Tutorial: Windows Object API Application Development

SAP Mobile Platform 2.3

DOCUMENT ID: DC01937-01-0230-01
LAST REVISED: March 2013

Contents

SAP Mobile Platform Tutorials

The SAP® tutorials demonstrate how to develop, deploy, and test mobile business objects, device applications, 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 SAP Mobile 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*

Getting Started with SAP Mobile Platform

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. *Installing SAP Mobile Platform*

   Install SAP Mobile SDK and SAP Mobile Platform Runtime.
2. *Starting SAP Mobile Platform Services*

   Start SAP Mobile Server, SAP Control Center, the sample database, the cache database (CDB), and other essential services.
3. *Starting SAP Mobile WorkSpace*

   Start the development environment, where you can create mobile business objects (MBOs), create connection profiles and manage SAP Mobile Server connections, develop Hybrid Apps, and generate Object API code.
4. *Connecting to SAP Control Center*

   Open SAP Control Center to manage SAP Mobile Server and its components.
5. *Learning SAP Mobile WorkSpace Basics*

   SAP Mobile WorkSpace features are well integrated in the Eclipse IDE. If you are unfamiliar with Eclipse, you can quickly learn the basic layout of SAP Mobile WorkSpace and the location of online help.

Installing SAP Mobile Platform

Install SAP Mobile SDK and SAP Mobile Platform Runtime.

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

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

1. Install these SAP Mobile Platform Runtime components:
   - Data Tier (included with single-server installation)
   - SAP Mobile Server
2. Install SAP Mobile SDK, which includes:

- Development support for native Object API and OData SDK applications, as well as HTML5/JS Hybrid Apps.
- SAP Mobile WorkSpace, the Eclipse-based development environment for MBOs and Hybrid Apps.

Starting SAP Mobile Platform Services

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

The way in which you start SAP Mobile Platform Services depends on the options you selected during installation. You may need to manually start SAP Mobile Platform Services.
Select **Start > (All) Programs > SAP > Mobile Platform > Start SAP Mobile Platform Services**.
The SAP Mobile Platform Services enable you to access the SAP Mobile Platform runtime components and resources.

Starting SAP Mobile WorkSpace

Start the development environment, where you can create mobile business objects (MBOs), create connection profiles and manage SAP Mobile Server connections, develop Hybrid Apps, and generate Object API code.
Select **Start > (All) Programs > SAP > Mobile Platform > Mobile WorkSpace 2.3**.
The SAP Mobile WorkSpace opens in the Mobile Development perspective. The Welcome page displays links to the product and information.

Next

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

Connecting to SAP Control Center

Open SAP Control Center to manage SAP Mobile Server and its components.

From SAP 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 SAP Mobile Server, click **Help > Help Contents**.

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

   Note: If SAP Control Center does not launch, make sure that the SAP Control Center service is started in the Windows Services dialog.
2. Log in by entering the credentials set during installation.

   SAP Control Center gives you access to the SAP Mobile Platform administration features that you are authorized to use.

Learning SAP Mobile WorkSpace Basics

SAP Mobile WorkSpace features are well integrated in the Eclipse IDE. If you are unfamiliar with Eclipse, you can quickly learn the basic layout of SAP Mobile WorkSpace and the location of online help.

- To access the online help, select **Help > Help Contents**. Some documents are for SAP Mobile WorkSpace, while others are for the Eclipse development environment.
- The Welcome page provides links to useful information to get you started.
  - To close the Welcome page, click **X** in the upper right corner of the page.
  - Reopen the Welcome page by selecting **Help > Welcome**.
  - To learn about tasks you must perform, select the **Development Process** icon.
- In SAP Mobile 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 SAP Mobile 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 datasource, when creating an MBO. For example, you may want to model your MBOs before creating the datasources to which they bind. This MBO development method is sometimes referred to as the top-down approach. • Drag and drop items from Enterprise Explorer to 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.

