Tutorial: Windows Mobile Application Development

Sybase Unwired Platform 2.1 ESD #2

DOCUMENT ID: DC01285-01-0212-01
LAST REVISED: March 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.

Tip: If you want to see the final outcome of a tutorial without performing the steps, the associated example project is available on SAP® SDN: *http://www.sdn.sap.com/irj/sdn/mobile?rid=/webcontent/uuid/40ea4956-b95c-2e10-11b3-e68c73b2280e*.

- 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*
- 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 Mobile Object API Application Development*
- Create a mobile business object, then develop a mobile workflow package that uses it:
  - *Tutorial: Mobile Workflow Package Development*

Task Flow

Use this tutorial to develop, deploy, and test a mobile device application on a simulator or an emulator.

Task	Goals	Steps
Getting started	• Install Sybase Mobile SDK and Sybase Unwired Platform Runtime. • Start Unwired Server and other platform services, if not already started. • Start Sybase Control Center. • Start Sybase Unwired WorkSpace, open the Mobile Development perspective, and become familiar with the views of the perspective and the Mobile Application Diagram.	• Installing Sybase Unwired Platform • Starting Unwired Platform Services • Starting Sybase Unwired WorkSpace • Connecting to Sybase Control Center • (Optional) Learning Unwired WorkSpace Basics These steps are prerequisites for the rest of this tutorial. You need to perform them only once.
Developing database mobile business objects	• Create a mobile application project and a connection to the database. • Create two mobile business objects, and create a relationship between them. • Deploy the mobile business objects to Unwired Server.	Complete the *Tutorial: Mobile Business Object Development*, or obtain the completed example project. **Note:** This tutorial is a prerequisite for the remaining steps. You need to perform it only once. If you want to download the final outcome of a tutorial without performing it, the associated example project is available on SAP® SDN: *http://www.sdn.sap.com/irj/sdn/mobile?rid=/webcontent/uuid/40ea4956-b95c-2e10-11b3-e68c73b2280e*

Task	Goals	Steps
Developing a native device application	- Generate code for the Windows Mobile platform, code a Windows Mobile device application, and run it on an emulator.	- Generating C# Object API Code - Creating the User Interface for the Windows Mobile Device Application - Deploying and Running the Device Application

Getting Started with Unwired Platform

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 Mobile SDK and Sybase 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. *Starting Sybase Unwired WorkSpace*

   Start the development environment, where you can perform tasks that include creating mobile business objects (MBOs), managing database and server connections, developing Mobile Workflow applications, and generating Object API code.
4. *Connecting to Sybase Control Center*

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

   Sybase Unwired WorkSpace features are well integrated in the Eclipse IDE. If you are not familiar with Eclipse, you can quickly learn the basic layout of Unwired WorkSpace and the location of online help.

Installing Sybase Unwired Platform

Install Sybase Mobile SDK and Sybase Unwired Platform Runtime.

Before starting this tutorial, be sure you have all the requisite Unwired Platform components installed. For complete installation instructions, see the Sybase Unwired Platform documentation at *http://sybooks.sybase.com/nav/summary.do?prod=1289*:

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

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

Starting Sybase Unwired Platform Services

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

How you start Unwired Platform services depend on the options you selected during installation. In some cases, you may need to manually start Unwired Platform services.
Select **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 the development environment, where you can perform tasks that include creating mobile business objects (MBOs), managing database and server connections, developing Mobile Workflow applications, and generating Object API code.
Select **Start > Programs > Sybase > Unwired Platform > Unwired WorkSpace**.
The Sybase Unwired WorkSpace opens in the Mobile Development perspective. The Welcome page displays links to the product and information.

Next

To read more about Unwired WorkSpace concepts and tasks, select **Help > Help Contents**.

Connecting to Sybase Control Center

Open the 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 application connections
- Set role mappings

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

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

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

2. In Sybase Control Center, 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.

Learning Unwired WorkSpace Basics

Sybase Unwired WorkSpace features are well integrated in the Eclipse IDE. If you are not familiar with Eclipse, you can quickly learn the basic layout of Unwired WorkSpace and the location of online help.

- To access the online help, select **Help > Help Contents**. Some documents are for Sybase Unwired Platform, while others are for the Eclipse development environment.
- The Welcome page provides links to useful information to get you up and running.
  - Reopen the Welcome page by selecting **Help > Welcome**.
  - To close the Welcome page, click **X**.
  - To learn about tasks you must perform, select the **Development Process** icon.
- In Unwired WorkSpace, look at the area (window or view) that you will use to access, create, define, and update mobile business objects (MBOs).

Window	Description
WorkSpace Navigator view	Use this view to create Mobile Application projects, and review and modify MBO-related properties. This view displays mobile application project folders, each of which contains all project-related resources in subfolders, including MBOs, datasource references to which the MBOs are bound, personalization keys, and so on.
Enterprise Explorer view	A view that provides functionality to connect to various enterprise information systems (EIS), such as database servers, SAP® back ends, and Unwired Server.

