



**Tutorial: Android OData Application  
Development with REST Services**

---

**SAP Mobile Platform 2.3 SP02**

DOCUMENT ID: DC01975-01-0232-02

LAST REVISED: February 2014

Copyright © 2014 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase trademarks can be viewed at the Sybase trademarks page at <http://www.sybase.com/detail?id=1011207>. Sybase and the marks listed are trademarks of Sybase, Inc. ® indicates registration in the United States of America.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

Java and all Java-based marks are trademarks or registered trademarks of Oracle and/or its affiliates in the U.S. and other countries.

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

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

# Contents

<b>SAP Mobile Platform Tutorials .....</b>	<b>1</b>
<b>Getting Started with SAP Mobile Platform (On-Premise)</b>	<b>3</b>
.....	3
Installing SAP Mobile Platform .....	3
Starting SAP Mobile Platform Services .....	4
Connecting to SAP Control Center .....	4
Creating a Security Configuration for a Domain .....	5
Creating an Application ID and Whitelisting the Application Endpoint .....	7
<b>Getting Started with SAP Mobile Platform, enterprise     edition, cloud version Cloud .....</b>	<b>9</b>
Connecting to SAP Mobile Platform, enterprise edition, cloud version - Administration and Monitoring .....	9
Creating an Application .....	12
Creating Application Endpoint URL .....	13
Creating a Security Profile .....	14
<b>Develop the Android OData Application .....</b>	<b>15</b>
Installing the Android Development Environment .....	15
Installing the Android SDK .....	16
Creating the Android Project .....	16
Adding Libraries to the Android Project .....	20
Adding User Permissions in Android Manifest File .....	22
Creating the User Interface .....	22
Defining the Application Logic using REST SDK .....	26
Registering a User .....	26
Sending Data Request to the Backend .....	27
Retrieving the Response from Backend .....	28
<b>Running your Android OData Application .....</b>	<b>31</b>
<b>Learn More About SAP Mobile Platform .....</b>	<b>37</b>

**Index** .....**39**

# SAP Mobile Platform Tutorials

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

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

- *Tutorial: Mobile Business Object Development*

---

**Note:** For all Object API tutorials, if you opt to use the Mobile Business Object example project instead of performing the Mobile Business Object Tutorial, you must deploy the mobile application project to 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*
- Create an OData mobile application with REST Services
  - *Tutorial: Android OData Application Development with REST Services*
  - *Tutorial: iOS OData Application Development with REST Services*



# Getting Started with SAP Mobile Platform (On-Premise)

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***

(Applicable to On-Premise version) 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. *Connecting to SAP Control Center***

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

**4. *Creating a Security Configuration for a Domain***

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

**5. *Creating an Application ID and Whitelisting the Application Endpoint***

Create a new application using SAP Control Center.

## Installing SAP Mobile Platform

---

(Applicable to On-Premise version) 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.**

## 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 following services will be started:

- SAP Control Center <Version>
- SAP Mobile Platform Cache DB
- SAP Mobile Platform SampleDB
- SAP Mobile Server

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

---

**Note:** The SAP Mobile Platform installer creates the Windows service (SAP Mobile Platform Sample DB) that runs the sampledbs server only when you install SAP Mobile Server with a Personal or Enterprise Development license. If you installed SAP Mobile Server with an Enterprise Server (production) license, you must create this service using the `sampledb.bat` command line utility. See *Create or Remove the Windows Service for sampledbs Server (sampledb) Utility* in *System Administration* for more information about using this command line utility.

---

## 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.

## Creating a Security Configuration for a Domain


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

### Prerequisites

Connect to SAP Control Center.

### Task

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



5. Click **OK**.  
SSO is created as desired security configuration in left navigation pane under **Security** node.
6. In SAP Control Center, select **View > Select > Mobile Server Cluster Management View**.
7. In the left navigation pane, select **Domains -> default -> Security** folder and click **Assign**.
8. Select **SSO** and click **OK**. In the left navigation pane, select **Security > SSO**. In the right pane, under **Authentication** click **New...**
9. In **Add Provider** dialog box:

## Getting Started with SAP Mobile Platform (On-Premise)

- a) Select the required loginModule in authentication provider from the drop-down list.
- b) Enter the authentication provider URL in **URL** field as `http://vmw3815.wdf.sap.corp:50009/sap/opu/odata/iwfind/RMTSAMPLEFLIGHT/`
- c) Click **Save**.

<input type="checkbox"/> Property	Value	<input type="checkbox"/>
<input type="checkbox"/> Implementation Class	com.sybase.security.http.HttpAuthenticationLog	<input type="checkbox"/> Delete
<input type="checkbox"/> Provider Type	LoginModule	
<input type="checkbox"/> Control Flag	optional	
<input type="checkbox"/> URL	http://vmw3815.wdf.sap.corp:50009/sap/opu/s	
<input type="checkbox"/> SSO Cookie Name	MYSAPSSO2	
<input type="checkbox"/> <ADD NEW PROPERTY>		

OK Cancel

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

Similarly, under **Authorization** and **Attribution** tabs, delete the default provider types: **NoSecAuthorizer** and **NoSecAtributer** respectively.

**11.** Under **General** tab, click **Validate** to validate the configuration before applying the changes to the SAP Mobile Server.

**12.** Click **Apply**.

### Next

In SAP Control Center, create the application ID.

## Creating an Application ID and Whitelisting the Application Endpoint

Create a new application using SAP Control Center.

1. In the left navigation pane of SAP Control Center, click the **Applications** node and select the **Applications** tab in the right administration pane.
2. Click **New...**
3. In the **Application Creation** dialog box, enter the required information:
  - **Application ID** - smp.tutorial.android
  - **Display name** - android application
  - **Description** - Application ID for SMP sample flight management application
  - Select **Security configuration** - SSO
  - Select **Domain** - default
4. Enable **Configure additional settings** checkbox.