Window	Description
Problems view	Displays validation errors or warnings that you may encounter in addition to errors in the Diagram editor and Properties view. Follow warning and error messages to adjust MBO properties and configurations to avoid problems, and use as a valuable source for collecting troubleshooting information when reporting issues to Customer Service and Support.
Error Log view	Displays error log information. This is a valuable source for collecting troubleshooting information.

Developing a Windows Application

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

Prerequisites

Note: This tutorial was created using SAP Mobile Platform 2.3, Microsoft .NET Framework 3.5, and Visual Studio 2008, and executed on Windows 7 Professional SP 1. If you use different versions, some steps may vary.

- Complete the tasks in *Getting Started with Mobile Platform*.
- Install Visual Studio 2008.
- Install Microsoft .NET Framework 3.5 from *http://www.microsoft.com/en-us/download/details.aspx?id=65*, if it was not installed with Visual Studio.
- Either:
  - create the MBO project by completing *Tutorial: Mobile Business Object Development*, or
  - download and deploy the MBO SMP101 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 SAP Mobile WorkSpace and then accepting the confirmation prompt.
- (Optional) To use as a reference and copy source code when completing this tutorial, download the 2.2 SP02 version of the Windows SMP 101 example project (source code only) from the SAP ® Community Network: *http://scn.sap.com/docs/DOC-8803*.

Task

Create a Windows native application that communicates with the mobile business objects that are deployed to SAP Mobile Server.

Generating C# Object API Code

Generate Object API code for Windows Mobile devices.

Prerequisites

1. Connect to both the sampledb database and SAP Mobile 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 SAP Mobile WorkSpace.

