlightManagement` under Projects and `SMPFlightManagement` folder under Targets.
2. In the Project Navigator > Summary, navigate to **LinkedFrameworks and Libraries** pane to add the frameworks. Click the **+** icon below the list, select the **MobileCoreServices.framework** libraries, and then click **Add** to add them to the project:



Note: The library version corresponds to the configuration you are building. In this tutorial, you work with the libraries for the Debug and Release version of the iPhone simulator.

3. Delete `ViewController.h` and `ViewController.m` files that are created by default.
4. Add the `LoginViewController.h` and `FlightDetailsViewController.h` source code files from the SMPFlightManagement folder in the REST API example project on SCN.
   a) In Xcode, ctrl-click the `SMPFlightManagement` project and select **Add Files to " SMPFlightManagement "**.
   b) Select the SMPFlightManagement > SMPFlightManagement folder from the SMPFlightManagement tutorial ZIP file on SCN.
   c) Select **Copy items into destination group's folder** (if needed).
5. Modify the code in AppDelegate.h and AppDelegate.m to include the newly created view controller. Replace ViewController with LoginViewController (first login screen displayed in the application). The project now looks like this:



6. In the Project Navigator, under Target, select **SMPFlightManagement** > **Build Phases**, then expand the **Copy Bundle Resources** section. Select `SMPFlightManagement-info.plist` and click on the **-** sign to remove it.
7. Hold the Option key, and select **Product > Clean** , then **Product > Build** to test initial project setup. If you correctly followed this procedure, you see a `Build Succeeded` message.

Creating the User Interface

Use Interface Builder to create and configure the user interface for the SMPFlightManagement application.

The SMPFlightManagement iOS example project contains the source code for the user interface for the sample application. Although the user interface is built automatically when you add the source files to the Xcode project, you can walk through the rest of the tasks and view the source code to see how to use Interface Builder to build the sample application.

See also

- *Creating an iOS Project* on page 17
- *Defining the Application Logic* on page 32

Viewing the Appdelegate Files

The `Appdelegate.h` and `Appdelegate.m` files are created when you create the Xcode project.

The `Appdelegate` files makes use of the `DataVault` API to store and retrieve the settings (such as Sybase Unwired Platform credentials).

When the application is launched, the native iOS method `didFinishLanunchingWithOptions` is called.

```
- (BOOL)application:(UIApplication *)application
 didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
```

Inside the `didFinishLanunchingWithOptions` method, initialize two view controllers `SettingsViewController` and `AppPasscodeViewController`.

The Navigation Controller is also initialized as a `rootViewcontroller` of the UI window. If the application is launched for the first time, the `SettingsViewController` view is called. After the settings are saved and the user is registered, the `AppPasscodeViewController` view is launched.

```
/* Initializing SettingsViewController*/
    SettingsViewController
*l_firstViewController=[[[SettingsViewController alloc]
    initWithNibName:@"SettingsViewController" bundle:nil]
autorelease];
   /* Initializing navigationController with SettingsViewController
as rootViewController */
       UINavigationController *navigationController =
[[[UINavigationController alloc]

initWithRootViewController:l_firstViewController]autorelease];
       [self setM_viewController:navigationController];
     /* Setting navigationController as a rootViewController of
UIWindow */
        self.window.rootViewController = self.m_viewController;
        @try {
             /*Checks whether specified vault exists. If the value
returned is Yes, then initialize
        AppPasscodeViewController and push the
AppPasscodeViewController as the first View*/
        if ([DataVault vaultExists:VAULT_NAME])
            {
              AppPasscodeViewController
*passCodeViewCOntroller=[[[AppPasscodeViewController alloc]
           initWithNibName:@"AppPasscodeViewController" bundle:nil]
autorelease];
              [navigationController
pushViewController:passCodeViewCOntroller animated:YES];
            }
            }
              @catch (DataVaultException *exception)
             {
             NSLog(@"Failed with error %@",[exception description]);
       }
```

Creating User Interface for Welcome Screen

On launching the SMPFlightManagement application, the Welcome splash screen is displayed.

Create a new splash screen (image file) named **Default.png** with size 320x480 for iPhone. Add the image file in your `Resources/iphone` folder in the **SMPFlightManagement** project. The splash screen appears when you first launch the application. When the Xcode build is executed, it checks for the **Default.png**, and displays it as the first screen of the application.

Creating User Interface for Settings Screen (SettingsViewController)

Create user interface for **SettingsViewController** using Interface Builder.

For the **SettingsViewController**, implement the following code:

```
    /* Set the title of navigation controller  */
        self.title=@"Settings";

    /* By default the color of navigation controller bar is blue. To
make it black translucent color:*/
        [self.navigationController.navigationBar
setTranslucent:YES];
        [self.navigationController.navigationBar setTintColor:
[UIColor blackColor]];

    /* To add Right bar button (Log In) in navigation controller
programmatically, use the following code*/
        UIBarButtonItem *l_loginButton = [[UIBarButtonItem alloc]
initWithTitle:@"Log In"

         style:UIBarButtonItemStyleBordered

         target:self

           action:@selector(loginbuttonClickHandler:)];

           [self.navigationItem setRightBarButtonItem:l_loginButton
animated:YES];
           [l_loginButton release];

    /* To add Left bar button (Cancel) in navigation controller
programmatically, use the following code*/
        UIBarButtonItem *l_cancelButton = [[UIBarButtonItem alloc]
initWithTitle:@"Cancel"

         style:UIBarButtonItemStyleBordered

         target:self
```

```
           action:@selector(cancelbuttonClickHandler:)];

          [self.navigationItem setRightBarButtonItem:l_cancelButton
animated:YES];
           [l_cancelButton release];
```