**Application Creation**

**Application general information**

Application ID:

Display name:

Description:

Security configuration:   Anonymous access

Domain:

**MBO Package**



Configure additional settings

<Back   Next>   Finish   Cancel

5. Click **Next**.
6. Under **Application connection template**, select **Proxy** from the list.

7. Enter the **Application Endpoint** as `http://vmw3815.wdf.sap.corp:50009/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/`

**Application Creation**

**Application connection template**

Template name:

Description:

Base template:

Property	Value
Application Endpoint	vmw3815.wdf.sap.corp:50009/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/
Push Endpoint	http://INLN50819858A:8000/notifica

<Back   Next>   Finish   Cancel

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

### Next

In eclipse, create the user interface and application logic. See, *Creating the User Interface* on page 22 and *Creating the Application Logic using REST SDK* on page 26.

# Getting Started with SAP Mobile Platform, enterprise edition, cloud version Cloud

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

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

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

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

**2. *Creating an Application***

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

**3. *Creating Application Endpoint URL***

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

**4. *Creating a Security Profile***

Create a new security profile and store the application settings.

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

---

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

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

---

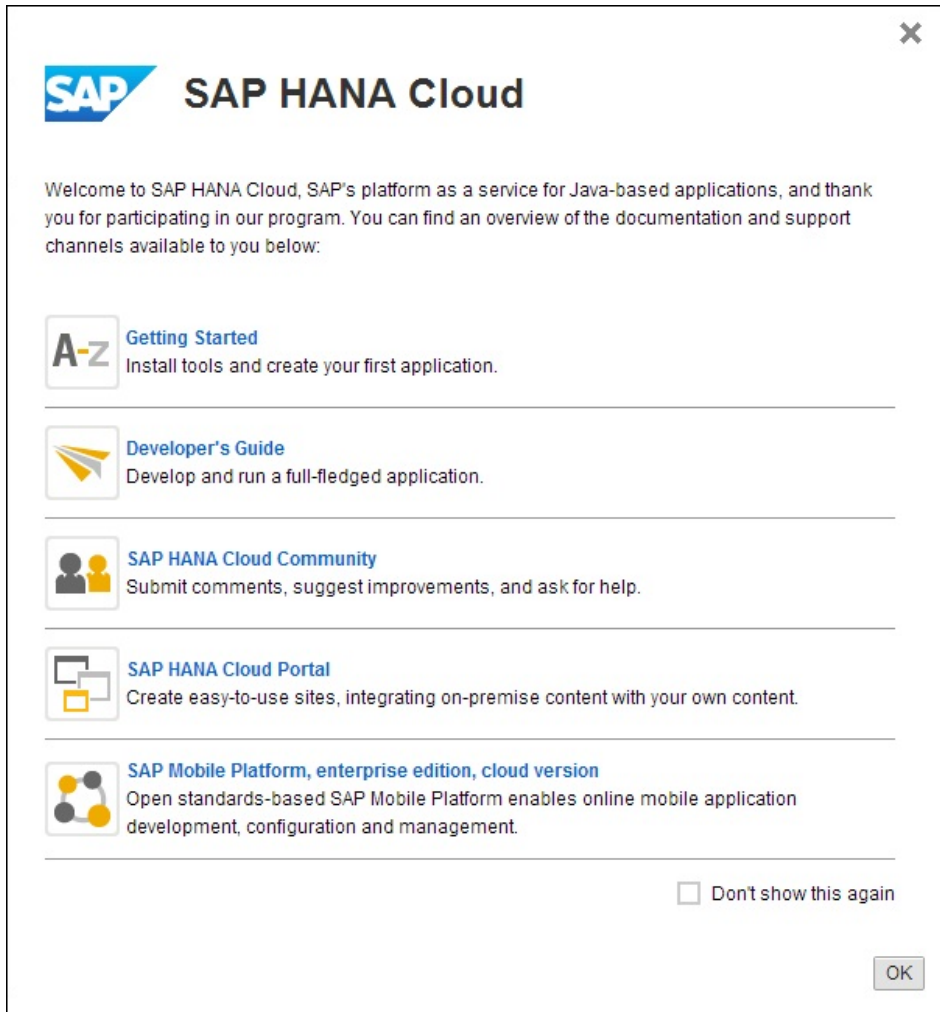
**Note:**

- If you are a user with an SAP HANA Cloud trial (developer) account, you are automatically subscribed and authorized to access **SAP Mobile Platform, enterprise edition, cloud version** and you can skip Step 2. Proceed with *Getting Started*.
  - If you are a user with a SAP HANA Cloud productive account, you must manually subscribe to **SAP Mobile Platform, enterprise edition, cloud version** and get authorization access using Step 2. After getting authorization, proceed with *Getting Started*.
- 

**2. Get administration, authentication, and authorization for SAP Mobile Platform, enterprise edition, cloud version, Administration and Monitoring portal.**

- a.** Go to the SAP HANA Cloud Account page: <https://account.hana.ondemand.com>.

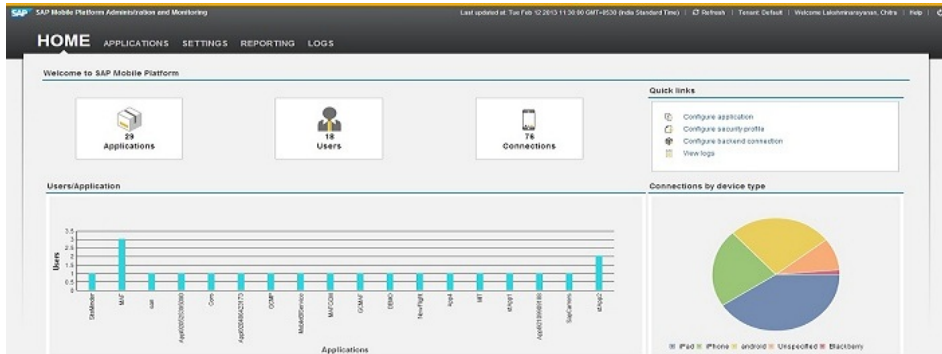
- b.** To assign users to roles, select the **AUTHORIZATIONS** tab.
  - c.** On the **Roles** subtab, enter user ID.
  - d.** Select the **Application** and **Role** from drop-down lists and click **Show**.
  - e.** Under **Assigned role <Role> for users:**, select **Users** or **Groups** from drop-down.
  - f.** Click **Assign**.
  - g.** In **Assign role <Role> for user** dialog, enter the **User ID** and click **Assign**.  
For more information on various roles and associated tasks, see *Platform Administration Roles and Tasks*.
- 3.** Open your SAP HANA Cloud trial (developer) account page using <https://account.hanatrial.ondemand.com/> and SAP HANA Cloud productive account using <https://account.hana.ondemand.com/>.  
The SAP HANA Cloud account welcome page is displayed with a link to the **SAP Mobile Platform, enterprise edition, cloud version**.