Window	Description
Mobile Application Diagram	The Mobile Application Diagram is a graphical editor where you create and define mobile business objects. 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 or defer binding to a data source, when creating an MBO. For example, you may want to model your MBOs before creating the data sources to which they bind. This MBO development method is sometimes referred to as the top-down approach. • Drag items from Enterprise Explorer and drop them (drag and drop) onto the Mobile Application Diagram to create the MBO – quickly creates the operations and attributes automatically based on the datasource artifact being dropped on the Mobile Application Diagram. Each new mobile application project generates an associated mobile application diagram.
Palette	The Palette is accessed from the Mobile Application Diagram and provides controls, such as the ability to create MBOs, add attributes and operations, and define relationships, by dragging-and-dropping the corresponding icon onto the Mobile Application Diagram or existing MBO.
Properties view	Select an object in the Mobile Application Diagram to display and edit its properties in the Properties view. While you cannot create an MBO from the Properties view, most development and configuration is performed here.
Outline view	Displays an outline of the active file and lists structural elements. The contents are editor-specific.
Problems view	Displays problems, errors, or warnings that you may encounter. This is a valuable source for collecting troubleshooting information.

Window	Description
Error Log view	Displays error log information. This is a valuable source for collecting troubleshooting information.

Developing a Windows Mobile Device Application

Generate code for the Windows Mobile platform based on the MBOs, add additional code to develop a Windows Mobile device application, and test its functionality.

Prerequisites

- *Getting Started* on page 5
- Install Visual Studio 2008.
- Install Microsoft .NET Compact Framework 3.5 from *http://www.microsoft.com/downloads/en/details.aspx?FamilyID=e3821449-3c6b-42f1-9fd9-0041345b3385&displaylang=en*, if it was not installed with Visual Studio.
- *Tutorial: Mobile Business Objects Development*, which provides the foundation tasks for this tutorial.
- In Unwired WorkSpace, open the SUP101 mobile application project. In WorkSpace Navigator, right-click the SUP101 folder and select **Open in Diagram Editor**.

Task

The device application communicates with the database mobile business objects that are deployed to Unwired Server.

1. *Installing Microsoft Synchronization Software*

   Install and configure Microsoft synchronization software so you can deploy and run a mobile application on a Windows Mobile emulator.
2. *Generating C# Object API Code*

   Generate Object API code for Windows Mobile devices.
3. *Creating the User Interface for the Windows Mobile Device Application*

   Import the SUP101 project in to Visual Studio 2008, configure the project, then download the tutorial code snippets from Product Documentation so you can create the application user interface.
4. *Deploying and Running the Device Application*

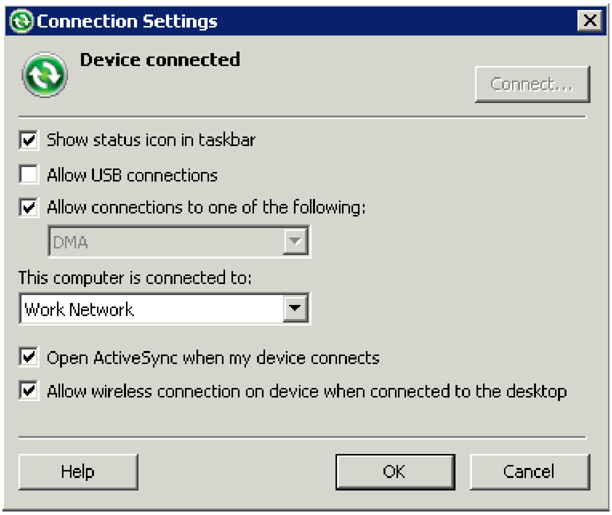
   Deploy the device application to a Windows Mobile 6 device emulator, and test its functionality.

Installing Microsoft Synchronization Software

Install and configure Microsoft synchronization software so you can deploy and run a mobile application on a Windows Mobile emulator.

Note: These instructions indicate how to install Microsoft ActiveSync for Windows XP. If you are using Windows Vista, Windows 7, or Windows 2008, install Virtual PC 2007 SP1 and Windows Mobile Device Center to manage synchronization settings. Download the Windows Mobile Device Center from *http://www.microsoft.com/windowsmobile/en-us/downloads/microsoft/device-center-download.mspx* and follow Microsoft instructions for installing and using that software instead of this procedure.

1. Download Microsoft ActiveSync:
   a) In a Web browser, open the Windows Phone page at *http://www.microsoft.com/windowsmobile/en-us/help/synchronize/device-synch.mspx*.
   b) Follow the instructions to select and download the sync software for the system's operating system. Windows XP requires ActiveSync version 4.5.
   c) In the Windows Phone downloads page, click the **ActiveSync** button.
   d) In the ActiveSync page, download the ActiveSync install file and save it to your local system.
2. Run the downloaded install file.
   For example, double-click `setup.msi` in Windows Explorer.
3. When the installation is complete, restart the system.
4. Start ActiveSync if it does not start automatically.
   For example, click **Start** > **Programs** > **ActiveSync**.
5. In ActiveSync, click **File** > **Connection Settings**.
6. Select **Allow connections to one of the following**, then select **DMA**.
7. For the option, **This computer is connected to**, select **Work Network**.



☑ Show status icon in taskbar

□ Allow USB connections

☑ Allow connections to one of the following:

DMA

This computer is connected to:

Work Network

☑ Open ActiveSync when my device connects

☑ Allow wireless connection on device when connected to the desktop

8. Click **OK**.

Generating C# Object API Code

Generate Object API code for Windows Mobile devices.

Prerequisites

1. Connect to both the sampledb database and Unwired Server. Code generation fails if the server-side (run-time) enterprise information system (EIS) data sources referenced by the MBOs in the project are not running and available to connect to when you generate object API code.
2. Switch to the Advanced developer profile to see advanced options in Unwired WorkSpace.