The application logic used for **Login** button is:

```
/* In Login button, click call back */

 /* To create a new secure vault using createVault method. This
method also assigns a password which is the appPasscode provided by
the user in Settings screen,
    and salt value to the vault. */
    DataVault *l_vault= nil;
       @try {
        l_vault = [DataVault createVault:VAULT_NAME
password:appPasscode salt:SALT_PASSWORD];

        /*Checks if a vault with the same name already exists. This
method throws an exception. A
        newly created vault is in the unlocked state. */

         if ([DataVault vaultExists:VAULT_NAME])
             {
               NSLog(@"Vault %@ created",VAULT_NAME);

        ////Register User and after successful registration add below
code to store the settings in DataVault

           /* Storing the settings in the vault already created and
lock the vault. */
               [l_vault setString:@"AppConID" value:appConnID];
               [l_vault setString:@"EnableHttp" value:YES];
               [l_vault setString:@"Hostname"
value:@"10.66.176.121"];
               [l_vault setString:@"Portnumber" value:@"8000"];
               [l_vault setString:@"Username" value:@"supuser"];
               [l_vault setString:@"Password" value:@"s3puser"];
               [l_vault lock];
             }
            }
            @catch (DataVaultException *exception) {
            NSLog(@"Vault creation failed with error %@",[exception
description]);
             if ([DataVault vaultExists:VAULT_NAME])
               {

               NSLog(@"Vault  exists before delete vault");

               [DataVault deleteVault:VAULT_NAME];


                 NSLog(@"deleted vault %@ successfully",VAULT_NAME);
```

```
                }

            }
```

For creating a radiobutton (switch), use the **UISwitch** control:

```
        /*Inside cellfcellForRowAtIndexPath method of tableview,
programmatically add UISwitch in third
        row of first section to determine whether request if HTTP/
HTTPS depending on ON/OFF state of
        UISwitch*/

          /* Initializing switch with frame CGRectZero -- equivalent
to CGRectMake(0, 0, 0, 0).*/               U
                ISwitch *switchView = [[UISwitch alloc]
initWithFrame:CGRectZero];
             /* Add switch as accessoryView of tableviewcell */

               httpEnableCell.accessoryView = switchView;

                /* Sets the state of Switch whether ON/OFF using
bollean parameter */
                   [switchView setOn:[[self.m_propDetailDict
objectForKey:[NSString
stringWithFormat:@"%i",indexPath.row]]boolValue] animated:NO];

              /* Add target/action for particular event. The action
may optionally include the sender and the
                 event in that order. So whenever switch is tapped,
state can be tracked in selector method*/
                    [switchView addTarget:self
action:@selector(switchChanged:)
forControlEvents:UIControlEventValueChanged];
                `
                  /* Release the object therefore memory allocated
during initialization */
                     [switchView release];
```

The application logic used for **Cancel** button is:

```
/* At first launch of application, when user is not registered,
pressing Cancel button will shows an alert message pop-up @"User not
registered".
Once user is registered, then pressing cancel button will push
another view controller with list of data (Carriers List in our
application). */
```

Configuring the SettingsViewController View

Use Interface Builder to configure the `SettingsViewController.xib` file and create the user interface. Although the provided XIB file is already configured, you can walk through the steps to see how to create the interface.

1. Click the `SettingsViewController.xib` file to reveal a view of the (presently empty) screen in the right pane and the following three items represented by icons in the middle pane:
   - File's Owner – the object that is set to be the owner of the user interface, which is typically the object that loads the interface. In this tutorial, this is the SettingsViewController.

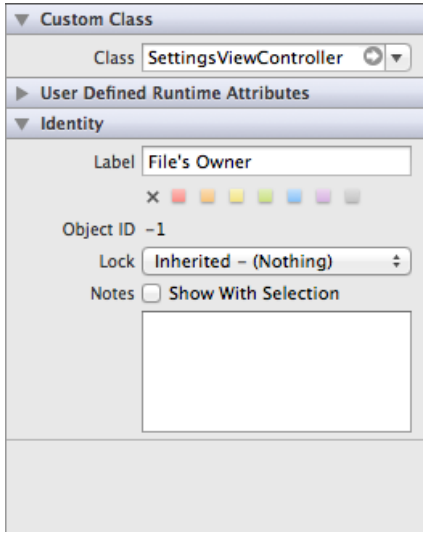
   - First Responder – the first responder proxy object handles events. Connecting an action to the first responder means that when the action is invoked, it is dynamically sent to the responder chain.

   - View – appears in a separate window to allow editing.

