SYBASE®
An SAP® Company

Tutorial: iOS Object API Application Development

Sybase Unwired Platform 2.1

DOCUMENT ID: DC01213-01-0210-01
LAST REVISED: January 2012

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

Contents

Sybase Unwired Platform Tutorials

The Sybase® Unwired Platform tutorials demonstrate how to develop, deploy, and test mobile business objects, device applications, and mobile workflow packages. You can also use the tutorials to demonstrate system functionality and train users.

- Learn mobile business object (MBO) basics, and create a mobile device application:
  - *Tutorial: Mobile Business Object Development*
- Create native mobile device applications:
  - *Tutorial: BlackBerry Application Development*
  - *Tutorial: iOS Application Development*
  - *Tutorial: Windows Mobile Application Development*
- Create a mobile workflow package:
  - *Tutorial: Mobile Workflow Package Development*

The tutorials demonstrate a cross section of basic functionality, which includes creating MBOs, and using various Sybase Unwired WorkSpace development tools, independent development environments, and device types. Tutorial projects are available if you want the finished tutorial without going through the steps.

Task Flow

Use this tutorial to develop a device application for an Apple iOS device. Test the application on a simulator.

Table 1. Eclipse Tutorials

Task	Goals	Steps required to complete the task
Getting started	• Install all required WorkSpace components and external resources. • Start Unwired Server. • Open the Mobile Development perspective, and become familiar with the views of the perspective, the Mobile Application Diagram.	• Install Sybase Unwired Platform 2.1. • Start the Unwired Platform services. • Start Sybase Unwired Workspace. **Note:** These tasks are prerequisites for all the other tutorials. You need to perform them only once.
Developing a device application	• Create an iOS device application, and run it on the iPhone simulator.	• Generate the Object API code for iOS. • Set up the iOS client project in Xcode. • Register the application connection in Sybase Control Center. • Create the iOS application user interface. • Deploy the iOS application to the simulator.

Getting Started

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*

   Install Sybase Unwired Platform.
2. *Starting Unwired Platform Services*

   Start Unwired Server and the sample database.
3. *Starting Sybase Unwired WorkSpace*

   Start Unwired WorkSpace.
4. *Connecting to Sybase Control Center*

   Open the Web-based Sybase Control Center administration console to manage Unwired Server and its components.
5. *Learning the Basics*

   Learn about Sybase Unwired WorkSpace and how to access help (optional).

Installing Sybase Unwired Platform

Install Sybase Unwired Platform.

Before starting this tutorial, be sure you have all the requisite Unwired Platform components installed. For complete installation instructions, see:

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

- Install Sybase Mobile SDK, which includes:
  - Development support for Native Object API applications, HTML5/JS Hybrid (Mobile Workflow) applications, and OData SDK applications.
  - Sybase Unwired WorkSpace, the Eclipse-based development environment for MBOs and Mobile Workflows.
- Install Unwired Platform Runtime:
  - Data Tier (included with single-server installation)
  - Unwired Server

Starting Unwired Platform Services

Start Unwired Server and the sample database.

Click **Start > Programs > Sybase > Unwired Platform > Start Unwired Platform Services**.
The Unwired Server services enable you to access the Unwired Platform runtime components and resources.

Starting Sybase Unwired WorkSpace

Start Unwired WorkSpace.
Select **Start > Programs > Sybase > Unwired Platform > Unwired WorkSpace**.

The Sybase Unwired Workspace opens in the Mobile Development perspective. The Welcome page displays links to product information, and to the product.

Next

To read more about Sybase Unwired WorkSpace concepts and tasks, select **Help > Help Contents** from the main menu.

Connecting to Sybase Control Center

Open the Web-based Sybase Control Center administration console 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 devices
- Set role mappings

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

1. Click **Start > Programs > Sybase > Sybase Control Center**.

   Note: If the Sybase Control Center service does not open, make sure that the Sybase Control Center service is started. See the *Installation Guide for Runtime*.

2. In Sybase Control Center, log in by entering the credentials set during installation.

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

Learning the Basics

Learn about Sybase Unwired WorkSpace and how to access help (optional).

Prerequisites

Start Unwired WorkSpace.

Task

1. In the Welcome page, click any of the links to explore the Unwired WorkSpace environment.
2. To enter the Sybase Unwired WorkSpace development environment, click **Start Development** or close the Welcome tab.

   The default Mobile Development perspective provides ready access to most of the tools you need to create, update, and manage mobile business objects (MBOs). This table describes the main windows and views of the Mobile Development perspective. Note that not all the views are open initially; some views become available only after you begin developing your MBOs:

View or Window	Description
WorkSpace Navigator	A view of mobile application projects. Each project folder includes resources and data source references to which the MBOs are bound, personalization keys, and so on. Use this view to review and modify MBO-related properties.
Enterprise Explorer	A view of enterprise back-end resources, such as database servers, SAP® servers, and Sybase Unwired Server.

View or Window	Description
Mobile Application Diagram	A graphical editor for designing mobile business objects. A Mobile Application Diagrem is associated with each project. Use the Mobile Application Diagram to create MBOs (including attributes and operations), then define relationships with other MBOs. You can: • Create MBOs in the Mobile Application Diagram using Palette icons and menu selections. Either bind to a data source now or or defer binding. For example, using a top-down approach, you might model your MBOs before creating the data sources to which they bind. • Drag items from Enterprise Explorer and drop them onto the Mobile Application Diagram to create the MBO – quickly creates the operations and attributes automatically based on the data source of the items.
Palette	A view from which you can drag controls onto an open Mobile Application Diagram and define their attributes, operations, and relationships to your application.
Properties	A view that shows the properties of the object currently selected in the Mobile Application Diagram, and lets you edit them. You cannot create an MBO from the Properties view, but generally, most development and configuration is performed here.
Outline	An outline of the file that is currently open in an editor, listing structural elements. The contents are editor-specific.
Error Log	The error log captures Eclipse warnings and errors, including stack traces.
Problem	A view that displays problems, errors, or warnings.

3. To access the online help, click **Help > Help Contents** in the main menu bar.
4. Expand any of the documents that appear in the left pane.

   Some documents are for Sybase Unwired Platform, while others are for the Eclipse development environment.

Developing an iOS Application

Generate Object API code for the iOS platform, develop a universal iOS device application with code, and test its functionality. The device application communicates with the database MBOs that are deployed to Unwired Server.