**4. Click SAP Mobile Platform, enterprise edition, cloud version.**

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

**Figure 1: SAP Mobile Platform Administration and Monitoring Portal**



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

## Next

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

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

## Creating an Application

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

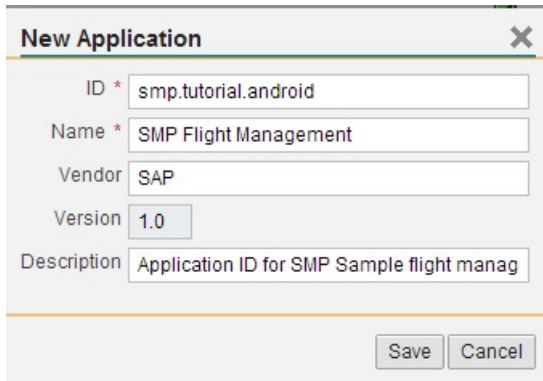
### Prerequisites

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

### Task

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





3. Click **Save**.

### Next

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

## Creating Application Endpoint URL

---

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

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

---

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

---



### Next

Navigate to **AUTHENTICATION** tab.

## Creating a Security Profile

---

Create a new security profile and store the application settings.

**1. Under AUTHENTICATION tab, select New Profile:**

- Enter **Security Configuration Name** - SSO
- Retain the default values under **General Settings**.
- Select **Authentication Type** - Basic Authentication.
- Enter **Authentication URL** - `http://vmw3815.wdf.sap.corp:50009/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/`

**2. Click Save.**

The **Confirm action** dialog box opens.

**3. Click OK.**

The application status turns to green and cloud destinations are created.

### Next

Create the user interface and application logic.

# Develop the Android OData Application

Develop an Android application using the REST SDK APIs, and test its functionality. The device application communicates with the databases that are deployed to SAP Mobile Server.

## 1. *Installing the Android Development Environment*

Set the Android development environment.

## 2. *Creating the Android Project*

Create a new android project in eclipse workSpace. Add library resources to the project and set other application properties.

## 3. *Creating the User Interface*

Use eclipse to build the user interface for the SMPFlightManagement application.

## 4. *Defining the Application Logic using REST SDK*

Create the application logic using REST SDK.

## Installing the Android Development Environment

---

Set the Android development environment.

### Prerequisites

Before starting ensure the following requirements are met:

- You should have basic knowledge of Open Data Protocol (OData). For more information, see *Open Data Protocol*.
- OData service document (Sample Flight) is available at <http://vmw3815.wdf.sap.corp:50009/sap/opu/odata/iwfnd/RMTSAMPLEFLIGHT/>.
- SAP Mobile Server is available with application referring to OData service document.
  - On-Premise - Create application ID and refer to OData service document, see *Creating an Application ID and Whitelisting the Application Endpoint* on page 7.
  - Cloud - Create application endpoint URL and refer to OData service document, see *Creating Application Endpoint URL* on page 13.

---

**Note:** This tutorial is created using SAP Mobile Platform 2.2 SP03, Android SDK r21.0.1, ADT Plugin for Eclipse 21.0.0, and run on an Android 4.1 - API Level 16 target emulator. If you use a different version, some steps may vary.

---

### See also

- *Creating the Android Project* on page 16

## Installing the Android SDK

Install the Android SDK.

1. Confirm that your system meets the requirements at <http://developer.android.com/sdk/requirements.html>.
2. Download and install the supported version of the Android SDK starter package.  
  
See the *Google Android Versions* topic for your platform in *Supported Hardware and Software* at <http://sybooks.sybase.com/sybooks/sybooks.xhtml?id=1289&c=firsttab&a=0&p=categories>. Select the appropriate version of the SAP Mobile Platform document set.
3. Launch the Android SDK Manager and install the Android tools (SDK Tools and SDK Platform-tools) and the Android API.
4. Launch the **Android Virtual Device Manager**, and create an Android virtual device to use as your emulator.

## Creating the Android Project

Create a new android project in eclipse workSpace. Add library resources to the project and set other application properties.