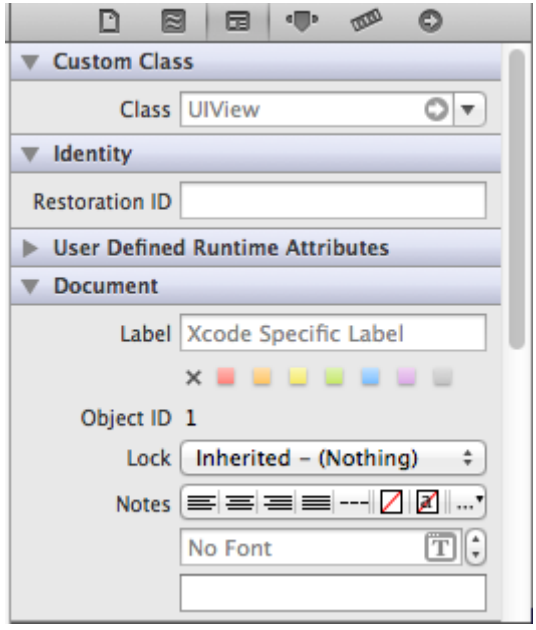
     

2. Select the **File's Owner** icon, click **View** in the utility area, click **Show the Identity Inspector**, and make sure `SettingsViewController` appears in the Class field under Custom Class.

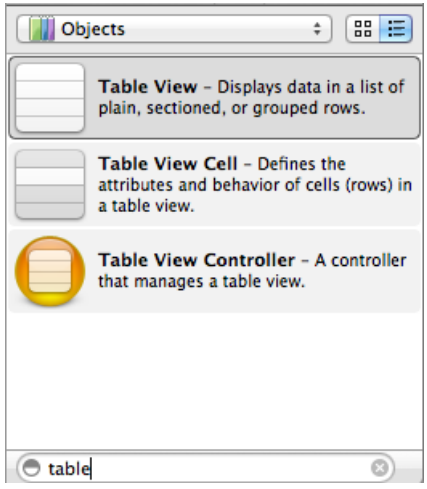
   

This tells Interface Builder the class of the object to allow you to make connections to and from the File's Owner.

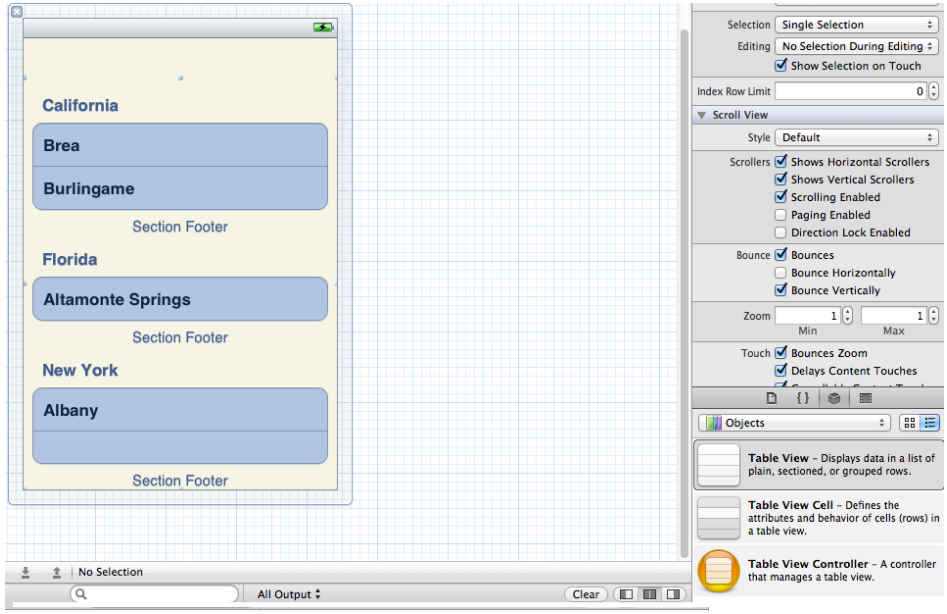
3. Click the **View** icon, and in the Identity Inspector panel, and make sure `UIView` appears in the Class field under Custom Class.



4. To add a Table view, select **View** > **Utilities** > **Show Object Library**.



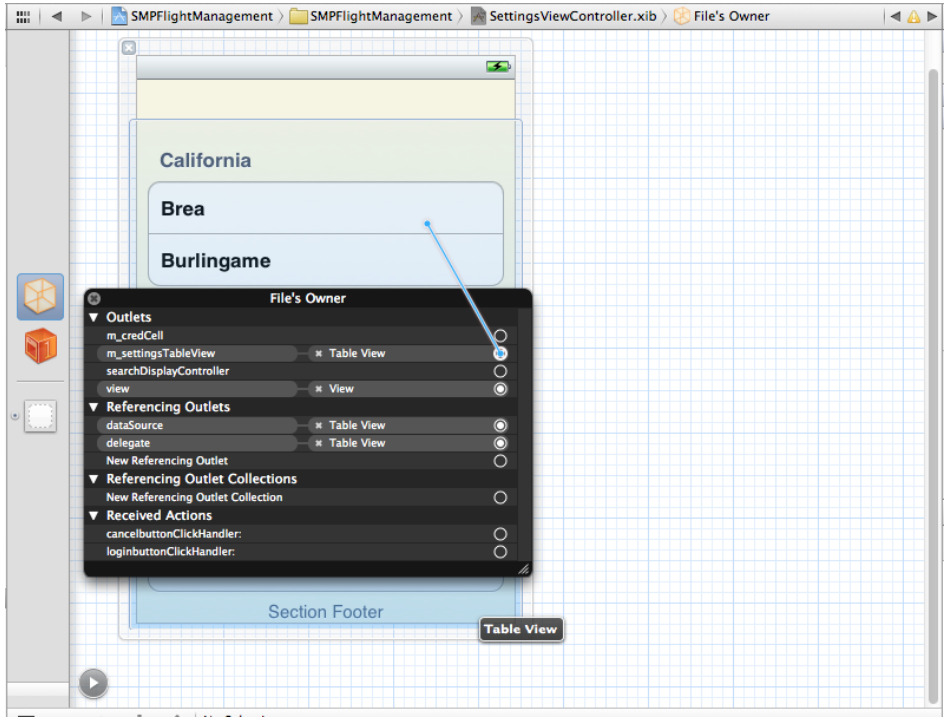
5. In the Object Library pane, select the **Table View** item, and drag it onto the view.