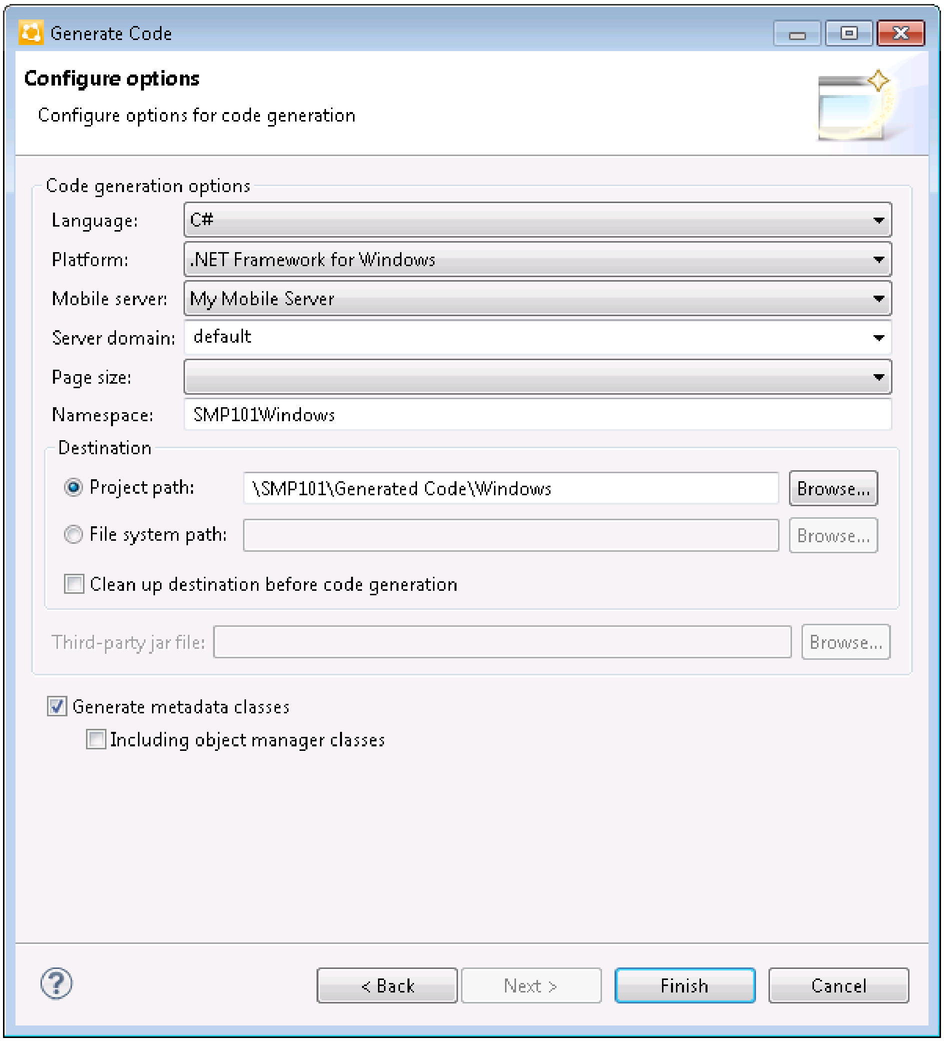
Task

1. Start SAP Mobile WorkSpace, using the same workspace location you used when you created the SMP101 project in the MBO tutorial.
2. In SAP Mobile WorkSpace, open the **SMP101** mobile application project.

   In WorkSpace Navigator, right-click the **SMP101** folder and select **Open in Diagram Editor**.
3. In WorkSpace Navigator, expand SMP101. Under Generated Code, add a folder named `Windows`.
4. Right-click anywhere in the SMP101 - Mobile Application Diagram and select **Generate Code**.
5. On the Code generation configuration screen, click **Next**.
6. In the Select Mobile Business Objects winow, select the Customer MBO, then click **Next**.

   Ignore any warning about unresolved mobile business object dependencies.
7. Enter the information for these configuration options:

Option	Description
Language	Select **C#**.
Platform	Select **.NET Framework for Windows**.
Mobile Server	Select **My Mobile Server**.
Server domain	Select **default**.
Page size	Select <blank> (replacing 1024).
Namespace	Enter `SMP101Windows`.
Project path	Enter `\SMP101\Generated Code \Windows`.



Code generation options

Language:	C#
Platform:	.NET Framework for Windows
Mobile server:	My Mobile Server
Server domain:	default
Page size:	
Namespace:	SMP101Windows

Destination

(●) Project path: \SMP101\Generated Code\Windows

( ) File system path:

☐ Clean up destination before code generation

Third-party jar file:

☑ Generate metadata classes

☐ Including object manager classes

8. Click **Finish**.

If you see a success notification dialog, click **OK**.

In the Generated Code directory, you see `Windows\src\SMP101Windows`.

Creating the User Interface for the Windows Mobile Device Application

Import the SMP101 project in to Visual Studio 2008, configure the project, then download the tutorial code snippets from SAP Community Network (SCN) so you can create the application user interface.

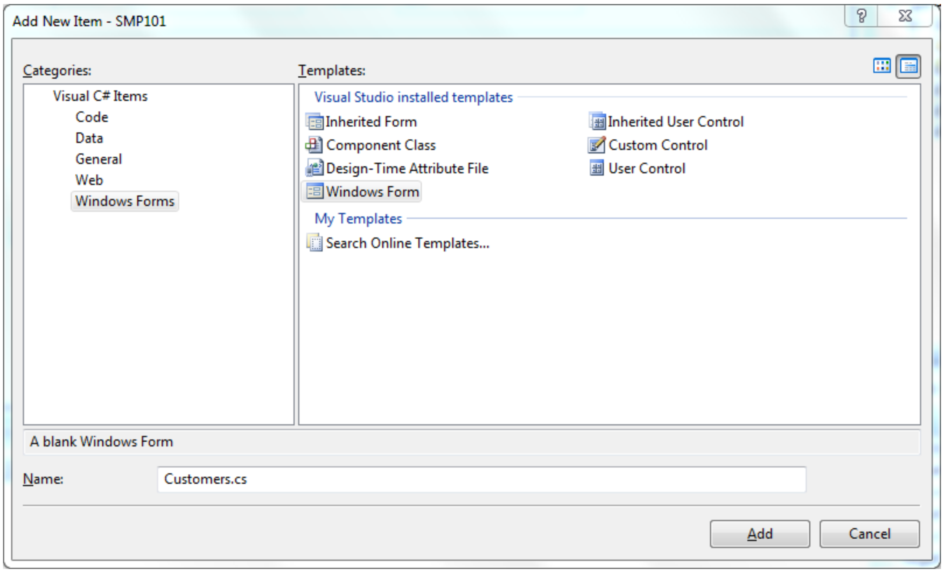
The C# Object API code you generated from the MBO project is located in your Eclipse workspace under `..\SMP101\Generated Code\Windows`.