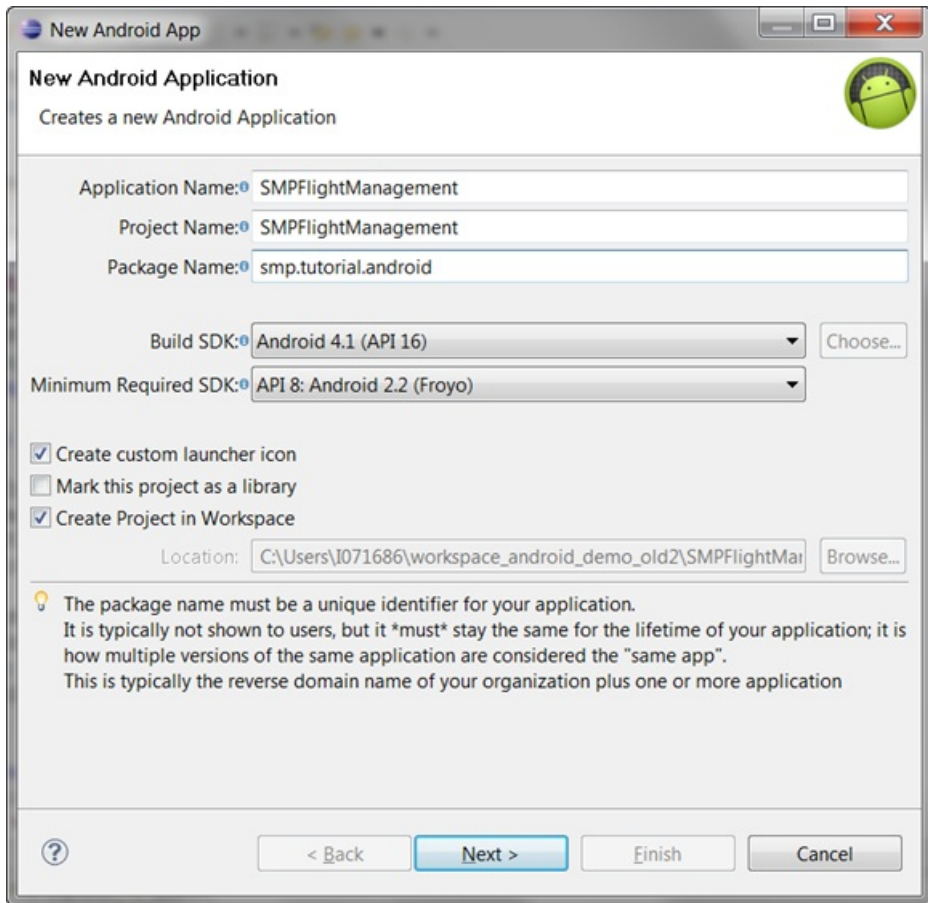
### **Prerequisites**

To help create your project—and in a subsequent topic, build the user interface—download the SMPFlightManagement Android OData Application (2.2 SP03) example project from the SAP Community Network (SCN) Web site at <http://scn.sap.com/docs/DOC-8803>.

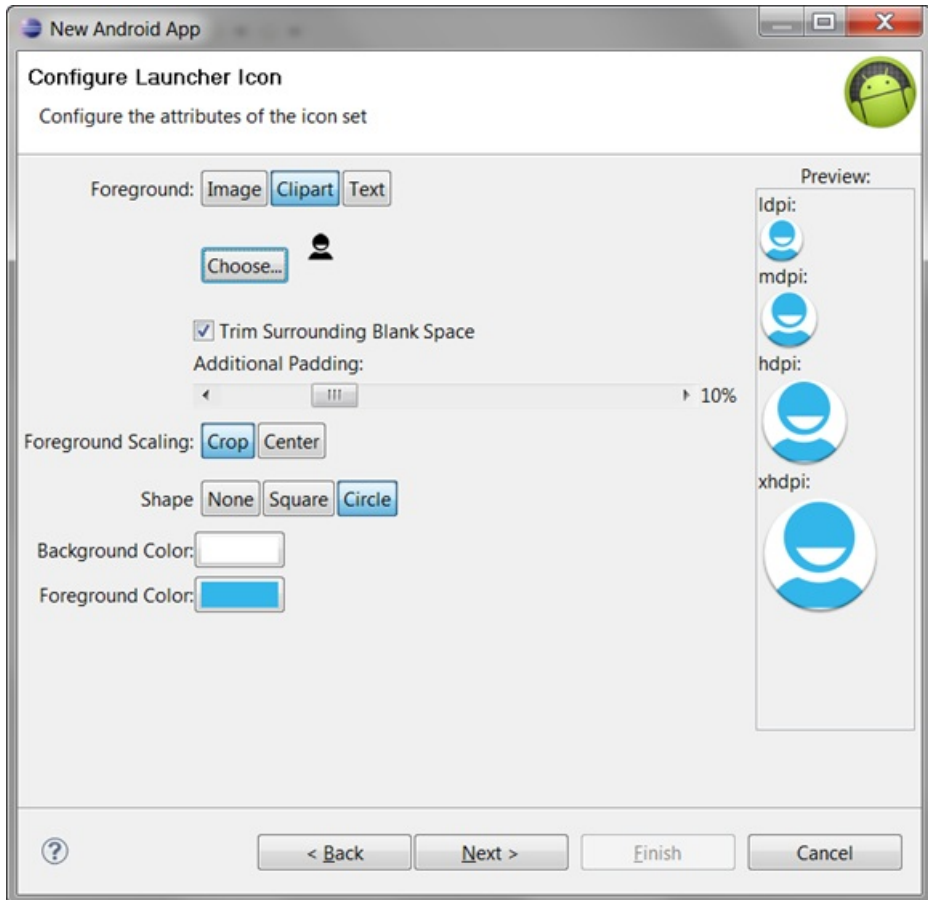
### **Task**

1. Start eclipse workspace.
2. Select **File > New > Project**.
3. Select **Android > Android Application Project**, then **Next**.  
If you are using an Android version other than the one used to design this tutorial, the screens you use to enter the information in the next several steps may be different.
4. In the Creates a new Android Application page of the New Android Application wizard, use these values: .
  - Enter SMPFlightManagement as **Application Name**.
  - By default, **Project Name** is populated with same application name..
  - Enter `smptutorial.android` as **Package Name**.
  - Choose Android 4.1 (API 16) from drop-down list for **Build SDK**

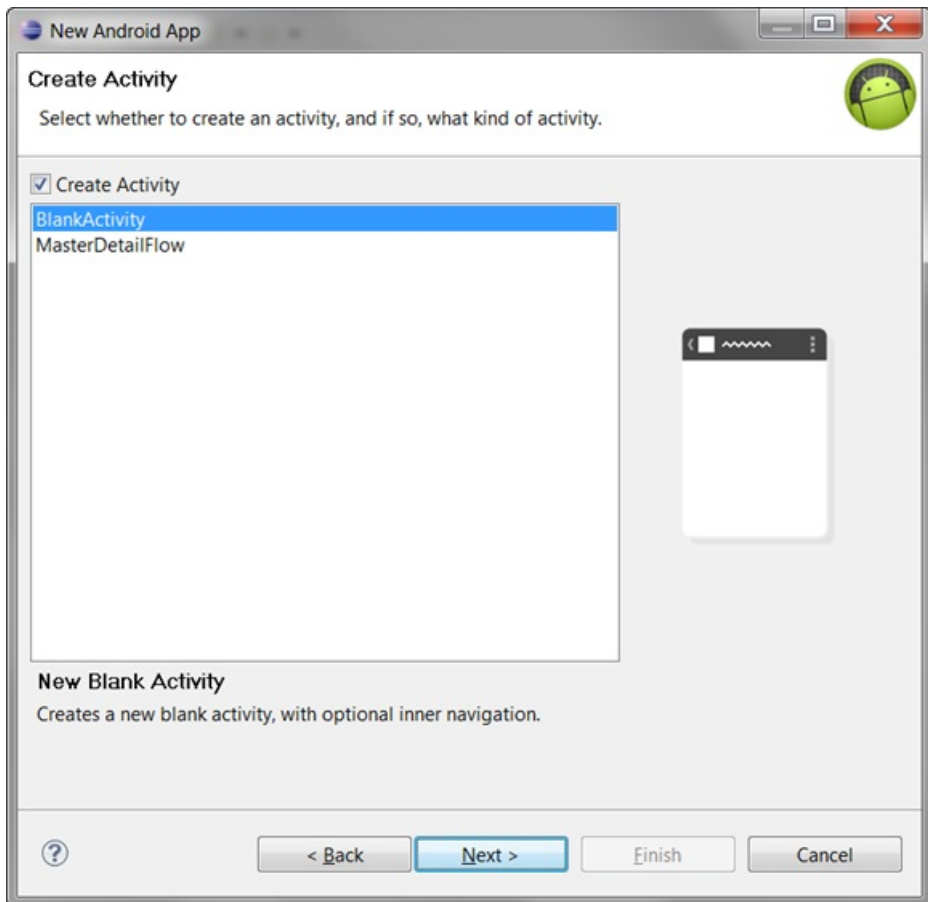
- Select the appropriate value for **Minimum Required SDK**
- Select **Create custom launcher icon** and **Create Project in Workspace**.
- Unselect **Mark this project as a library**.