Prerequisites

Complete these tasks:

1. Install Sybase Unwired Platform Mobile SDK and Runtime as indicated in *Getting Started* on page 5.
2. Complete *Tutorial: Mobile Business Object Development*, which provides the foundation tasks for this tutorial.
3. Open the SUP101 mobile application project. In WorkSpace Navigator, right-click the SUP101 folder and select **Open in Diagram Editor**.
4. Be sure you are using the Advanced developer profile.

Task

Note: This tutorial was developed using Mac OS X 10.6 (Snow Leopard), Xcode 4.0.2, and iOS SDK 4.3. If you use a different version of Xcode, 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. *Generating Object API Code*

   Launch the code generation wizard and generate the object API code for a message-based iOS application.
2. *Setting Up an iOS Client Application in Xcode*

   Set up an iOS client application in the Xcode IDE.
3. *Registering the Application Connection in Sybase Control Center*

   Register the iPhone Simulator in Sybase Control Center.
4. *Viewing the SUP101CallbackHandler File*

   In Xcode, view and understand the SUP101CallBackHandler file.
5. *Creating the User Interface*

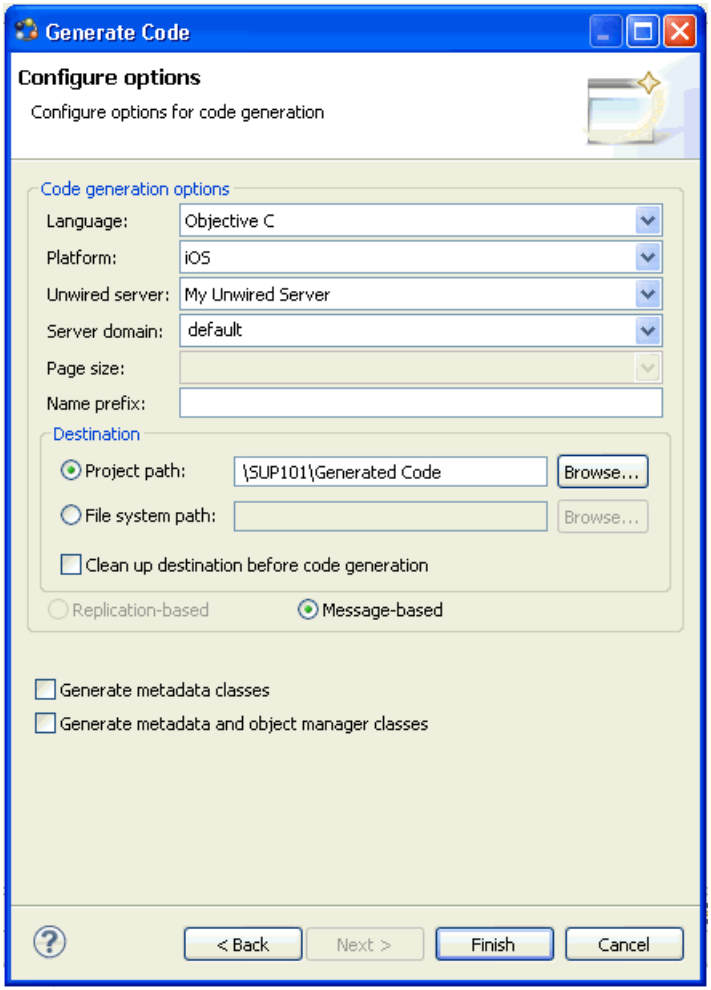
   Use Interface Builder to create and configure the user interface for the SUP101 application.
6. *Deploying the Device Application*

   Deploy the SUP101 application to the iPhone simulator for testing.

Generating Object API Code

Launch the code generation wizard and generate the object API code for a message-based iOS application.

1. Right-click in the SUP101 Mobile Application Diagram and select **Generate Code**.
2. In the code generation wizard, accept the default, **Continue without a configuration**, and click **Next**.
3. Make sure the Customer and Sales_order MBOs are selected, then click **Next**.
4. Enter these configuration options and click **Next**:



Option	Description
Language	Select **Objective C**.
Platform	Accept the default, **iOS**.
Unwired server	Select **My Unwired Server**.
Server domain	Accept **default**. If you are not connected to Unwired Server, this field is empty. Connect to Unwired Server to proceed.
Name prefix	The prefix for the generated files. Leave blank.
Project path	Accept the default or enter a different location for the generated project files.
(Optional) Clean up destination before code generation	Select this option to delete all items in the destination folder before generating the device client files.
Message-based	Default for Objective-C.

5. Click **Finish**.
   Objective-C code is generated into the specified output location.

Setting Up an iOS Client Application in Xcode

Set up an iOS client application in the Xcode IDE.

Prerequisites

- Generate Objective-C code in to an output location.
- Ensure the directory where Sybase Unwired Platform is installed is a shared directory so you can access it from your Mac.
- Obtain the header and Objective-C source code files you need to build the user interface from the `SUP_iOS_Custom_Dev_Tutorial_code.zip` file. This way, you can easily copy and paste the code into the corresponding files that are created in Xcode.
  - If you are viewing this guide as a PDF, you can obtain the files from the Sybase Product Documentation Web site at *http://sybooks.sybase.com/nav/summary.do?prod=1289&lang=en&submit=%A0Go%A0&prodName=Sybase+Unwired+Platform&archive=0*. Navigate to this topic in the tutorial, then click the link for the zip file to access the provided source code files.
  - If you are viewing this guide online from the Sybase Product Documention Web site, click *SUP_iOS_Custom_Dev_Tutorial_code.zip* to access the source code files.

Task

1. On your Mac, start Xcode and select **Create a new Xcode project**.
2. Select **iOS Application** and **Window-based Application** as the project template, and then click **Next**.
3. Enter `SUP101` as the **Product Name**, `MyCorp` as the **Company Identifier**, select **Universal** as the **Device Family** product.

   Unselect **Include Unit Tests**, and then click **Next**.
4. Select a location to save the project and click **Create** to open it.