Task

1. In Unwired WorkSpace, open the SUP101 mobile application project.

   In WorkSpace Navigator, right-click the SUP101 folder and select **Open in Diagram Editor**.
2. Right-click the SUP101 - Mobile Application Diagram and select **Generate Code**.
3. On the **Code generation configuration** screen, click **Next**.
4. Select the Customer MBO, then click **Next**.
5. Enter the information for these configuration options:

Option	Description
Language	Select **C#**.
Platform	Select **NET Compact Framework 3.5 for Windows Mobile**.
Unwired Server	Select **My Unwired Server**.
Server domain	Select **default**.
Page size	Leave blank.
Namespace	Enter `SUP101`.
Destination	Accept the default. The default location for generated code is `\SUP101\Generated Code`.

6. Click **Finish**.

   A success notification dialog might appear. If so, click **OK**. Note the location of the resultant code file -- you will import this file into Visual Studio in the next task.

Creating the User Interface for the Windows Mobile Device Application

Import the SUP101 project in to Visual Studio 2008, configure the project, then download the tutorial code snippets from Product Documentation so you can create the application user interface.

Prerequisites

- Install Windows Mobile 6 Professional SDK from *http://www.microsoft.com/downloads/en/details.aspx?FamilyID=06111a3a-a651-4745-88ef-3d48091a390b&displaylang=en*.
- Obtain the text files that contain the code snippets you need to build the user interface from the `SUP_WM_Custom_Dev_Tutorial_code.zip` file. The code snippets are used to create the `Customer`, `CustomerDetails`, and `CustomerUpdate C#` files.
  - If you are viewing this guide as a PDF, you can obtain the text 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*. Select the appropriate version, navigate to this topic in the tutorial, then click the link for the zip file to save the files locally.
  - If you are viewing this guide online from the Sybase Product Documentation Web site, click *SUP_WM_Custom_Dev_Tutorial_code.zip* to access the text files.

Task

In the prior task, you generated C# Object API code from the MBO project. That file is located in your Eclipse workspace under `..\SUP101\Generated Code\src`.

1. In Visual Studio 2008, open `SUP101.csproj`.
2. Edit the project properties.
   a) In Solution Explorer, right-click SUP101, then select **Change Target Platform**. Select **Windows Mobile 6 Professional SDK**, then **OK**.
   b) Right-click SUP101, then select **Properties**.
   c) In the Application tab, select **Console Application** for **Output type**.
   d) In the Devices tab, select **Windows Mobile 6 Professional Emulator** for the **Target device**.
   e) Select **File** > **Save All**, then save the solution as `SUP101.sln`.

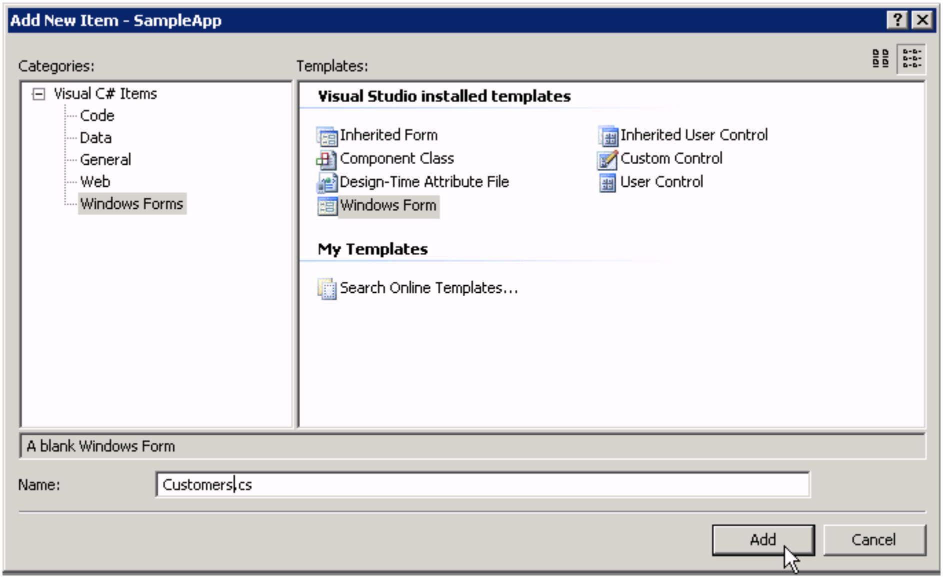
See also

- *Deploying and Running the Device Application* on page 26

Creating the Customers Form

Create the user interface for the Customers form.

1. In Solution Explorer, right-click the SUP101 project and select **Add** > **New Item**.
2. In the Add New Item Categories section, select **Windows Forms**. From the Templates section, select **Windows Form**. Enter `Customers` as the form name and click **Add**.