5. Click **Next**.
6. In the **Configure Launcher Icon** window, accept the default settings and click **Next**.

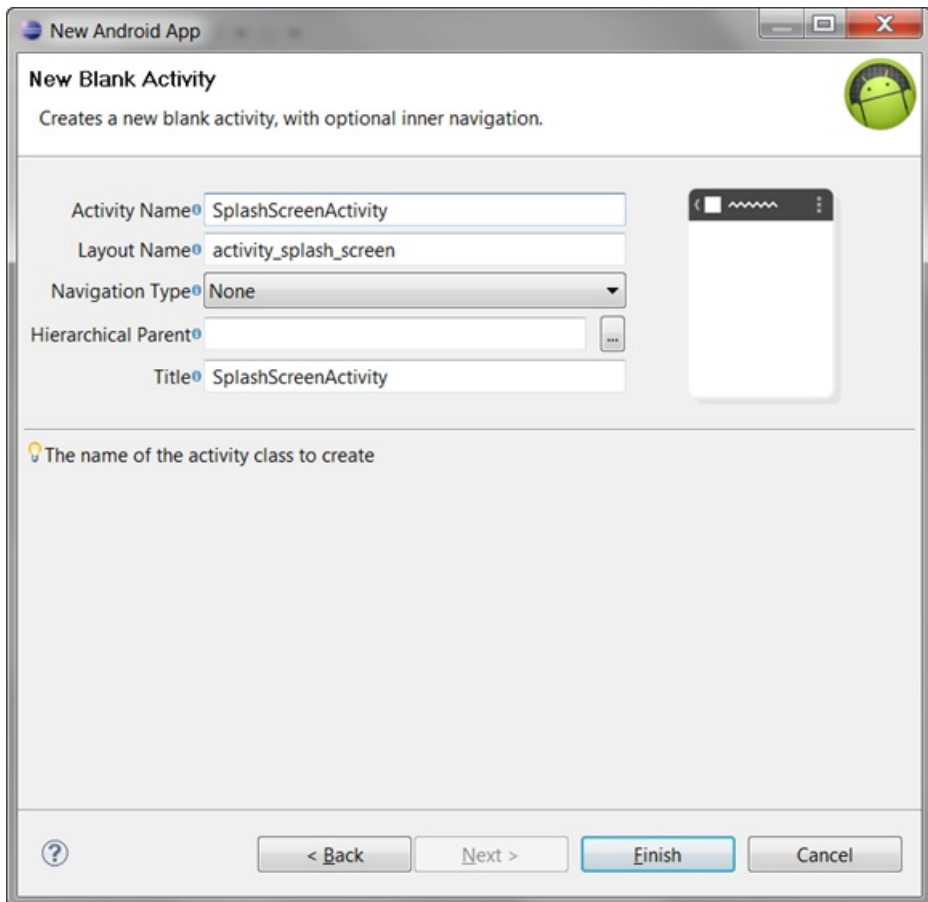


7. In the **Create Activity** window, select **Create Activity**, then select **BlankActivity**, and click **Next**.



8. In the **New Blank Activity** window, use these values and click **Finish**.

- Enter `SplashScreenActivity` as **Activity Name**.
- Enter `activity_splash_screen` as **Layout Name**.
- Select `None` from drop-down list as **Navigation Type**.



The left pane of the Workspace Navigator should list the SMPFlightManagement project. In the `src` folder, a default Sample Activity class was automatically generated when you created the project.

---

**Tip:** To correct a misspelled package name, right-click the package and select **Refactor > Rename** to change the name and update all references.

---

### See also

- *Installing the Android Development Environment* on page 15
- *Creating the User Interface* on page 22

## Adding Libraries to the Android Project

Add library resources to the Android project.

Download the following REST SDK library .jar files from <http://scn.sap.com/docs/DOC-8803> to your host development system:



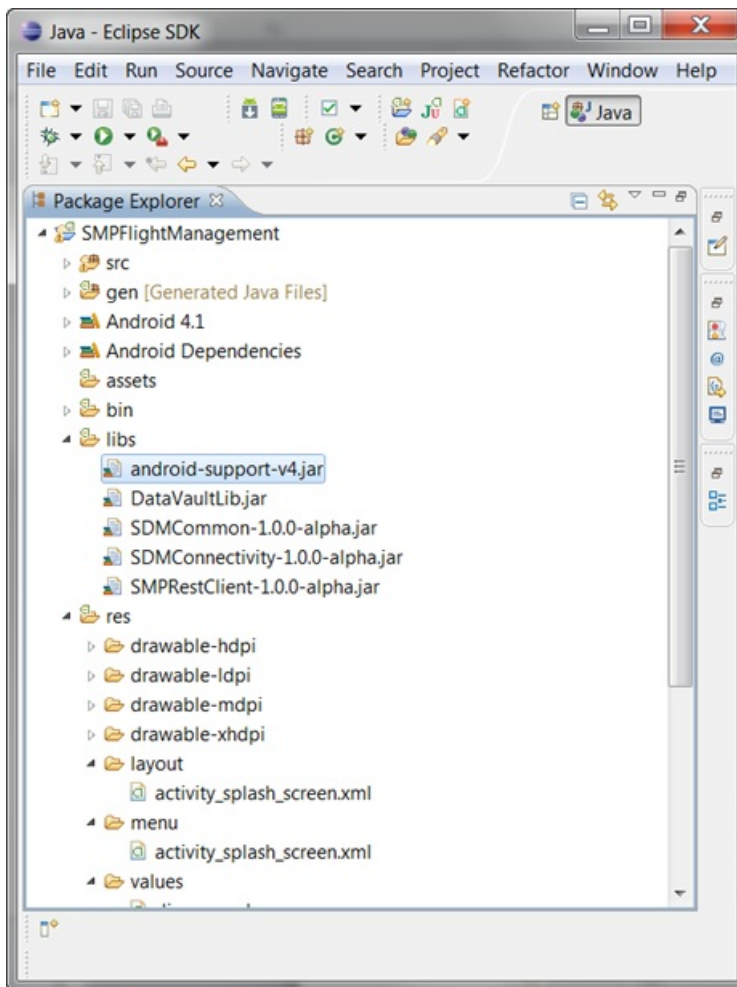
- `SMP_HOME\MobileSDK<Version>\OData\Android\libraries\:`
  - `SDMCommon.jar`
  - `SDMConnectivity.jar`
  - `SDMParser.jar`
  - `SMPRestClient.jar`
- `SMP_HOME\MobileSDK<Version>\OData\Android\libraries\Utils\:`  
`DataVaultLib.jar`

---

**Note:** You can add more jar files from the same location as required.

---

1. Click **Window > Show View > Package Explorer.in**



2. Navigate to **libs** folder under SMPFlightManagement.

## Adding User Permissions in Android Manifest File

Add user permissions to the Android project. Also add a **Detail Activity** class to the `AndroidManifest.xml` file. This declaration launches a customer detail screen where you can make changes when you test the application.

1. If needed, open the Android manifest file.
2. Select the **AndroidManifest.xml** tab.
3. Declare your activity and add user permission in the `AndroidManifest.xml` file:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/
android"
    package="smp.tutorial.android"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="15" />
    <uses-permission
android:name="android.permission.INTERNET"></uses-permission>
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" ></uses-
permission>
    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".SplashScreenActivity"
            android:label="@string/title_activity_splash_screen" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
                    android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

4. Select **File > Save**.

## Creating the User Interface

Use eclipse to build the user interface for the SMPFlightManagement application.

The SMPFlightManagement Android OData example project available at <http://scn.sap.com/docs/DOC-8803> contains the source code for the user interface for the sample application. You can build the user interface automatically by importing the source files to the eclipse project.

In this section you will be provided with the sample of XML layout which is created when you drag and drop the elements from **Palette** pane to **Graphical Layout** editor.

---

**Note:** You can add the source code in XML layout to match your requirement.

---

1. To add the empty Android XML file navigate to **Package Explorer** > **SMPFlightManagement** > **res** > **layout**.
2. Right click on layout folder and go to **New** > **Android XML File**.

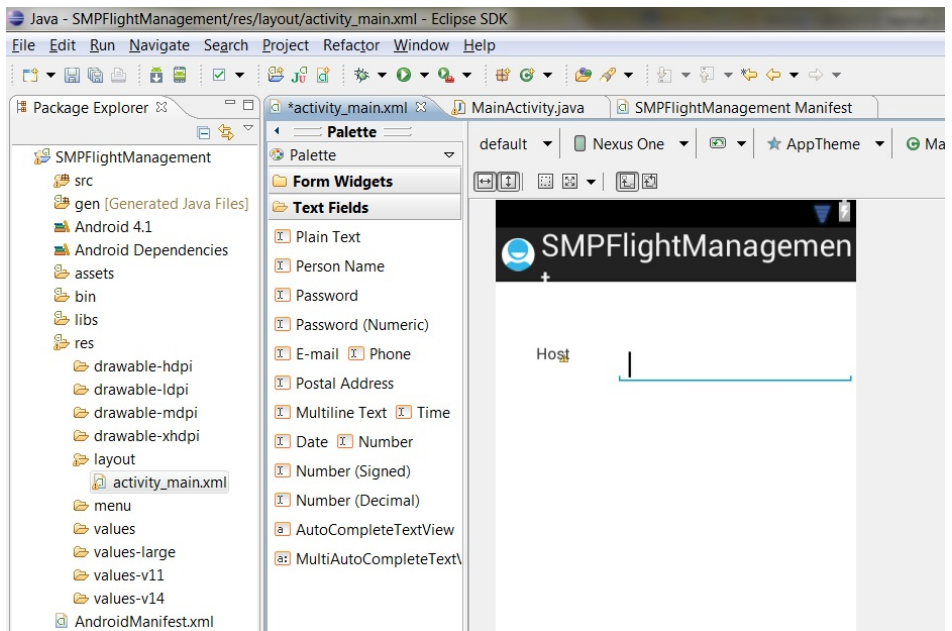
### Welcome screen

The welcome splash screen is displayed on the first launch of the SMPFlightManagement application. Create and include the image file. When the application is executed, it checks for the **sample\_flight.png**, and displays it as the first screen of the application.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/sample_flight" >
</RelativeLayout>
```

### Settings screen

Create the empty Android XML file and drag and drop the elements in **Graphical Layout** editor as shown below:



Below is the sample XML generated code of the text label and text box. You can add more elements in similar fashion.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentLeft="true"
            android:layout_alignParentTop="true"
            android:layout_marginLeft="36dp"
            android:layout_marginTop="55dp"
            android:text="Host" />

        <EditText
            android:id="@+id/editText1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignTop="@+id/textView1"
            android:layout_marginLeft="40dp"
            android:layout_toRightOf="@+id/textView1"
            android:ems="10"
            android:inputType="textPersonName" >

            <requestFocus />
        </EditText>

    </RelativeLayout>
```

### Application passcode screen

For subsequent launches you need to provide application passcode. Drag and drop the elements in **Graphical Layout** editor as required to retrieve the passcode as user inputs.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/sample_flight_bg"
    tools:context=".PasswordActivity" >

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="100dp"
        android:text="Enter Application Passcode"
        android:textAppearance="?android:attr/textAppearanceSmall" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
```

```

        android:layout_marginTop="134dp"
        android:ems="10"
        android:hint="Passcode"
        android:inputType="textPassword"
        android:textSize="15dip" >

        <requestFocus />
    </EditText>

    <Button
        android:id="@+id/button1"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/editText1"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="25dp"
        android:height="30dip"
        android:onClick="onLoginClick"
        android:text="Log In" />

</RelativeLayout>

```

### Carriers list screen

For displaying the list of flight carriers, add the code snippet for layout.