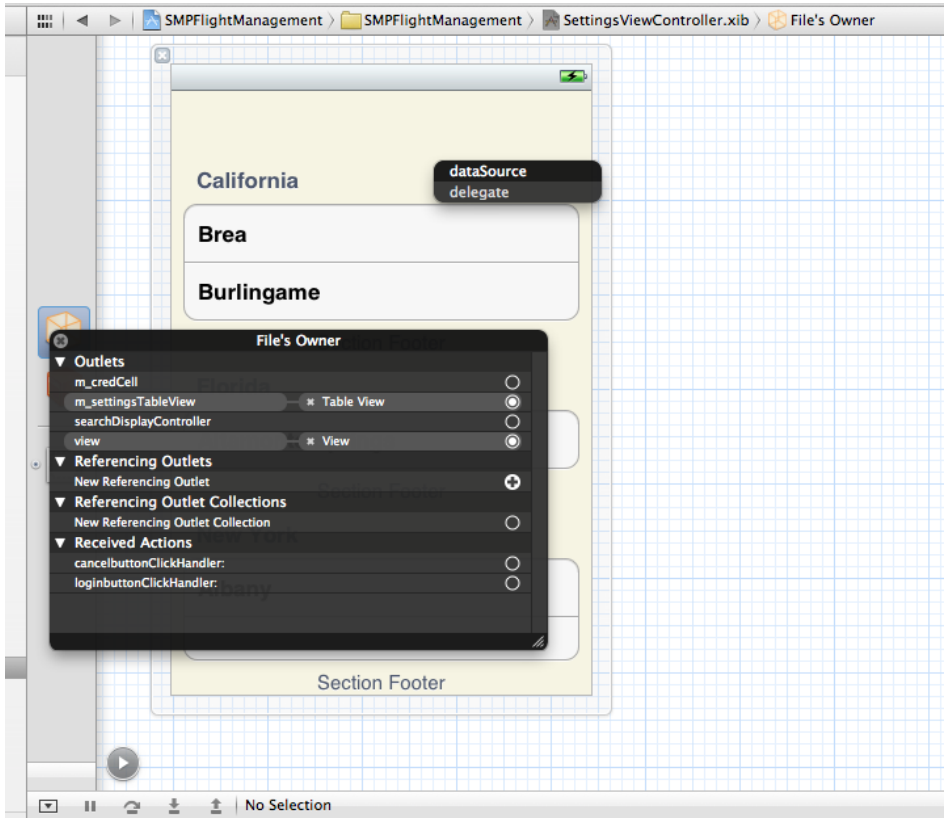
6. To make connections to the user interface from the view controller, the `SettingsViewController.h` file contains the outlets.

```
@ interface SettingsViewController : UIViewController
                {
                    IBOutlet UITableView *m_settingsTableview;
                }
```

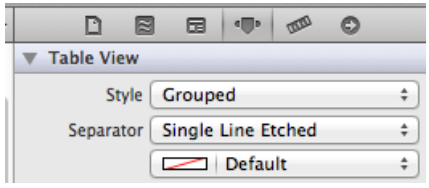
7. Go to `SettingsViewController.xib`, select and right-click the **File Owner**. In Outlets, **m_settingsTableview** appears.



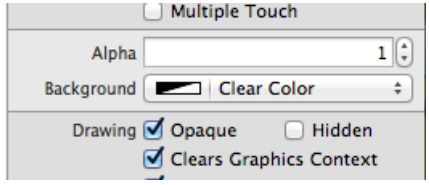
8. Drag from the **m_settingsTableview** circle to view and release. The connection to the **File Owner** has been made. A pop-up with datasource and delegate appears in the table view.



9. Select the **Table View**, go to **IBInterfaceBuilder**, select **Style to Grouped**.



10. Set the background color of the view to **Clear Color**.



11. In **Referencing Outlets -> New Referencing Outlet**, drag from the right circle to table view and release. A pop-up with datasource and delegate appears in the view. Select **datasource** to establish a connection between table view and table view datasource methods.

12. In **Referencing Outlets -> New Referencing Outlet**, drag from the right circle to table view and release. A pop-up with datasource and delegate appears in the view. Select **delegate** to establish a connection between table view and table view callback methods.