1. From WorkSpace Navigator, double-click on `\SMP101\Generated Code\Windows\src\SMP101.csproj` to open the project in Visual Studio 2008.
2. Edit the project properties:
   a) In Solution Explorer, under 'SMP101' (1 Project), right-click **SMP101**, then select **Properties**.
   b) In the Application tab, set "Output type" to **Console Application**.
   c) Select **File** > **Save All**, then save the solution as `SMP101.sln`.

Creating the Customers Form

Create the user interface for the Customers form.

1. In Solution Explorer, under Solution 'SMP101' (1 project), right-click the **SMP101** 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.cs` as the form name and click **Add**.

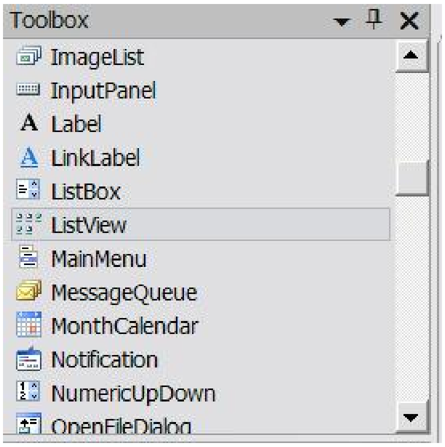


A blank Windows Form

Name: Customers.cs

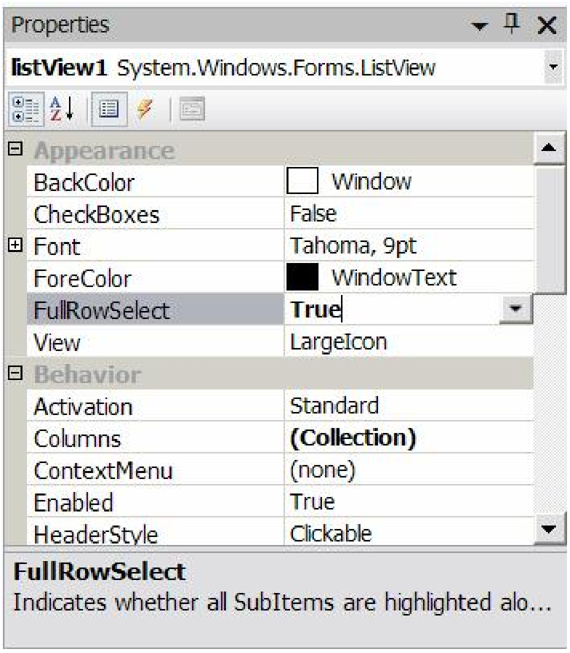
An empty form, **Customers**, appears on the Customers.cs [Design] tab.

3. From the Toolbox, drag and drop three buttons onto the form.
4. Select each button and, in the Properties view, change the **Text** of the buttons to:
   - button1 – `Initialize Application`
   - button2 – `Load Data`
   - button3 – `Update Customer`
5. In the Toolbox, under Common Controls, drag **ListView** and drop it onto the Customers form.