Add New Item - SampleApp

Categories: Visual C# Items > Windows Forms

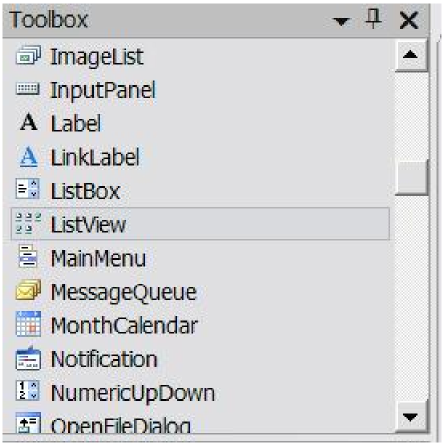
Templates: Windows Form

A blank Windows Form

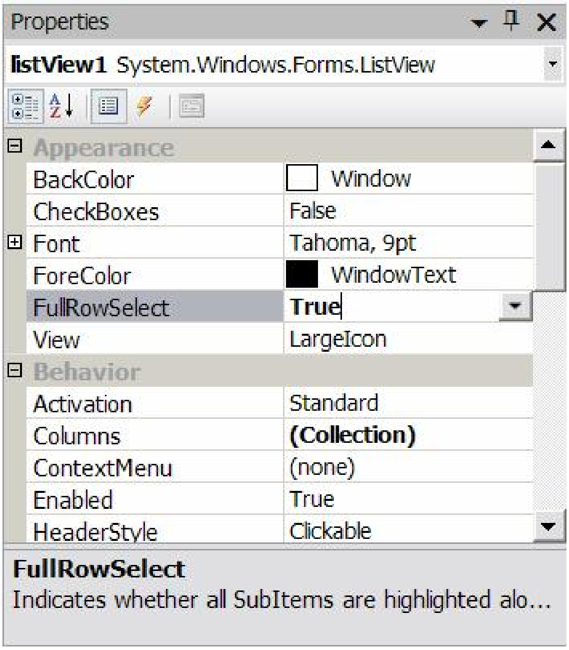
Name: Customers.cs

An empty form, **Customers**, displays on the **Customers.cs [Design]** tab.

3. From the Toolbox, drag and drop three buttons on to the form.
4. Select each button, and in the Properties view, change the **Text** of the buttons to:
   - button1 – `InitializeApplication`
   - button2 – `LoadData`
   - button3 – `Update Customer`
5. In the Toolbox, select **ListView** from **Common Device Controls**, and drag and drop it to the Customers form.



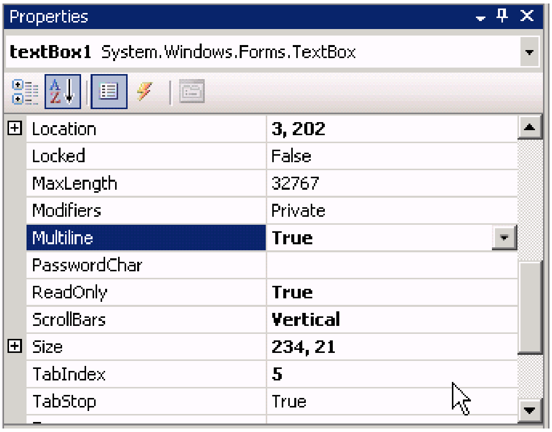
6. In the Toolbox, select **Textbox** from **Common Device Controls**, and drag and drop it to the Customers form.
7. In the **Customers** form, click the **ListView**, then in the Properties pane, set **FullRowSelect** to **True**.



Appearance	
BackColor	□ Window
CheckBoxes	False
Font	Tahoma, 9pt
ForeColor	■ WindowText
FullRowSelect	True
View	LargeIcon
Behavior	
Activation	Standard
Columns	(Collection)
ContextMenu	(none)
Enabled	True
HeaderStyle	Clickable

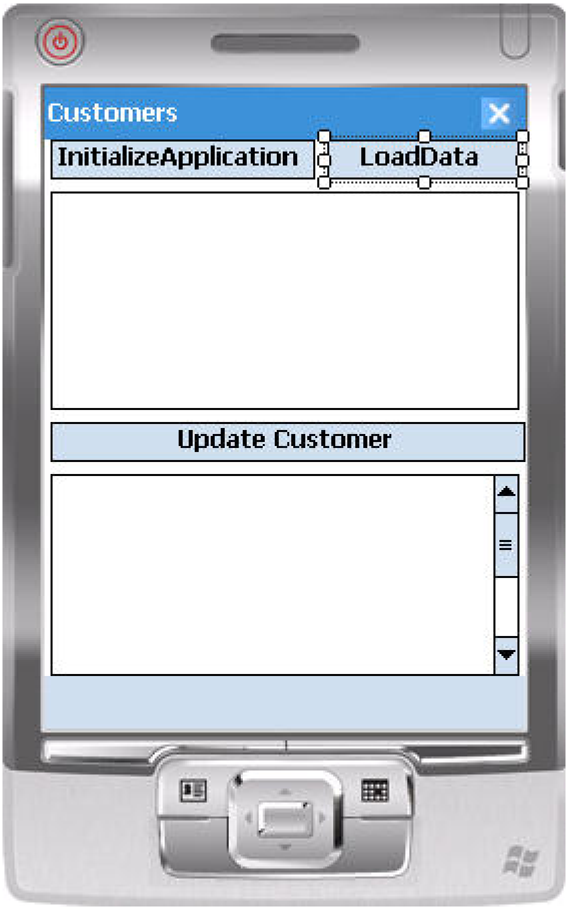
FullRowSelect
Indicates whether all SubItems are highlighted alo…

8. In **Customers**, select the **Textbox**, then in the Properties pane, set these properties:
   - Multi-line – True
   - Read-only – True
   - Scrollbars – Vertical



Location	3, 202
Locked	False
MaxLength	32767
Modifiers	Private
Multiline	True
PasswordChar	
ReadOnly	True
ScrollBars	Vertical
Size	234, 21
TabIndex	5
TabStop	True