Creating User Interface for App Passcode Screen (AppPasscodeViewController)

Add **UIAlertview** and **UITextField** using Interface Builder to take passcode as user input.

Add **UIAlertview** and **UITextField** using the following code:

```
UIAlertView *passCodeAlertView = [[UIAlertView alloc]
initWithTitle:@"Enter application passcode:"
                              message:@"\n"
                              delegate:self
                            cancelButtonTitle:@"Log In"
                            otherButtonTitles:nil];
       CGRect frame = CGRectMake(12.0, 45.0, 260.0, 30.0);  //(20.0,
45.0, 245.0, 25.0) //(12.0, 64.0, 260.0, 30.0)
        UITextField *l_passCodeTextField = [[UITextField alloc]
initWithFrame:frame];
        l_passCodeTextField.placeholder = @"Passcode";
        l_passCodeTextField.borderStyle =
UITextBorderStyleRoundedRect;  //UITextBorderStyleBezel
        l_passCodeTextField.keyboardAppearance =
UIKeyboardAppearanceAlert;
        l_passCodeTextField.secureTextEntry=YES;
       l_passCodeTextField.keyboardType = UIKeyboardTypeDefault;

        l_passCodeTextField.returnKeyType = UIReturnKeyDone;

l_passCodeTextField.autocapitalizationType=UITextAutocapitalization
TypeNone;
        l_passCodeTextField.clearButtonMode =
UITextFieldViewModeWhileEditing;
        l_passCodeTextField.delegate = self;
        l_passCodeTextField.tag = 100;

        [l_passCodeTextField becomeFirstResponder];
        [passCodeAlertView addSubview:l_passCodeTextField];

        [passCodeAlertView show];
        [passCodeAlertView release];
        [l_passCodeTextField release];
```

Creating Flight Collection Screen (FlightDetailsViewController)

Create a table using **Table View** option in the Interface Builder, as implemented in **SettingsViewcontroller**.

For displaying the list of flight carriers, add the following code snippet in `cellForRowAtIndexPath` method of tableview:

```
/*Iterate through the entries saved in carriers List array got after
parsing the OData document and get the
```

```
       required fields and display in the format CARRNAME (carrid)*/

        SDMODataEntry* displayEntry = [sCarriersList objectAtIndex:
[indexPath row]];
        SDMODataPropertyValueObject* valueObject = [displayEntry
getPropertyValueByPath:@"CARRNAME"];
        SDMODataPropertyValueObject* codeValue = [displayEntry
getPropertyValueByPath:@"carrid"];

        cell.textLabel.text = [NSString stringWithFormat:@"%@(%@)",
[valueObject getValue], [codeValue getValue]];
```

Defining the Application Logic

Define the application logic using REST SDK.

See also

- *Creating the User Interface* on page 21
- *Deploying the Device Application on iPhone Simulator* on page 36

Registering a User

Register a user using a predefined authentication mechanism asynchronously.

Initialize the `ClientConnection` class using `SMPClientConnection* clientConn = [SMPClientConnection initializeWithAppID:@"application ID" domain:@"domain" secConfiguration:@"secconfig"];`

```
SMPClientConnection  *clientConn=[ SMPClientConnection
initializeWithAppID : @”com.sap.NewFlight”   domain : @”default”
secConfiguration : @”HttpAuth” ];
```

Provide server details such as host and port, and set `enableHTTP` to `Yes` for HTTP Request. By default, the connections are HTTPS, using `[clientConn setConnectionProfileWithHost:host port:port farm:farm relayServerUrlTemplate:relayserverurltemplate enableHTTP:isHttpRequest];`

```
[clientConn setConnectionProfileWithHost:@”10.66.187.60”
        port:@”8000” farm:nil  relayServerUrlTemplate:nil
enableHTTP:YES];
```

Set the delegate in case of asynchronous registration. This class has to implement `SMPUserManagerDelegate` to get the callback

```
[SMPUserManager setDelegate:self];
```

Initialize the `UserManager` with `clientConnection` as one of the parameter and using `SMPUserManager* userManager = [SMPUserManager initializeWithConnection:clientConn];`

```
SMPUserManager *userManager = [SMPUserManager
initializeWithConnection:clientConn];
```

Asynchronously register the user using `[userManager registerUser:@"username" password:@"password" error:&error isSyncFlag:NO];`

Note: The `isSyncFlag` can be set to `YES` (for synchronous onboarding) or `NO` (for asynchronous onboarding).

```
[userManager registerUser:@”supuser” password:@”s3puser” error :nil
isSyncFlag:NO ];
```

Implement these two callbacks for successful registration or registration failure.

```
-(void)userRegistrationSuccessful:(SMPUserManager *)userManager

{
                    //Registration Successful
          /* Get the Application connection ID send by server after
successful registration using
       [userManager applicationConnectionID] which can be set using

        [clientConn
setApplicationConnectionID:ApplicationConnectionId]; whenever
application
        restarts. */

               NSString *appConnID=[userManager
applicationConnectionID];

                }
                  -(void)userRegistrationFailed:( SMPUserManager
*)userManager
                {
                  NSError  *error=[userManager registrationError];
                 NSLog(@"User Registration Failed with error:   %@",
[error description]);
                }
```

Sending Data Request to the Backend

Send a data request to the back-end through the Unwired Server asynchronously.