6. In the Toolbox, under Common Controls, drag **Textbox** and drop it onto the Customers form.

7. In the Customers form, click the **ListView** control, then in the Properties pane, set **FullRowSelect** to **True**.

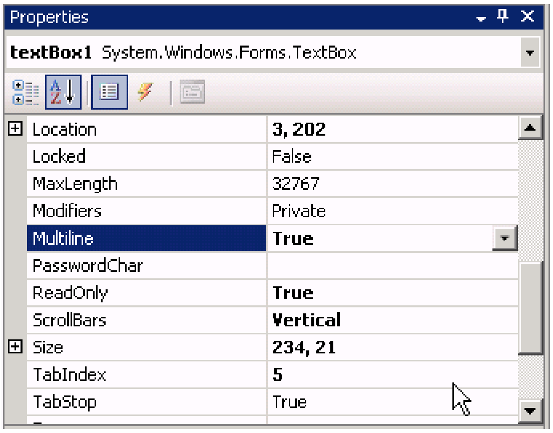


Appearance	
BackColor	Window
CheckBoxes	False
Font	Tahoma, 9pt
ForeColor	WindowText
FullRowSelect	True
View	LargeIcon
Behavior	
Activation	Standard
Columns	(Collection)
ContextMenu	(none)
Enabled	True
HeaderStyle	Clickable

FullRowSelect
Indicates whether all SubItems are highlighted alo...

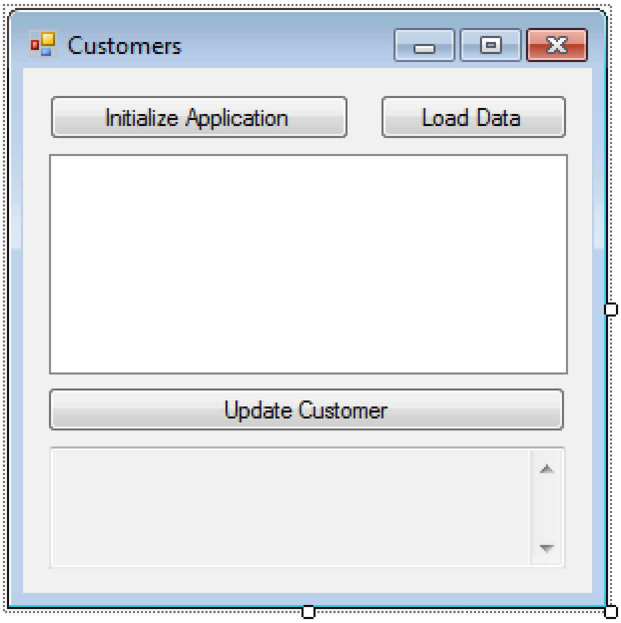
8. In the Customers form, select the **Textbox**, then in the Properties pane, set these properties:

- Multiline – True
- ReadOnly – 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 SMP101 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 SAP Community Network (SCN) Web site, also provided below:

    Edit the bolded code lines to match the SAP Mobile Platform Admin login, password, and host you indicated during installation.

```
using System;
using System.Linq;

using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using SMP101Windows;
using Sybase.Mobile;
using Sybase.Persistence;
using Sybase.Collections;

namespace SMP101Windows
{
    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 = "SMP101";
            MyCallbackHandler.textBox1 = textBox1;
            SMP101DB.RegisterCallbackHandler(new
MyCallbackHandler());
            SMP101DB.SetApplication(app);
            SMP101DB.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 (!SMP101DB.IsSynchronized("default"))
            {
                SMP101DB.DisableChangeLog();
                AddString("Package synchronizing ...");
                SMP101DB.Synchronize(); // Initial Synchronize

                ISynchronizationGroup sg =
SMP101DB.GetSynchronizationGroup("default");
                sg.EnableSIS = true;
                sg.Save();
                SMP101DB.Synchronize();
                AddString("Package synchronized");
            }
            SMP101DB.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",
Sybase.Persistence.SortOrder.ASCENDING);

            int index = 0;
            QueryResultSet rs = SMP101DB.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 = 
SMP101DB.GetChangeLogs(query);
                    foreach (IChangeLog changeLog in changeLogs)
                    {
                        if (changeLog.EntityType == 
EntityType.Customer)
                        {
                          invokeDelegate(changeLog.SurrogateKey);
                        }
                    }
                }

                return SynchronizationAction.CONTINUE;
            }
        }
    }
}
```