9. Arrange the controls on the Customers form so they look like this:



10. Save the `Customers.cs` form.
11. In Solution Explorer, in the SUP101 project, right-click `Customers.cs`, then select **View Code**.
12. Replace the code with the source code from the `Customers.cs` file you downloaded from the Sybase Product Documentation Web site, also provided below:

```
using System;

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using SUP101;
using Sybase.Mobile;
using Sybase.Persistence;
using Sybase.Collections;
```

```
namespace SUP101
{
    public delegate void DelegateAddString(String s);
    public delegate void DelegateRefreshItem(long i);
    public partial class Customers : Form
    {
        private const String USERNAME = "supAdmin";
        private const String PASSWORD = "s3pAdmin";
        private const String HOST = "...";
        private const int PORT = 5001;
        private const int TIMEOUT = 600;
        private System.Collections.Generic.Dictionary<string, 
long> IdToSK = new System.Collections.Generic.Dictionary<string, 
long>();
        internal static 
System.Collections.Generic.Dictionary<long, int> SKToIndex = new 
System.Collections.Generic.Dictionary<long, int>();

        DelegateAddString m_DelegateAddString;

        DelegateRefreshItem m_refreshItem;

        public Customers()
        {
            InitializeComponent();
            m_DelegateAddString = new 
DelegateAddString(this.AddString);
            m_refreshItem = new 
DelegateRefreshItem(this.RefreshListItem);
        }

        internal void RefreshListItem(long sk)
        {
            int index = SKToIndex[sk];
            listView1.BeginUpdate();
            ListViewItem item = listView1.Items[index];
            String id = item.Text;

            Customer thisCustomer = 
Customer.FindByPrimaryKey(Int32.Parse(id));
            item.SubItems[1].Text = thisCustomer.Fname;
            item.SubItems[2].Text = thisCustomer.Lname;
            listView1.EndUpdate();
        }

        private void AddString(String s)
        {
            textBox1.Text += s + "\r\n";
            textBox1.SelectionStart = textBox1.Text.Length;
            textBox1.ScrollToCaret();
            textBox1.Refresh();
        }

        private void InitializeApplication_Click(object sender, 
EventArgs e)
```

```
        {
            Sybase.Mobile.Application app = 
Sybase.Mobile.Application.GetInstance();
            app.ApplicationIdentifier = "SUP101";
            MyCallbackHandler.textBox1 = textBox1;
            SUP101DB.RegisterCallbackHandler(new 
MyCallbackHandler());
            SUP101DB.SetApplication(app);
          SUP101DB.GetSynchronizationProfile().ServerName = HOST;

            ConnectionProperties connProps = 
app.ConnectionProperties;
            LoginCredentials loginCredentials = new 
LoginCredentials(USERNAME, PASSWORD);

            connProps.LoginCredentials = loginCredentials;
            connProps.ServerName = HOST;
            connProps.PortNumber = PORT;

            if (app.RegistrationStatus != 
RegistrationStatus.REGISTERED)
            {
                AddString("Application registering ... ");
                app.RegisterApplication(TIMEOUT);
                AddString("Application registered");
            }
            else
            {
                AddString("Connecting to server ...");
                app.StartConnection(TIMEOUT);
                AddString("Connected to server");
            }

            if (!SUP101DB.IsSynchronized("default"))
            {
                SUP101DB.DisableChangeLog();
                AddString("Package synchronizing ...");
                SUP101DB.Synchronize(); // Initial Synchronize

                ISynchronizationGroup sg = 
SUP101DB.GetSynchronizationGroup("default");
                sg.EnableSIS = true;
                sg.Save();
                SUP101DB.Synchronize();
                AddString("Package synchronized");
            }
            SUP101DB.EnableChangeLog();
            AddListView();
        }

        private void LoadData_Click(object sender, EventArgs e)
        {
            if 
(Sybase.Mobile.Application.GetInstance().RegistrationStatus == 
RegistrationStatus.REGISTERED)
            {
                Cursor.Current = Cursors.WaitCursor;
                AddString("Loading data from database ...");
                AddDataToListView();
                Cursor.Current = Cursors.Default;
            }
            else
            {
                AddString("Application is not initialized!");
            }
        }

        private void AddListView()
        {
            this.listView1.Clear();
            listView1.Columns.Add("Id", listView1.Width / 4, 
HorizontalAlignment.Left);
           listView1.Columns.Add("First Name", listView1.Width / 
3, HorizontalAlignment.Center);
          listView1.Columns.Add("Last Name", listView1.Width / 3, 
HorizontalAlignment.Right);
            listView1.View = View.Details;
            listView1.FullRowSelect = true;
        }

        private void AddDataToListView()
        {
            this.listView1.Clear();
            listView1.Columns.Add("Id", listView1.Width / 4,
            HorizontalAlignment.Left);
            listView1.Columns.Add("First Name", listView1.Width /
            3, HorizontalAlignment.Center);
           listView1.Columns.Add("Last Name", listView1.Width / 3,
            HorizontalAlignment.Right);
            listView1.View = View.Details;
            listView1.FullRowSelect = true;


            Query query = new Query();
           query.Select("x.fname, x.lname, x.surrogateKey, x.id");
            query.From("Customer", "x");
            query.OrderBy("id", SortOrder.ASCENDING);

            int index = 0;
            QueryResultSet rs = SUP101DB.ExecuteQuery(query);
            while (rs.Next())
            {
                String fname = rs.GetString(1);
                String lname = rs.GetString(2);
                long sk = rs.GetLong(3);
                String id = rs.GetString(4);

                ListViewItem item = new ListViewItem(id);
                item.SubItems.Add(fname);
                item.SubItems.Add(lname);
                listView1.Items.Add(item);
                IdToSK.Add(id, sk);
                SKToIndex.Add(sk, index++);
            }
        }
        private void update_Click(object sender, EventArgs e)
        {
            if (listView1.FocusedItem != null)
            {
                Program.setCustomer(listView1.FocusedItem.Text);
                Program.getForm2().Visible = true;
                Program.getForm1().Visible = false;
            }
            else
                MessageBox.Show("Please select a row");
        }
        private void Send_Click(object sender, EventArgs e)
        {
            Customer.SubmitPendingOperations();
        }
        public class MyCallbackHandler :
Sybase.Persistence.DefaultCallbackHandler
        {
            public static TextBox textBox1 = new TextBox();

            private void invokeDelegate(long i)
            {
                Customers f = Program.getForm1();
                f.Invoke(f.m_refreshItem, new Object[] { i });
            }

            public override SynchronizationAction
OnSynchronize(GenericList<ISynchronizationGroup> groups,
SynchronizationContext context)
            {

                if (context.Status ==
SynchronizationStatus.FINISHING || context.Status ==
SynchronizationStatus.ASYNC_REPLAY_UPLOADED)
                {
                    Query query = new Query();
                    GenericList<IChangeLog> changeLogs =
SUP101DB.GetChangeLogs(query);
                    foreach (IChangeLog changeLog in changeLogs)
                    {
                        if (changeLog.EntityType ==
EntityType.Customer)
                        {
                          invokeDelegate(changeLog.SurrogateKey);
                        }
                    }
                }

                return SynchronizationAction.CONTINUE;
            }
        }
```

```
    }
}
```