Initialize the `SMPAppSettings` class which takes client connection object as one of the parameter using `SMPAppSettings *appSettingsObj=[SMPAppSettings initializeWithConnection:clientConn userName:username password:password];`

```
/*Initialize the application settings class which takes client
connection object as one of the parameters */

   SMPAppSettings *appSettingsObj=[ SMPAppSettings
```

```
initializeWithConnection :clientConn  userName : @”supuser”
password : @”s3puser” ];

   /* Get the application endpoint using [appSettingsObj
getApplicationEndpointWithError:error]        */

    NSString * appEndPoint =[appSettingsObj
getApplicationEndpointWithError:nil];

   /*Initialize the SDMRequesting class object with the endpoint URL
after successful registration */
      id<SDMRequesting> request=nil;
      if([[self.m_reqTypeDic objectForKey:KEY_REQUEST_TYPE]
isEqualToString:SERVICE_DOC_REQUEST])
      {
       /* Get Service Document    */
         request = [SDMRequestBuilder requestWithURL:[NSURL
URLWithString:appEndPoint]];
       }
       else if([[self.m_reqTypeDic objectForKey:KEY_REQUEST_TYPE]
isEqualToString:METADATA_REQUEST])
       {
        /* Get metadata   */
        request=[SDMRequestBuilder requestWithURL:[NSURL
URLWithString:[NSString stringWithFormat:@"%@/
$metadata",m_AppEndPoint]]];
       }
        else if([[self.m_reqTypeDic objectForKey:KEY_REQUEST_TYPE]
isEqualToString:ODATA_REQUEST])
        {
         /* To get required flight collection. For example:
CarrierCollection */
         request=[SDMRequestBuilder requestWithURL:[NSURL
URLWithString:[NSString stringWithFormat:@"%@/CarrierCollection",
appEndPoint]]];
        }
         /*Set Username in SDMRequesting object*/
         [request setUsername: @”supuser”];

         /*Set Password in SDMRequesting object*/
         [request setPassword: @”s3puser”];

         /*Set the Delegate. This class must adhere to
SDMHttpRequestDelegate to get callback*/
         [request setDelegate:self];

         /*Call startAsynchronous API to request object to retreive
Data asynchronously in the call backs */
         [request startAsynchronous];
```

Retrieving the Response from Backend

Retrieve the response from the backend to the device.

Upon performing request using [`request startAsynchronous`], the response will arrive at either `- (void)requestFailed:(SDMHttpRequest*) request {}` or `- (void)requestFinished:(SDMHttpRequest*) request {}` depending on whether the transaction was a **Failure** or **Success** respectively.

Add the following in the `- (void)requestFinished(SDMHttpRequest*)` request method:

```
   /* The response parsing depends on the type of document queried
whether its service document or metadata or
    OData*/
    if([[self.m_reqTypeDic objectForKey:KEY_REQUEST_TYPE]
isEqualToString:SERVICE_DOC_REQUEST])
    {
     /*Initialize the SDMODataServiceDocumentParser class object for
parsing response service document
        data*/
        SDMODataServiceDocumentParser
        *svcDocParser=[[[SDMODataServiceDocumentParser
alloc]init]autorelease];
       /* Parses the service document XML and converts it to an
Objective-C service document object.*/
           [svcDocParser parse:[request responseData]];
          /* Service document instance can be accessed via the
"serviceDocument" property of the parser after
             parsing*/
             SDMODataServiceDocument
             *svcDoc=svcDocParser.serviceDocument;
         }

         else if([[self.m_reqTypeDic objectForKey:KEY_REQUEST_TYPE]
isEqualToString:METADATA_REQUEST])
         {
        /*Initialize the SDMODataMetaDocumentParser class object for
parsing response meta data*/
           SDMODataMetaDocumentParser
*metaDocParser=[[[SDMODataMetaDocumentParser
           alloc]initWithServiceDocument:svcDoc]autorelease];

          /*Parses and matches the schema with the service document
and its collections.*/

             [metaDocParser parse:[request responseData]];

             /* The parser creates the schema of the input service
document's collections and returns a collection by name */
               SDMODataCollection *carrierCollection=[svcDoc.schema
getCollectionByName:@"CarrierCollection"];
```

```
                }
                else if([[self.m_reqTypeDic
objectForKey:KEY_REQUEST_TYPE] isEqualToString:ODATA_REQUEST])
                {
               /* Initialize the SDMODataDataParser class object for
parsing any "inlined"entries or feed(s) when service
                    document is passed to the "initWithEntitySchema"
variant that accepts service document as
                 input. If "inlined" feed(s) or entries should not be
returned pass nil as the service
                    document parameter or use SDMODataDataParser*
dataParser = [[SDMODataDataParser alloc] initWithEntitySchema:
entitySchema] */

                SDMODataDataParser *dataParser=[[[SDMODataDataParser
alloc]initWithEntitySchema: carrierCollection.entitySchema
andServiceDocument: svcDoc]autorelease];

                /*Parses a feed or an entry xml.*/
                  [dataParser parse:[request responseData]];


             /* The array of parsed entry/entries can be accessed via
the "entries" property of the parser after parsing.
                Array of SDMOdata Entries can be iterated and diplay
the requisite data in tableview */

                 NSMutableArray * carriersList=[l_dataParser.entries
retain];
         }
```

Deploying the Device Application on iPhone Simulator

Deploy the **SMPFlightManagement** application to the iPhone simulator for testing.