```

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <ListView
        android:id="@+id/android:list"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:cacheColorHint="#ff6a00"
        android:divider="#F0AB00"
        android:dividerHeight="1px"
        android:drawSelectorOnTop="false" >
    </ListView>

    <TextView
        android:id="@+id/label"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:padding="10dip"
        android:textColor="@android:color/black"
        android:textSize="24dip"
        android:textStyle="bold" />

</LinearLayout>

```

**See also**

- *Creating the Android Project* on page 16

## Defining the Application Logic using REST SDK

---

Create the application logic using REST SDK.

### Registering a User

Register a user using a predefined asynchronous authentication mechanism.

For registering the user, perform the following:

1. Establish the connection between the client and mobile platform server by providing details such as application ID, domain, security configuration and so on. . To Before initializing connection between client and server, you need to initialize the `SDMRequestManager` to get user the details. To initialize the `SDMRequestManager`, perform the following:
  - a. Initialize the `SDMLogger` method to obtain the mobile platform framework logs
  - b. Initialize the `SDMPreferences` method to provide the connection preference
  - c. Initialize the `SDMConnectivityParameters` method to provide user credentials and to enable XSRF (Cross-site request forgery) to avoid malicious exploit of website.

```
SDMLogger logger = new SDMLogger();
SDMPreferences pref = new SDMPreferences(context,
logger);
pref.setStringPreference(ISDMPreferences.SDM_CONNECTIVITY_HAND
LER_C
LASS_NAME, SDMConstants.SDM_HTTP_HANDLER_CLASS);
SDMConnectivityParameters param = new
SDMConnectivityParameters();
/*Set the Username and Password in
SDMConnectivityParameters.*/
param.setUsername(Utilities.userName);
param.setUserPassword(Utilities.password);
/*Enable XSRF support.*/
param.enableXsrf(true);

/* Initialize the SDMRequestManager */
SDMRequestManager reqMan = new
SDMRequestManager(logger,pref, param, 1);

/*Initialize the ClientConnection using
ClientConnection(<Context>,<Application ID>, <Domain>,
<SecurityConfig>,<SDMRequestManager>*/
ClientConnection cc = new
ClientConnection(getApplicationContext(),com.sap.NewFlight,
default, SSO ,reqMan);
```

Also, create the objects for user manager and application settings classes before implementation.

```

/*Initialize the UserManager using
UserManager(ClientConnection)*/
    UserManager um = new UserManager(cc);

/* Initialize the AppSettings class using
AppSettings(ClientConnection)*/
    AppSettings as = new AppSettings(cc);

```

## 2. Provide the server details.

```

/*Provide the server details such as the request type HTTP/HTTPS,
host, port, and so on. relay server URL template, farm ID (can be
set to "null" if relay server is not used) */
    cc.setConnectionProfile(isHttp,host,port,null, null);

```

## 3. Set the ISMPUserRegistrationListener listener for asynchronous registration.

```

/*set the Listener in case of Asynchronous Registration. This
class has to implement ISMPUserRegistrationListener to get the
callback*/
um.setUserRegistrationListener(this);

```

## 4. Register the user asynchronously.

```

/*Asynchronously register the user using
um.register(isSynchronous)*/
um.registerUser(false);

```

## 5. (optional) To check if the user registration is successful or not. Failure or success of registration is obtained in the callback onAsyncRegistrationResult.

```

@Override
    public void onAsyncRegistrationResult(State
registrationState, ClientConnection clientConnection, int
errCode, String errMsg)
    {
        if(registrationState ==
ISMPUserRegistrationListener.State.SUCCESS)
        {
            Log.i("Tutorial", "Registration
successful");
        }
    }

```

## Sending Data Request to the Backend

Send a data request to the backend through the server, either synchronously or asynchronously.

Initialize the SDMBaseRequest object to send the data to the server such as backend URL, request type, and ISDMNetListener.

```

/*Initialize the SDMBaseRequest and set the following*/
SDMBaseRequest getrequest = new SDMBaseRequest();

/*set the endpoint URL. This can be obtained by
AppSettings.getApplicationEndPoint()*/
getrequest.setRequestUrl(as.getApplicationEndPoint());

/*set whether GET/PUT/POST/DELETE*/

```

```
getRequest.setRequestMethod(SDMPBaseRequest.REQUEST_METHOD_GET);

/*All request-response operations are Asynchronous. Set the listener
to handle the response.This class has to implement ISDMNetListener to
get the callback.*/
getRequest.setListener(this);
```

Send the request using already defined SDMPRequestManager in *Registering a User* on page 26.

```
/*Perform the request using the above defined SDMPRequestManager.*/
reqMan.makeRequest(getRequest);
```

## Retrieving the Response from Backend

Retrieve the response from the backend to the device.

Server response will be obtained in the form of callbacks `onError` and `onSuccess`:

- `onError`: **public void onError(ISDMRequest arg0, ISDMResponse arg1, ISDMRequestStateElement arg2)**
- `onSuccess`: **public void onSuccess(ISDMRequest aRequest, ISDMResponse aResponse)**

On successful operation, `onSuccess` method will be invoked. Perform the following in `onSuccess` method to parse the response.

1. Create the `SDMPParser` object.
2. Depending on the type of queried document (service document, metadata document, and so on).