13. Be sure the bolded code line matches the Sybase Unwired Platform Admin login, password, and host you indicated during installation.

```
private const String USERNAME = "supAdmin";
  private const String PASSWORD = "s3pAdmin";
  private const String HOST = "…";
```

14. Click the Customers.cs[Design] tab to go back to the Customers form design view to add event handlers to the buttons.

    a) Click the InitializeApplication button on the form.

    b) In the Properties view for the button, click the Event icon (lightning bolt).

    c) Next to the Click databinding, select **InitializeApplication_Click**.

    d) Repeat these steps for each button, selecting these events for each Click databinding.

    - LoadData – **LoadData_Click**
    - Update Customer – **update_Click**

15. Save your changes.

Creating the Customer Details Form

Create the user interface for the Customer_Details form.

1. Add another Windows Form to the project, and name this one `Customer_Details`.
2. From the Toolbox, drag and drop three labels onto the Customer_Details form.
3. Align the labels on the left side of the form.

   In the Properties view, in the Text field, rename the labels **Cust ID**, **First Name**, and **Last Name**.
4. From the Toolbox, drag and drop three text boxes onto the Customer_Details form and align them to the right of each of the three labels.
5. From the Toolbox, drag and drop one button from Common Device Controls to the Customer_Details form below the labels and text boxes.
6. In the Properties view, in the **Text** field, rename the button to **Submit**.
7. Save the `Customer_Details.cs` form.
8. In Solution Explorer, in the SUP101 project, right-click `Customer_Details.cs`, then select **View Code**.
9. Replace the existing code with the code from the `Customer_Details.cs` file you downloaded from the Sybase Product Documentation Web site:

```
using System;

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;

using SUP101;
using Sybase.Collections;
using Sybase.Persistence;

namespace SUP101
{
    public partial class Customer_Details : Form
    {
        Customer thisCustomer;
        public Customer_Details()
        {
            InitializeComponent();
        }

        private void submit_Click(object sender, EventArgs e)
        {
            thisCustomer.Fname = textBox2.Text;
            thisCustomer.Lname = textBox3.Text;
            thisCustomer.Save();
            thisCustomer.SubmitPending();

            ISynchronizationGroup sg = 
SUP101DB.GetSynchronizationGroup("default");
            GenericList<ISynchronizationGroup> syncGroups = new 
GenericList<ISynchronizationGroup>();
            syncGroups.Add(sg);
            SUP101DB.BeginSynchronize(syncGroups, "");

            Program.getForm1().Visible = true;

Program.getForm1().RefreshListItem(thisCustomer.SurrogateKey);
            Program.getForm2().Visible = false;
        }
        private void AddDataToForm()
        {
            textBox1.Text = Program.getCustomer();
            int id = Int32.Parse(Program.getCustomer());
            thisCustomer = Customer.FindByPrimaryKey(id);
            textBox2.Text = thisCustomer.Fname;
            textBox3.Text = thisCustomer.Lname;
        }
       private void Customer_Details_Load(object sender, EventArgs 
e)
        {
            AddDataToForm();
        }
        private void Customer_Details_paint(object sender, 
PaintEventArgs e)
        {
            AddDataToForm();
        }
    }
}
```

10. In the Customer_Details.cs[Design] view, click the **Submit** button, then in the Properties view, click the **Event** icon (lightning bolt), add the submit_Click event to the Submit button.