13. Click the **Customers.cs[Design]** tab to return to the Customers form design view to add event handlers to the buttons.

    a) Click the **Initialize Application** button on the form.

b) In the Properties view for the button, click the **Event** icon (lightning bolt).
c) Next to Click, select **InitializeApplication_Click**.
d) Repeat these steps for each button, selecting these events for each Click data binding:
   - LoadData – **LoadData_Click**
   - Update Customer – **update_Click**

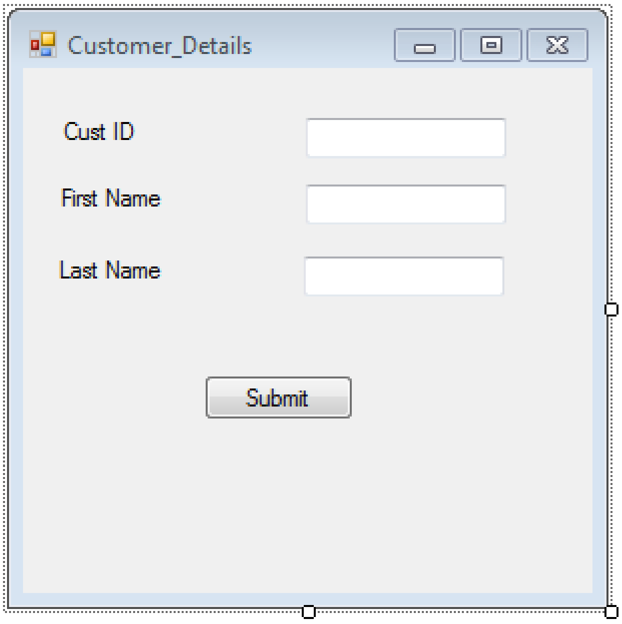
14. Save your changes.

Creating the Customer Details Form

Create the user interface for the Customer_Details form.

1. Add another Windows Form, named `Customer_Details.cs`, to the project.
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 Controls onto the Customer_Details form below the labels and text boxes.
6. In the Properties view, in the Text field, rename the button `Submit`.