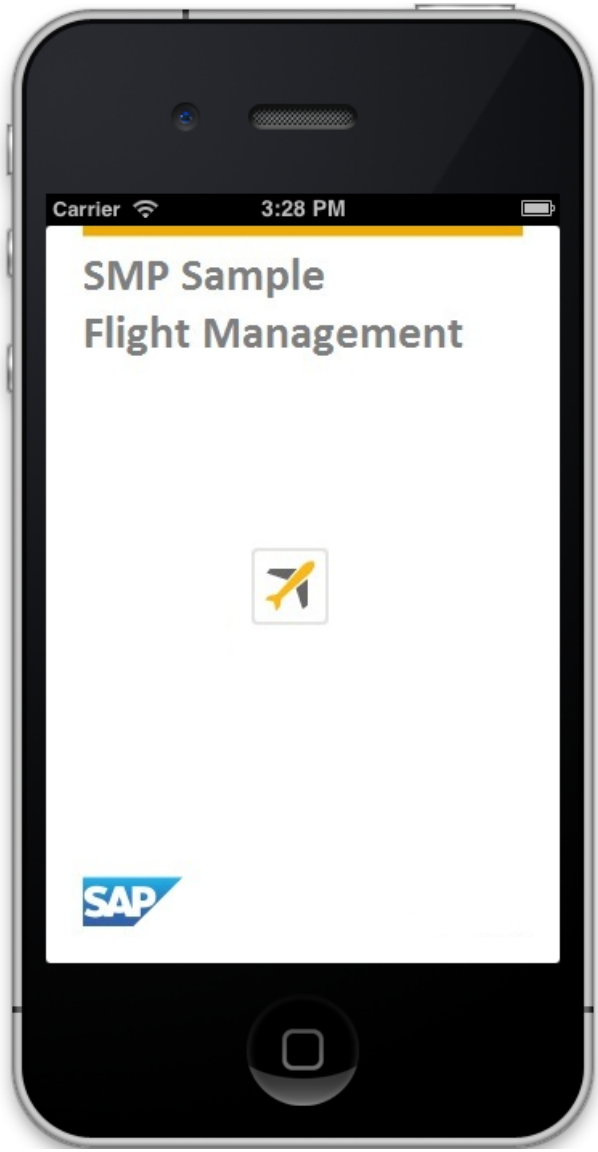
Prerequisites

Register an application connection in Sybase Control Center or SAP Mobile Platform, enterprise edition, cloud version - Administration and Monitoring.

You must be connected to the server where the mobile application project is deployed.

Task

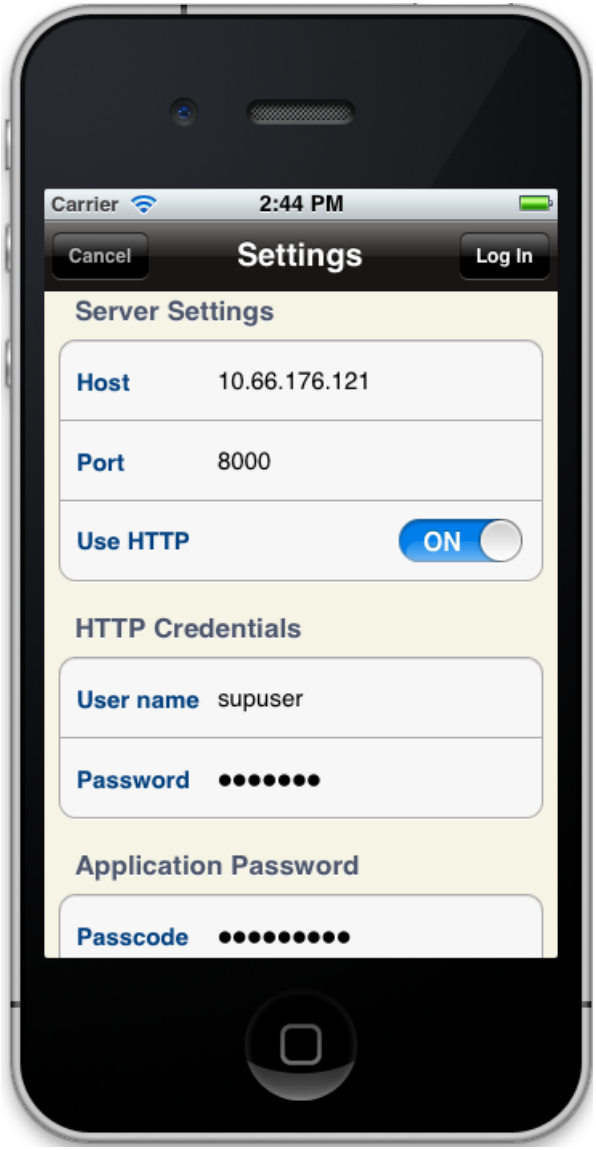
1. From the top menu, select **Product > Edit Scheme** to **iPhone 6.0 Simulator**.
2. Select **Product > Build** then **Product > Run** to build the project and start the iPhone simulator.
3. In the iPhone applications screen, open the **SMPFlightManagement** application. The **SMPFlightManagement** application's welcome page is displayed.