   Xcode creates a folder, SUP101, to contain the project file, `SUP101.xcodeproj` and another SUP101 folder, which contains a number of automatically generated files.
5. Delete some of the automatically generated files created by default for the Xcode project.
   a) Delete `SUP101AppDelegate.h` and `SUP101AppDelegate.m`.
   b) Under **Supporting Files**, delete the `main.m` file.
6. Copy the files from the SUP101 folder on your Windows machine in to the SUP101 folder on your Mac that Xcode created to contain the the SUP101 project.
   a) Connect to the Microsoft Windows machine where Sybase Unwired Platform is installed
   b) From the Apple Finder menu, select **Go** > **Connect to Server**.
   c) Enter the name or IP address of the machine, for example, `smb://<machine DNS name>` or `smb://<IP Address>`, then click **Connect**.

      You see the shared directory.
   d) In the Unwired Platform installation on your Windows machine, navigate to the `\UnwiredPlatform\ClientAPI\MBS\ObjectiveC` directory in the Unwired Platform installation directory, and copy the `ObjectiveC` folder to the `SUP101/SUP101` directory on your Mac.
   e) On your Windows machine, navigate to the SUP101 mobile application project and copy the `Generated Code` folder to the SUP101 directory on your Mac.
   f) To prepare for adding the source code files to the Xcode project, unzip the `SUP_iOS_Custom_Dev_Tutorial_code.zip` archive.

Next

Add libraries, resources, and source code to the SUP101 Xcode project.

See also

- *Registering the Application Connection in Sybase Control Center* on page 16

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

Once you set up the initial project in Xcode, you need to add files from the Sybase Unwired Platform folders you copied from your Windows machine.

1. In the Xcode Project Navigator, **Control-click** the SUP101 folder, then select **Add Files to "SUP101"**.

   Select the `Generated Code` folder, unselect **Copy items into destination group's folder (if needed)**, and click **Add**.

   The `Generated Code` folder is added to the project in the Project Navigator.
2. **Control-click** the SUP101 folder, then select **Add Files to "SUP101"**.
   a) In the ObjectiveC folder you copied from the Sybase Unwired Platform installation, navigate to the `libs/Debug-iphonesimulator` directory.
   b) Select the `libclientrt.a`, `libSUPObj.a`, and `libMO.a` libraries.
   c) Be sure **Copy items into destination group's folder (if needed)** is unselected.
   d) Click **Add**.

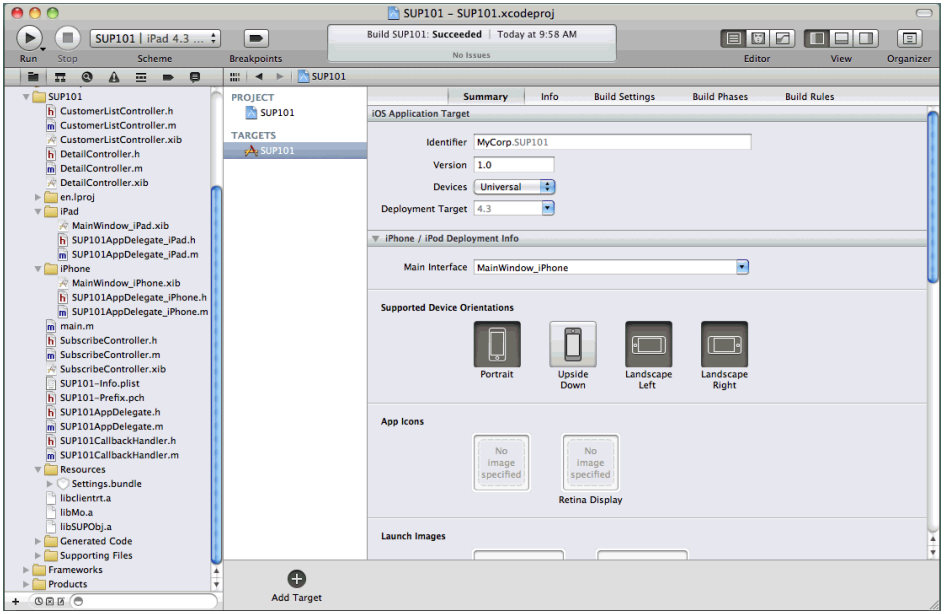
   The libraries are added to the project in the Project Navigator.

   Note: The library version corresponds to the configuration you are building. In this tutorial, you work with the libraries for the Debug version of the iPhone simulator.
3. **Control-click** the SUP101 folder, then select **New Group**, then rename it to **Resources**.
4. **Control-click** the Resources folder, then select **Add Files to "SUP101"** to add the `Settings.bundle` resource to the project.

   This bundle adds resources that lets iOS device client users input information such as server name, server port, user name and activation code in the Settings application.
   a) In the ObjectiveC folder, navigate to the **includes** directory, then select the `Settings.bundle` file.
   b) Unselect **Copy items into destination group's folder (if needed)**.
   c) Click **Add**

   The bundle `Settings.bundle` is added to the project in the Project Navigator.
5. Add the source code files from the `SUP_iOS_Custom_Dev_Tutorial_code.zip` archive.
   a) **Control-click** the SUP101 folder, then select **Add Files to "SUP101"**.
   b) Select all files, then click **Add**.

   The project now looks like this:



Next
Configure the build settings.

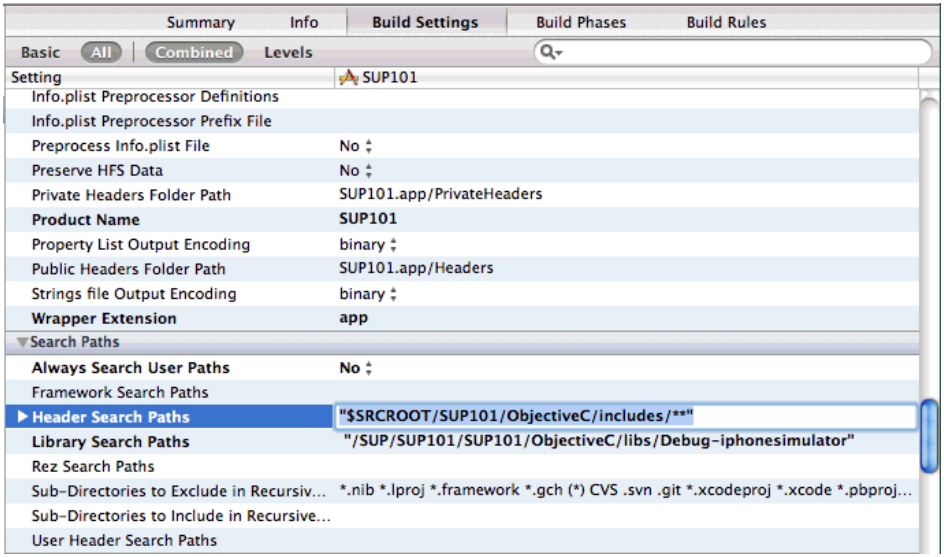
Configuring the Build Settings

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

1. In the right pane, click the **Build Settings** tab, then scroll down to the **Search Paths** section, then enter the location of the iPhone simulator libraries in the **Header Search Paths** and **Library Search Paths** fields.