11. Add events to the `Customer_Details.cs`.

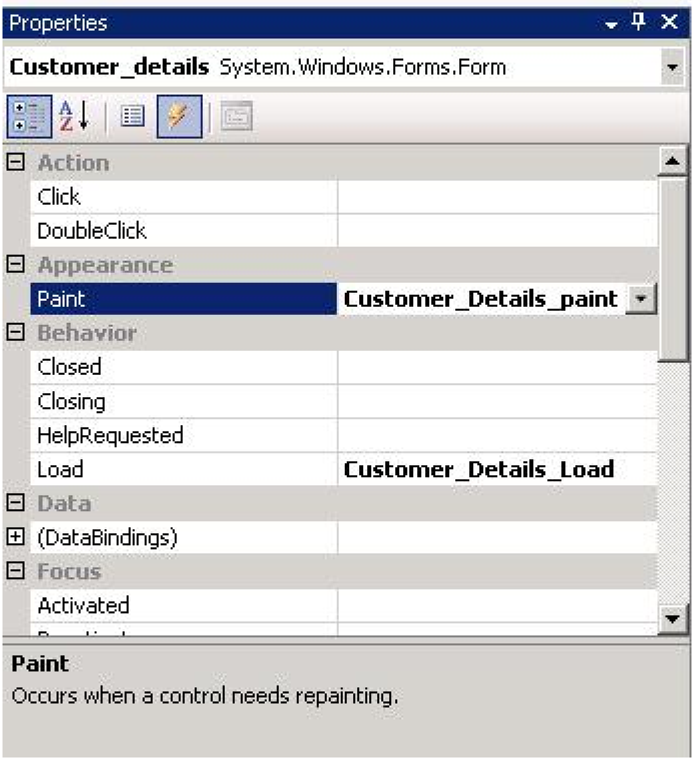
a) Click the Customer_Details.cs [Design] tab.

b) In Properties view, select **Customer_Details System.Windows.Forms.Form.** from the drop-down list at the top of the view.

c) Click the **Events** icon (lightning bolt).

d) In **Load**, add the **Customer_Details_Load** event.

e) In **Paint**, add the **Customer_Details_paint** event.



12. Save your changes.

Creating the Main Program File

Create the `Program.cs` file, which is the main entry point for the application.

1. In the Solution Explorer, right-click SUP101, then select **Add** > **New Item**.
2. In Categories, select **Code** and in Templates, select **Code File**.
3. Name the code file `Program.cs`, then click **Add**.

4. Right-click `Program.cs`, then select **View Code**.
5. Replace the existing code with the code from the `Program.cs` file you downloaded from the Sybase Product Documentation Web site:

```
using System;
using SUP101;
using System.Windows.Forms;

namespace SUP101
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>private static Customers _form1 = new 
Form1();
        private static Customers _form1 = new Customers();
        private static Customer_Details _form2 = new 
Customer_Details();
        private static string _custid;
        public static string getCustomer()
        {
            return _custid;
        }
        public static void setCustomer(string custid)
        {
            _custid = custid;
        }
        public static Customers getForm1()
        {
            return _form1;
        }
        public static Customer_Details getForm2()
        {
            return _form2;
        }
        static void Main(string[] args)
        {
            Application.Run(_form1);
        }
    }
}
```

6. Save your changes.
7. Build the project by pressing **Control+Shift+B**.

Deploying and Running the Device Application

Deploy the device application to a Windows Mobile 6 device emulator, and test its functionality.

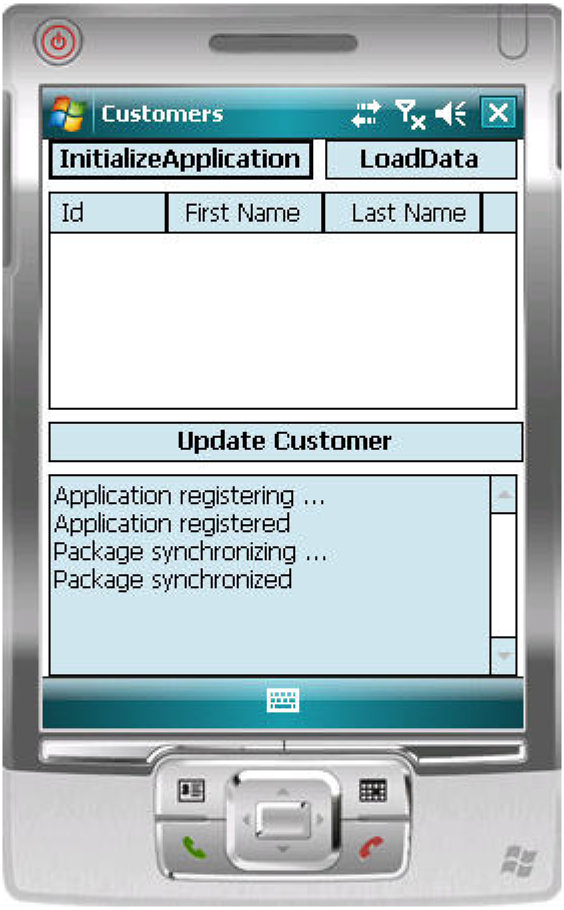
1. In Visual Studio, choose **Tools** > **Device Emulator Manager**.