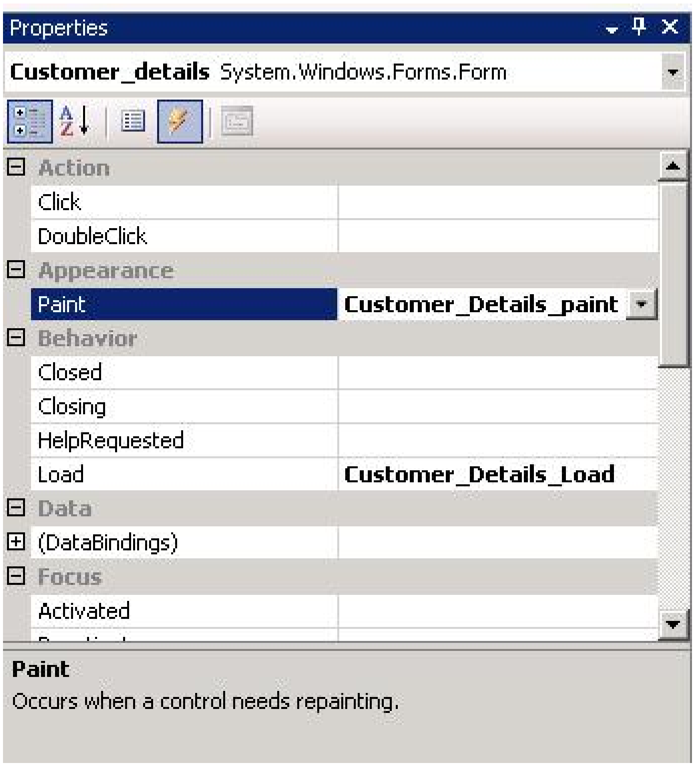


7. Save the `Customer_Details.cs` form.

8. In Solution Explorer, in the SMP101 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 SAP Community Network (SCN) 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 SMP101Windows;
using Sybase.Collections;
using Sybase.Persistence;

namespace SMP101Windows
{
    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 =
SMP101DB.GetSynchronizationGroup("default");
            GenericList<ISynchronizationGroup> syncGroups = new
GenericList<ISynchronizationGroup>();
            syncGroups.Add(sg);
            SMP101DB.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). Next to Click, select the submit_Click event from the drop-down list.
11. Add events to `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.



Paint	Customer_Details_paint
Load	Customer_Details_Load

Paint
Occurs when a control needs repainting.

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 **SMP101**, 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**.

   An empty code tab titled Program.cs opens.
4. Copy the code from the `Program.cs` file you downloaded from the SAP Community Network (SCN) Web site, also provided below, into the Program.cs tab.

```
using System;
using SMP101Windows;
using System.Windows.Forms;

namespace SMP101Windows
{
    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);
        }
    }
}
```

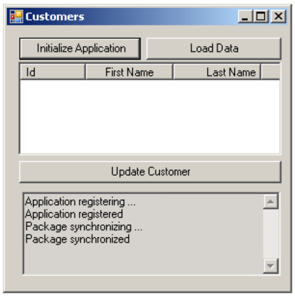
5. Save your changes.
6. Build the project by pressing **Ctrl+Shift+B**.

Deploying and Running the Device Application

Deploy the application and test its functionality.

1. In Visual Studio, run the application by pressing **Ctrl+F5**.
2. Click **Initialize Application**.

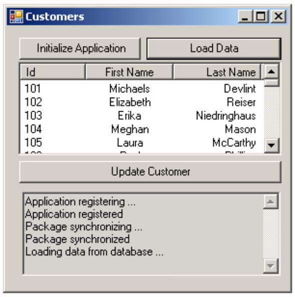
   Inside InitializeApplication, the application is registering and synchronizing data from the backend server. A message displays in the output box below when complete.



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

3. Click **Load Data** to populate Customer data in the list view.

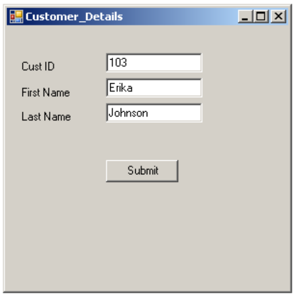
To improve performance, SMP101DB.ExecuteQuery queries the customer list, selecting only required columns (fname, lname...) instead of the entire customer object.



Id	First Name	Last Name
101	Michaels	Devlint
102	Elizabeth	Reiser
103	Erika	Niedringhaus
104	Meghan	Mason
105	Laura	McCarthy

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

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



Cust ID	103
First Name	Erika
Last Name	Johnson

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

Inside Submit, customer information is updated and SMP101DB.beginSynchronize is called in the background to avoid blocking the user interface.

Learn More About SAP Mobile Platform

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

Check the Product Documentation Web site regularly for updates: *http://sybooks.sybase.com/sybooks/sybooks.xhtml?id=1289&amp;c=firsttab&amp;a=0&amp;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 SAP Mobile 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 SAP Mobile 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*

Customize and automate:

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

Javadoc and HeaderDoc are also available in the installation directory.

Index