   `$SRCROOT` is a macro that expands to the directory where the Xcode project file resides. This tutorial project was created in the `/SUP/SUP101` directory.

   - In **Header Search Paths**, enter the path to the `ObjectiveC/includes` directory. In this example, the path is indicated as `"$SRCROOT/SUP101/ObjectiveC/includes/**"`.
   - In **Library Search Paths**, enter the path to the `ObjectiveC/libs/Debug-iphonesimulator` directory. In this example, the path is indicated as `"$SRCROOT/SUP101/ObjectiveC/libs/Debug-iphonesimulator"`.

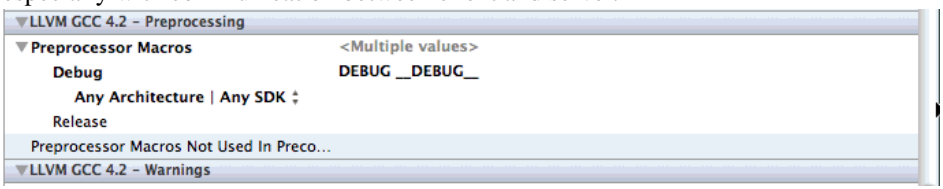


2. Since you are setting up this application on the iOS simulator for debugging, scroll down to Preprocessing to add the __DEBUG__ macro to the build settings for the SUP101 target.

   a) Click **Debug** to highlight it, then click the + icon that appears once you highlight that field.

   b) Enter `__DEBUG__` (two underscores, DEBUG, two underscores).

   The __DEBUG__ macro enables code in `MBODebugLogger.m` that prints message headers to the simulator console. This feature is useful for debugging many issues, especially with communication between client and server.



3. In the right pane, select the **Build Phases** tab, then expand the **Link Binary with Libraries** section.

   Click the + icon below the list, select the following libraries, and then click **Add** to add them from the SDK to the project:

   - `AddressBook.framework`
   - `CoreFoundation.framework`
   - `libicucore.A.dylib`
   - `libstdc++6.dylib`
   - `libz.1.2.3.dylib`

- `QuartzCore.framework`
- `Security.framework`

4. Select **Product > Clean**, then **Product > Build** to test the initial set up of the project. If you correctly followed this procedure, you see a **Build Succeeded** message.

Registering the Application Connection in Sybase Control Center

Register the iPhone Simulator in Sybase Control Center.

Prerequisites

Connect to Sybase Control Center.

Task

1. Log in to Sybase Control Center using the credentials you indicated during installation.
2. In Sybase Control Center, select **View > Select > Unwired Server Cluster Management View**.
3. In the left pane, select **Applications**.
4. In the right pane, click **Application Connections**.
5. Click **Register**.
6. In the Register Application Connection window, enter the required information, leaving the application ID and domain blank:
   - User name – `user1`
   - Server name – *<localhost.sybase.com>*

     Note: The information should match the input on the client and "localhost.sybase.com" should be the actual name of your machine and domain.
   - Port – the Unwired Server port, `5001`.
   - Farm ID – `0`
   - Activation code – `123`
7. Click **OK**.

Next

In Xcode, view the application source files and walk through how they are created.

See also

- *Setting Up an iOS Client Application in Xcode* on page 11

Viewing the SUP101CallbackHandler File

In Xcode, view and understand the `SUP101CallBackHandler` file.

`SUP101CallbackHandler` is a subclass of `SUPDefaultCallbackHandler`, and is used to listen for events sent from the server. The header, `SUP101CallbackHandler.h`, is referenced in a number of classes in this application, so you would create it first. You can create new Objective-C class files from the main menu: **File > New > New File**.

There are two threads involved in the SUP101 application — the main thread, which is driven by the client application user interface controller, and the mobile object client access thread, which takes charge of message transportation with the server and synchronization with the application through the mobile object. In iOS, all code that updates the user interface must be called on the main thread, so it is a good idea to send notifications that might trigger changes to the interface from the main thread.

1. Click the `SUP101CallbackHandler.h` file to view the provided source code.
2. Click the `SUP101CallbackHandler.m` file to view the provided source code.

Creating the User Interface

Use Interface Builder to create and configure the user interface for the SUP101 application.

The `SUP_iOS_Custom_Dev_Tutorial_code.zip` contains the source code for the user interface for the sample application. Although the user interface is already built once 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

- *Deploying the Device Application* on page 27

Viewing the SubscribeController View Controller

A view controller functions as the root view screen for the SUP101 mobile application.

When you create the user interface, you assign a target action to a control object — in this example a Subscribe button so that a message (the action) is sent to another object (the target) in response to a user event, for example, a touch on the button. The view controller manages and configures the view when asked.

In Xcode, you can create the view controller by creating a new file using the **UIViewController subclass**. Be sure to indicate **With XIB for user interface**. Xcode creates the corresponding `.h`, `.m`, and `.xib` files.

1. In the SUP101 Xcode project, click `SubscribeController.m` to view the logic for the view controller.
2. Click `SubscribeController.h` to view the header file.

See also

- *Creating the CustomerListController* on page 24
- *Adding the DetailController and Configuring the View* on page 25

Configuring the SUP101Appdelegate Files

The `SUP101Appdelegate.h` and `SUP101Appdelegate.m` files are created when you create the Xcode project; however, you deleted the automatically generated versions and replaced them with the ones added from the source code zip file.

Delegates extend the functionality of reusable objects. A delegate allows one object to send messages to another object specified as its delegate to ask for input, or to be notified when an event occurs.

The `applicationDidFinishLaunching` method contains code needed to start the application, register for notifications, and start up the messaging layer to begin communications with the server. The `onConnectSuccess` and `onConnectFailure` methods are notification handlers. When the the messaging layer calls the callback handler's `onConnectionStatusChange` method, indicating that communications are established, the ON_CONNECT_SUCCESS notification is sent. When it gets this notification, the app delegate runs `onConnectSuccess` to begin the login process. If communication cannot be established, `onConnectionFailure` runs.