When you run the application for the first time, it exits immediately with a dialog asking you to enter the application settings in the **Settings** application.

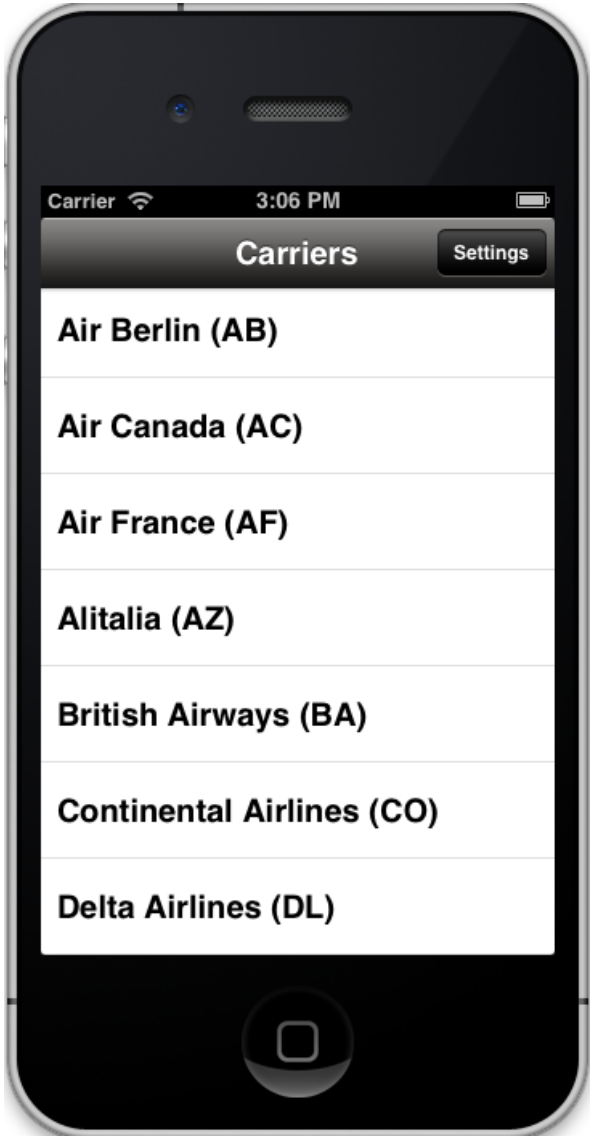
4. In the iPhone simulator, go to **Settings** > **SMPFlightManagement** to enter the connection settings.
   - **Host –** The name of the machine where SAP Mobile Server is running.
   - **Port –** SAP Mobile Server port number. The default port number for HTTP channel is 8000.

- **Use HTTP –** Enable if you want to use HTTP backend.
- **User Name –** The name of the user.
- **Password –** The user account password used to authenticate the user name entered.
- **App Passcode –** The application pin to securely store your application password, and a database encryption key that is generated when the application launches.



5. For subsequent launches of the application, you need enter only the **App Passcode** in **Log In** screen.

6. After you have successfully logged in to the application, you can see the carriers list.



7. Close the simulator to stop the **SMPFlightManagement** iOS application.

See also

- *Defining the Application Logic* on page 32

Learn More About Sybase Unwired Platform

Once you have finished, try some of the other samples or tutorials, or refer to other development documents in the Sybase Unwired Platform documentation set.

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

Tutorials

Try out some of the other getting started tutorials available on the Product Documentation Web site to get a broad view of the development tools available to you.

Example Projects

An example project contains source code for its associated tutorial. It does not contain the completed tutorial project. Download example projects from the SAP® Community Network (SCN) at *http://scn.sap.com/docs/DOC-8803*.

Samples

Sample applications are fully developed, working applications that demonstrate the features and capabilities of Sybase Unwired Platform.

Check the SAP® Development Network (SDN) Web site regularly for new and updated samples: *https://cw.sdn.sap.com/cw/groups/sup-apps*.

Online Help

See the online help that is installed with the product, or available from the Product Documentation Web site.

Developer Guides

Learn best practices for architecting and building device applications:

- *Mobile Data Models: Using Data Orchestration Engine* – provides information about using Sybase Unwired Platform features to create DOE-based applications.
- *Mobile Data Models: Using Mobile Business Objects* – provides information about developing mobile business objects (MBOs) to fully maximize their potential.
- *SAP Mobile WorkSpace: Mobile Business Object Development* – provides information about using SAP Mobile Platform to develop MBOs and generate Object API code that can be used to create native device applications and Hybrid Apps.

Use the appropriate API to create device applications:

- *Developer Guide: Android Object API Applications*
- *Developer Guide: BlackBerry Object API Applications*

- *Developer Guide: iOS Object API Applications*
- *Developer Guide: Windows and Windows Mobile Object API Applications*
- *Developer Guide: Hybrid Apps*
- *Developer Guide: OData SDK*
- *Developer Guide: REST API Applications*

Customize and automate:

- *Developer Guide: Unwired Server Runtime > Management API* – customize and automate system administration features.

Javadoc and HeaderDoc are also available in the installation directory.

Index