2. Scroll down to **Windows Mobile 6 Professional** right-click, and select **Connect**.
3. Right-click **Windows Mobile 6 Professional** again and select **Cradle**.
4. In the Solution Explorer of Visual Studio, right-click the C# project and select **Deploy**.
5. Then in the Windows Mobile emulator go to **Start** > **Programs** > **File Explorer**.
6. Navigate to the installed path by accessing **Program Files** > **sup101**.
7. Click **SUP101** to start the application.
8. Click **InitializeApplication**.

   Inside InitializeApplication, the application is registering and synchronizing data from the back end server. The initialization process also enables Target Change Notification by:

   ```
   ISynchronizationGroup sg =
   SUP101DB.GetSynchronizationGroup("default");
   sg.EnableSIS = true;
   sg.Save();
   ```

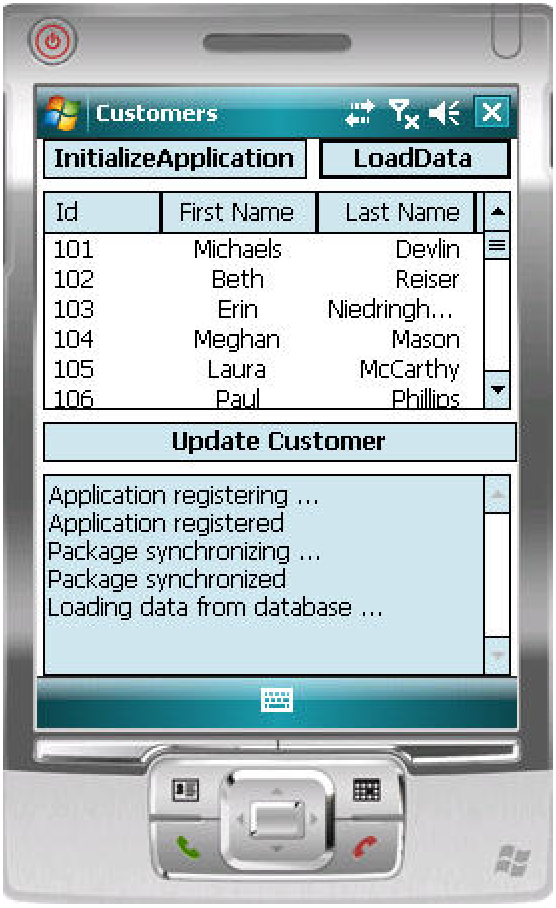
   A message displays in the output box below when complete.



```
Application registering ...
Application registered
Package synchronizing ...
Package synchronized
```

9. Click **LoadData** to populate Customer data in the list view.

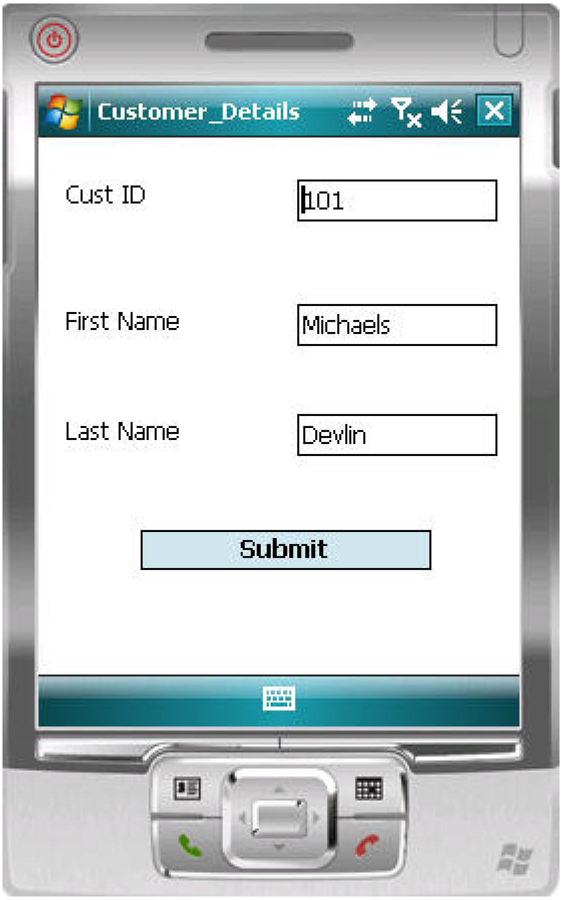
   To improve performance, SUP101DB.ExecuteQuery is used to query customer list, selecting only required columns (fname, lname...) instead of the entire customer object.



Id	First Name	Last Name
101	Michaels	Devlin
102	Beth	Reiser
103	Erin	Niedringh...
104	Meghan	Mason
105	Laura	McCarthy
106	Paul	Phillips

```
Application registering ...
Application registered
Package synchronizing ...
Package synchronized
Loading data from database ...
```

10. Highlight a customer record and click **Update Customer**.



Customer_Details

Cust ID: 101

First Name: Michaels

Last Name: Devlin

11. Make changes, then click **Submit** to return to the Customers List screen.

Inside Submit, customer info is updated and SUP101DB.beginSynchronize is called in the background to avoid blocking the user interface.

See also

- *Creating the User Interface for the Windows Mobile Device Application* on page 14

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: *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.

Example Projects

Example projects are available for download, if you want the finished tutorial without going through the steps. Download example 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 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 how to develop mobile business objects (MBOs) to fully maximize their potential.

Learn about using the 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: Mobile Workflow Packages*

Customize and automate:

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

Javadoc and HeaderDoc are also available in the installation directory.

Index