1. View the `SUP101Appdelegate.h` file.
2. View the `SUP101Appdelegate.m` file and update the Sybase Unwired Platform administration credentials.

   Replace the bolded text with the administration credentials you indicated during installation.

```
-(void)onConnectSuccess:(NSNotification *)obj
{
    // Connection to the server was made, so log in.
    // See [CallbackHandler onLoginSuccess] and [CallbackHandler
onLoginFailure]. One of those
    // callbacks will be called at some point in the future.
    [[NSNotificationCenter defaultCenter] removeObserver:self
name:ON_CONNECT_SUCCESS object:nil];
    [[NSNotificationCenter defaultCenter] removeObserver:self
name:ON_CONNECT_FAILURE object:nil];
    [SUP101_SUP101DB beginOnlineLogin:@"supAdmin"
password:@"s3pAdmin"];
}
```

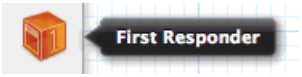
Configuring the SubscribeController View

Use Interface Builder to configure the `SubscribeController.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.

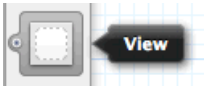
1. Click the `SubscribeController.xib` file to reveal a view of the (presently empty) screen in the right pane and the following three items represented by icons in the middle pane:
   - **File's Owner** – the object that is set to be the owner of the user interface, which is typically the object that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

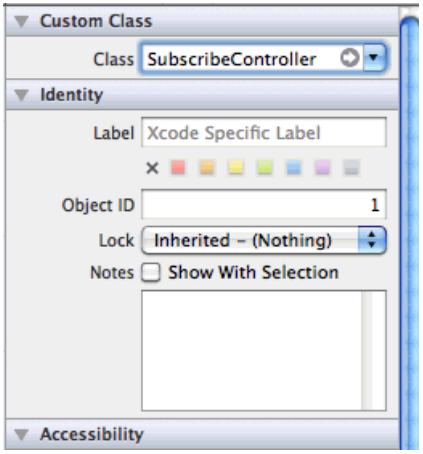
   - **First Responder** – the first responder proxy object handles events. Connecting an action to the first responder means that when the action is invoked, it is dynamically sent to the responder chain.

   - **View** – displayed in a separate window to allow you to edit it.

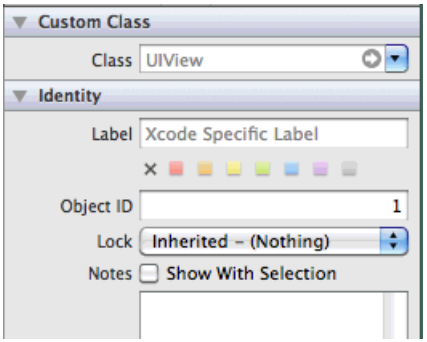
     

2. Select the **File's Owner** icon, select **View > Utilities > Identity Inspector**, and make sure `SubscribeController` appears in the **Class** field under **Custom Class**.

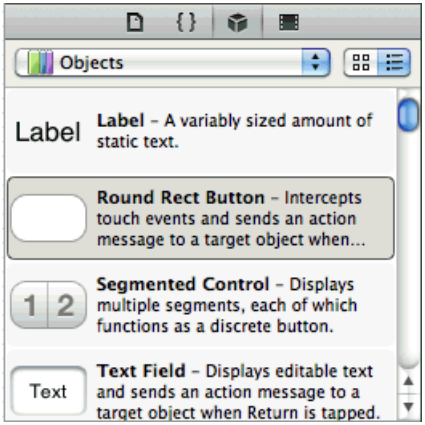
   

   This tells Interface Builder the class of the object to allow you to make connections to and from the File's Owner.

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

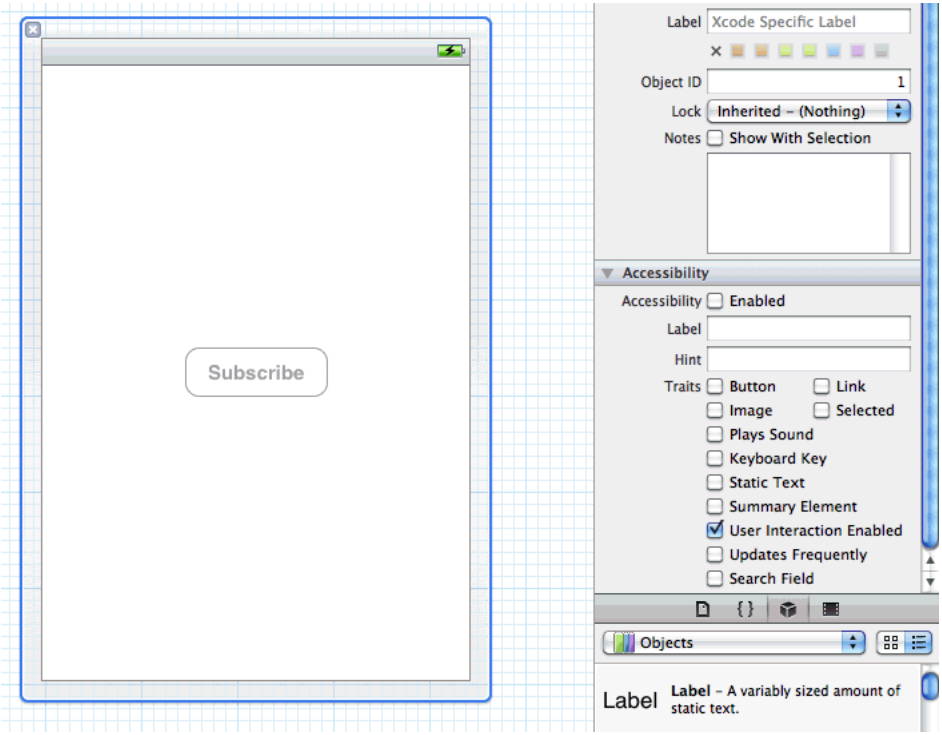


4. To create a Subscribe button, select **View** > **Utilities** > **Object Library**.
   a) In the Object Library panel, select the **Round Rect Button** item, drag it onto the view.

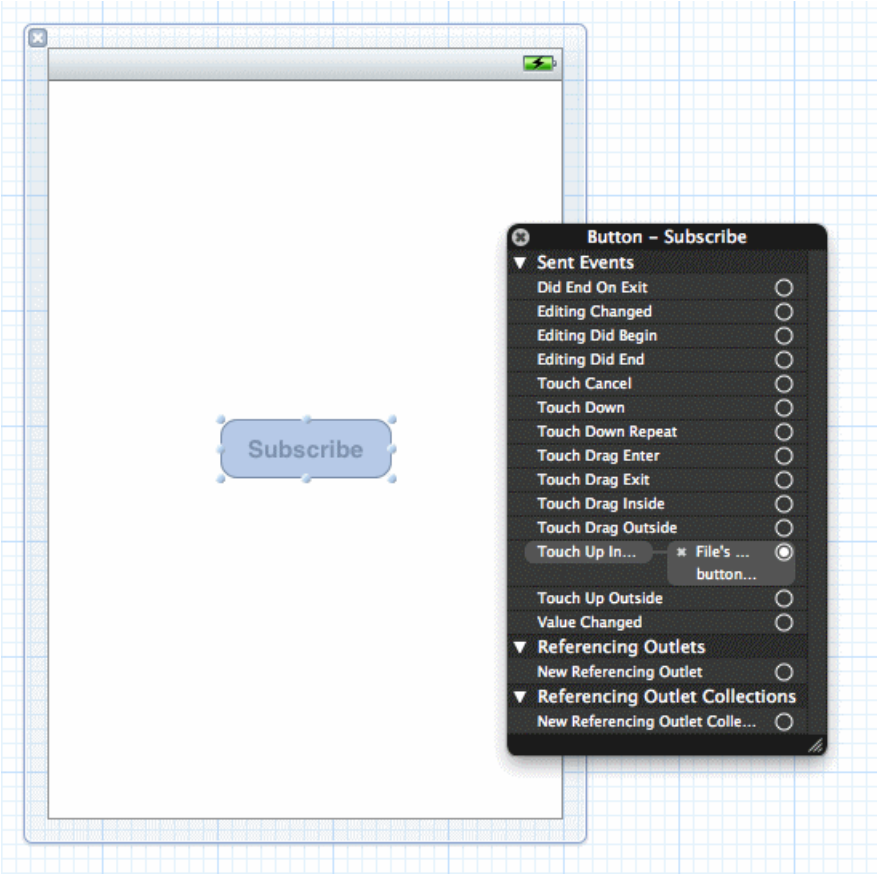


   b) Double-click it and enter `Subscribe` and press **Return**.
5. In the Accessibility section of the Identity Inspector, make sure **Enabled** is unselected.

   Disable the button because the application cannot subscribe to the server for updates until it is connected.



6. Control-click the **Subscribe** button to show the inspector.
7. Drag from the circle to the right of **Touch Up Inside** to the **File's Owner** icon and release, then click on **buttonPressed** to establish a connection between the **Subscribe** button and the button's action method:

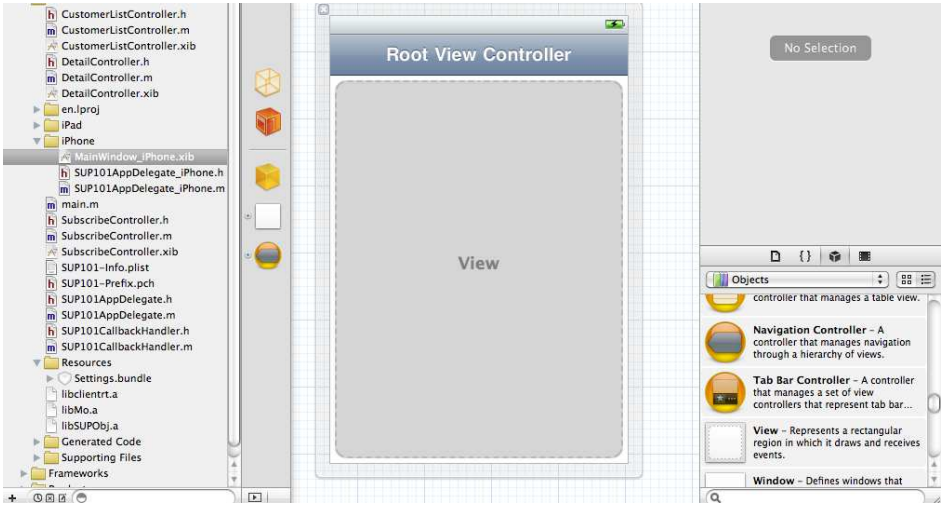


Making Connections

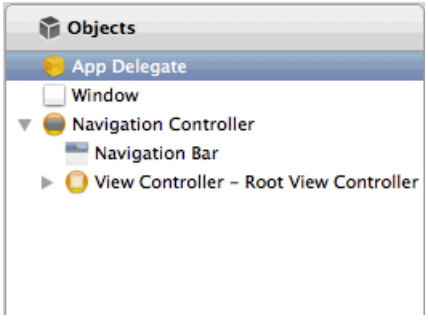
Add Navigation Controllers to `MainWindow_iPhone.xib` and `MainWindow_iPad.xib` and create a connection from the AppDelegate to the Navigation Controller.

1. In the left pane, under the iPhone folder, click the `MainWindow_iPhone.xib` file.

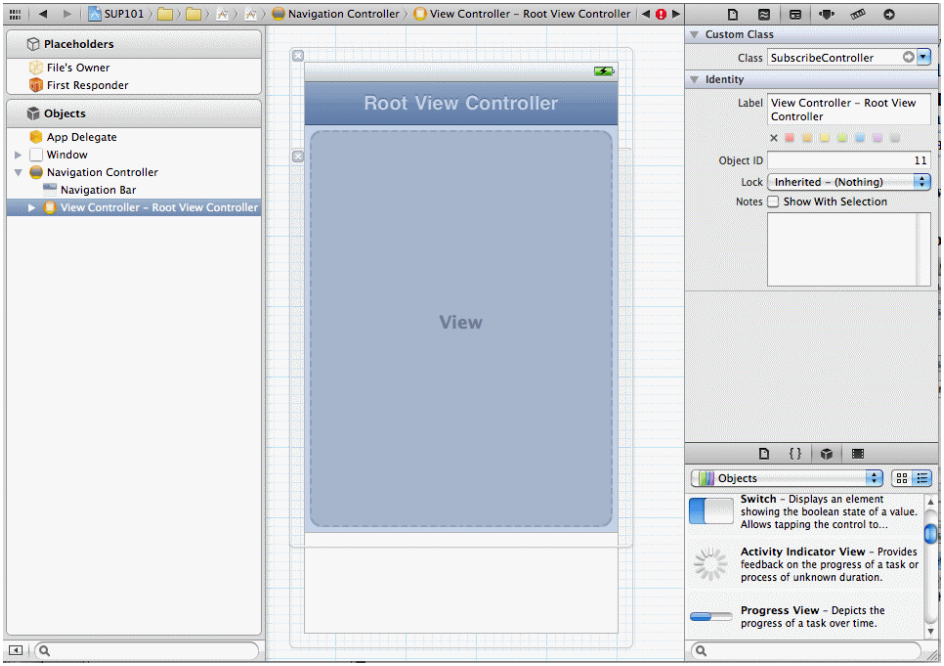
   If you do not see the Navigation Controller in the middle pane, drag it from **Objects** to the middle pane:



2. Under **Objects** in the middle pane, control-drag from the **AppDelegate** icon to the **Navigation Controller** icon to create a **navController** outlet.



3. Click on the expansion arrow at the bottom of the middle pane to switch to list view, select **Navigation Controller** > **View Controller**, and in the Identity Inspector, select `SubscribeController` in the **Class** field.



Once the class is selected, the **ViewController** name in the hierarchy changes to **SubscribeController** and the connection from the AppDelegate to the Navigation Controller is created.

4. Repeat these steps to add a navigation controller to `MainWindow_iPad.xib`.

Creating the CustomerListController

Create the customer list view.

The source files you added from the SUP_iOS_Custom_Dev_Tutorial_code.zip file contain the `CustomerListController.h`, `CustomerListController.m`, and `CustomerListController.xib` files that create the customer list view. To create these files manually in Xcode, you would create a new file using the **UIViewController subclass** template, then indicate it is a subclass of `UITableViewController`. Be sure to indicate **With XIB for user interface**.

1. View the `CustomerListController.h` file.
2. View the `CustomerListController.m` file.

   `CustomerListController.m` is a table view controller that displays the customer data in the client database. The `viewWillAppear` method uses the Object API to query the database for a list of all Customer objects, and builds an NSArray that is used by this class as the data source for displaying the table view.

If a row is tapped, the `accessoryButtonTappedForRowWithIndexPath` method is run, which pushes a `DetailController` onto the stack to display additional information and allow the data to be modified.

See also

- *Viewing the SubscribeController View Controller* on page 17
- *Adding the DetailController and Configuring the View* on page 25

Adding the DetailController and Configuring the View

Create the `DetailController.xib`.

The detail controller view displays information about a single customer in the client database. he source files you added from the SUP_iOS_Custom_Dev_Tutorial_code.zip file contain the `DetailController.h`, `DetailController.m`, and `DetailController.xib` files that create the customer detail view. To create these files manually in Xcode, you would create a new file using the **UIViewController subclass** template, then indicate it is a subclass of `UIViewController`. Be sure to indicate **With XIB for 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 `DetailController.xib` file to open Interface Builder.
2. Select **View** > **Utilities** > **Object Library**.
3. In the Object Library panel, select the **Text Field** item, drag it onto the view three times to create three text fields aligned vertically to the right of the screen.

   You can resize the text fields using the resize handles and position the button by dragging it to the desired location.
4. In the Object Library panel, select the **Label** item, drag it onto the view three times to create three labels to the left of and aligned with the three text fields. Replace the default `Label` text with:
   - First Name
   - Last Name
   - Phone
5. In the Object Library panel, select the **Round Rect Button** item, drag it onto the view, and rename it to `Submit`.

   To make connections to the user interface from the view controller, the `DetailController.h` file contains the outlets, property declarations for the instance variables, and a declaration for the action method.

```
#import <UIKit/UIKit.h>
#import "SUP101_Customer.h"

@interface DetailController : UIViewController {
}
```

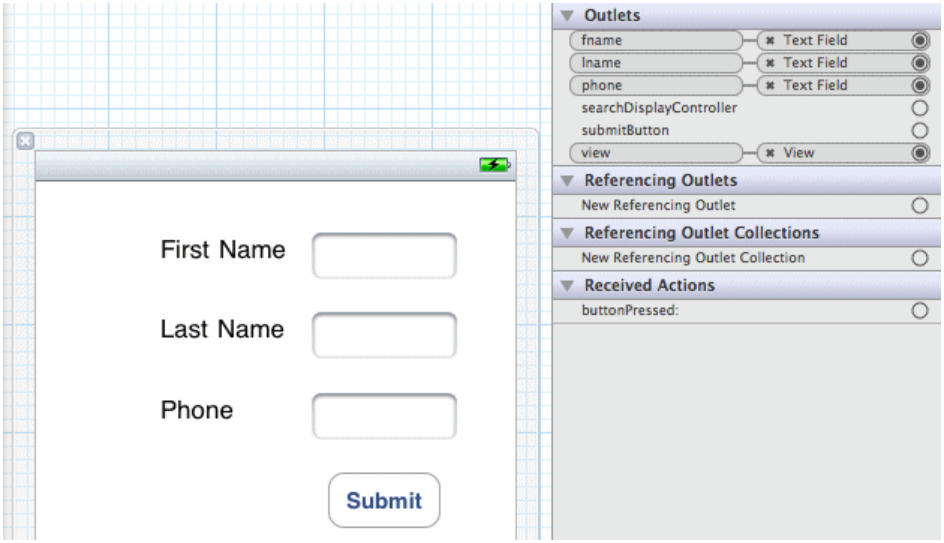
```
@property (nonatomic, retain) IBOutlet UITextField *fname;
@property (nonatomic, retain) IBOutlet UITextField *lname;
@property (nonatomic, retain) IBOutlet UITextField *phone;
@property (nonatomic, retain) SUP101_Customer *originalObj;
@property (nonatomic, retain) IBOutlet UIButton *submitButton;

-(IBAction)buttonPressed:(id)sender;

@end
```

6. View the `DetailController.m` file.

   This class displays detailed information about a single customer in the client database. The information can be edited. If the data is changed and the Submit button is pressed, the `buttonPressed` method uses Object API calls to save the changes in the client database, send the changes to the server, and disable the Submit button.

   If the server accepts the changes, the callback handler posts an ON_REPLAY_SUCCESS notification, which causes the `onReplaySuccess` notification handler to run. The cached UI data is refreshed from the database and the Submit button is re-enabled.

   This class also registers for the ON_REPLAY_FAILURE notification to handle the case where the server rejects the changes, or an error occurs on the server side.
7. Click the `DetailController.xib` file to open it in Interface Builder, click the **First Name** text field, and select **View** > **Utilities** > **Attributes Inspector**.
8. In the Attributes Inspector panel , scroll to the **View** section and enter `1` in the **Tag** field.
9. Set the tags for the **Last Name** and **Phone** text fields to `2` and `3` respectively.
10. Control-drag from the **File's Owner** icon in the middle pane to each of the text fields and select the **fname**, **lname**, and **phone** outlets, respectively, to create connections between the text fields and the outlets defined in the `DetailController.m` file.
11. Select **View** > **Utilities** > **Connections Inspector** to confirm that the outlets have been correctly configured:



12. Control-drag from the **File's Owner** icon in the middle pane to the **Submit** button and select **submitButton**.

See also

- *Viewing the SubscribeController View Controller* on page 17
- *Creating the CustomerListController* on page 24

Deploying the Device Application

Deploy the SUP101 application to the iPhone simulator for testing.

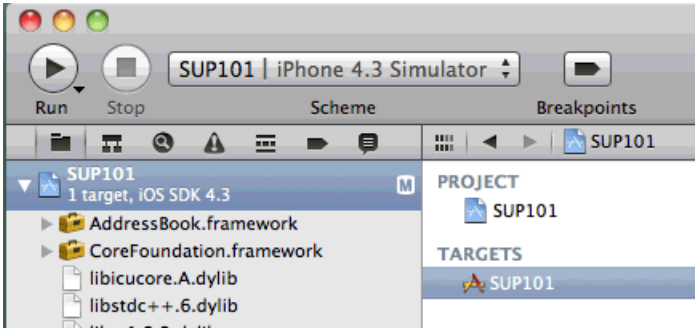
Prerequisites

Register an application connection in Sybase Control Center.

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

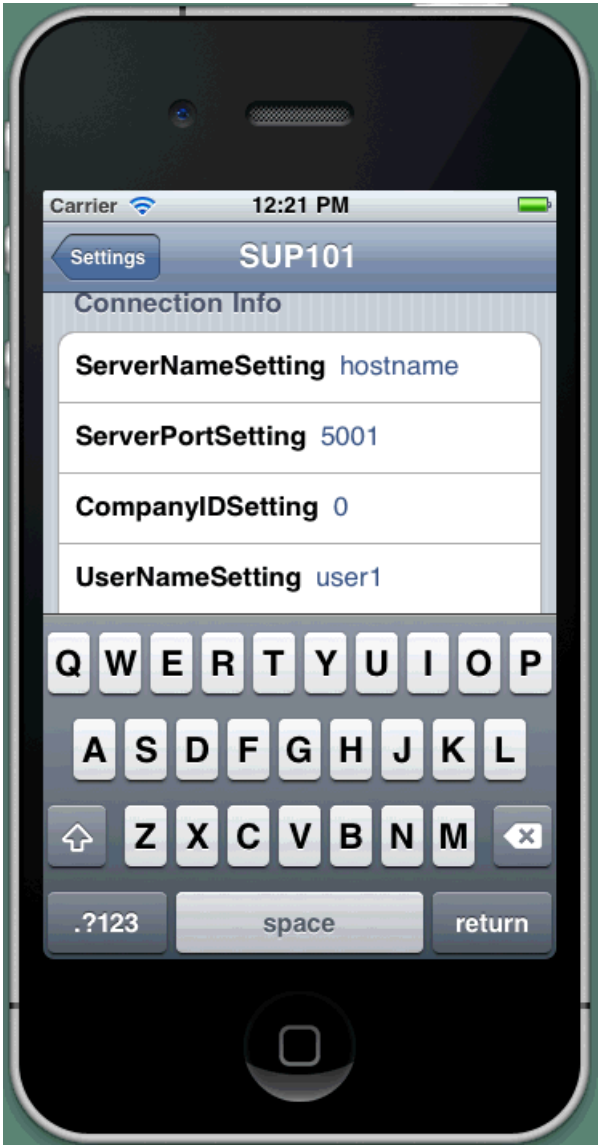
Task

1. In the upper-left corner of XCode, make sure that the **Scheme** is set to **SUP101 | iPhone 4.3 Simulator**.



2. Select **Product** > **Build** and then **Product** > **Run**

   The project is built and the iPhone simulator starts.
3. In the iPhone simulator, go to **Settings** > **SUP101** to enter the connection settings.
   - **ServerNameSetting** – the machine that hosts the server where the SUP101 mobile application project is deployed.
   - **ServerPortSetting** – Unwired Server port number. The default is **5001**.
   - **CompanyIDSetting** – the company ID you entered when you registered the device in Sybase Control Center, in this case, `0`.
   - **UserNameSetting** – the user you registered in Sybase Control Center, `user1`.
   - **ActivationCodeSetting** – the activation code for the user, `123`.



4. In the iPhone applications screen, open the **SUP101** application.
5. Click **Subscribe**.
   The customer list appears.
6. Select a customer record from the customer list and double-click to open the detail view.
   The customer detail shows the fields: **First Name**, **Last Name**, and **Phone**.
7. Change the **First Name** to something else, and click **Submit**.

See also

- *Creating the User Interface* on page 17

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

Tutorials

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

Tutorial Projects

Tutorial projects are available for download, if you want the finished tutorial without going through the steps. Download tutorial projects from: *http://www.sdn.sap.com/irj/sdn/mobile?rid=/webcontent/uuid/40ea4956-b95c-2e10-11b3-e68c73b2280e*.

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 the Product Documentation Web site.

Developer Guides

Learn about using the API to create device applications:

- *Developer Guide: BlackBerry Native Applications*
- *Developer Guide: iOS Native Applications*
- *Developer Guide: Windows and Windows Mobile Native Applications*
- *Developer Guide: Mobile Workflow Packages*

Customize and automate:

- *Developer Guide for Unwired Server Management API* – customize and automate system administration features.
- *Developer Guide: Unwired Server* – customize and automate server-side implementations for device applications, and administration, such as data handling.

Javadoc and HeaderDoc are also available in the installation directory.

Index

T

U

V

W

X