```
/*Upon performing request using
SDMPRequestManager.makeRequest(SDMPBaseRequest), the response will
arrive at either public void onError(ISDMRequest arg0, ISDMResponse
arg1, ISDMRequestStateElement arg2) {} or public void
onSuccess(ISDMRequest aRequest, ISDMResponse aResponse) {} depending
on whether the transaction was a failure or success respectively.
Add the following in the onSuccess() method: */
```

```
    HttpEntity responseEntity = aResponse.getEntity();
    SDMPParser parser = new SDMPParser(mPreferences, mLogger);
    /*The response parsing depends on the type of document queried*/
    if(serviceDoc)
    {
        ISDMDataServiceDocument serviceDoc =
        parser.parseSDMODataServiceDocumentXML(EntityUtils.toString(responseEntity));
    }
    else if(metadata)
    {
        ISDMDataSchema metaDoc =
        parser.parseSDMODataSchemaXML(EntityUtils.toString(responseEntity),
        serviceDoc);
    }
```



```
else if(data)
{
    List<ISDMODataEntry> dataSet =
    parser.parseSDMODataEntriesXML(EntityUtils.toString(responseEntity)
    , "CarrierCollection", metaDoc);
}
```

#### Consume the parse data.

```
/*Iterate through the entries and get the required fields*/
Iterator<ISDMODataEntry> iterator = dataSet.iterator();
while (iterator.hasNext())
{Map<String,String> propertyValues =
iterator.next().getPropertyValues();
Log.i("SMP",propertyValues.get("CARRNAME")
+"\n"+propertyValues.get("carrid"));
}
```

Develop the Android OData Application

# Running your Android OData Application

Run the SMPFlightManagement Android application on real device or on emulator.

## Prerequisites

- Register an application connection in SAP Control Center or SAP Mobile Platform, enterprise edition, cloud version - Administration and Monitoring for On-Premise or Cloud respectively.
- The device or emulator must be connected to the server.

## Task

You can run your application either on the real Android-powered device or the Android emulator. This section shows how to run your application on the device emulator. For this you first need to create an Android Virtual Device (AVD). For more information on how to create AVD, see *Run on the Emulator*.

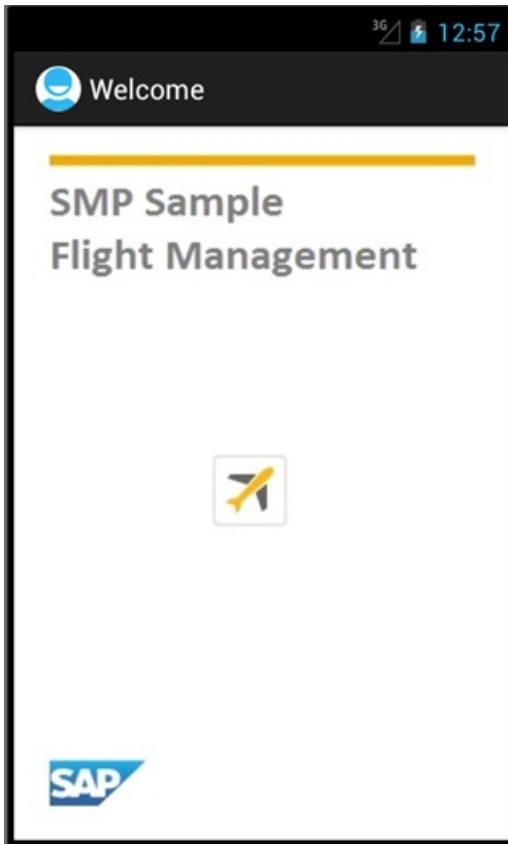
1. In WorkSpace Navigator, right-click **SMPFlightManagement** and select **Run As > Android Application**.

---

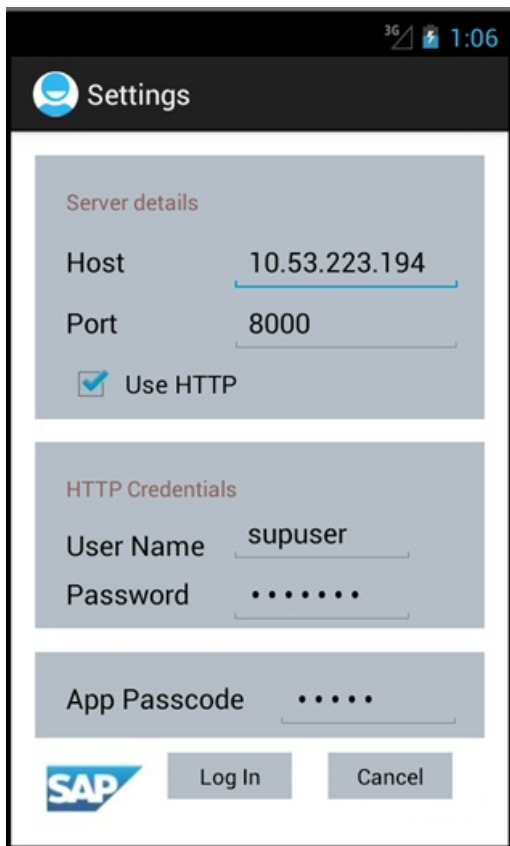
**Note:** It may take several minutes for the Android emulator's home screen to appear.

---

2. In the Android applications screen, open the **SMPFlightManagement** application's welcome page.



3. When you run the application for the first time, it exits immediately with a dialog asking you to enter the application settings in the **Settings** page.
4. In the Android simulator, go to **Settings** to enter the server details and HTTP credentials.
  - **Host** – The name of the machine where the Server is running.
  - **Port** – Server port number. The default port number for HTTP channel is 8000.
  - **Use HTTP** – Enable if you want to establish HTTP connection between client and server.
  - **User Name** – The name of the user.
  - **Password** – The user account password used to authenticate the user.
  - **App Passcode** – The application pin to securely store your application password, and a database encryption key that is generated when the application launches.



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



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



7. Close the emulator to stop the SMPFlightManagement Android application.





# 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&c=firsttab&a=0&p=categories>, then navigate to the most current version.

## *Tutorials*

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

## *Example Projects*

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

## *Samples*

Sample applications are fully developed, working applications that demonstrate the features and capabilities of 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*

## Learn More About SAP Mobile Platform

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

Customize and automate:

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

Javadoc and HeaderDoc are also available in the installation directory.

# Index

## A

- Android project 16
  - manifest file 22
- Android SDK 16
- AndroidManifest.xml 16
  - Detail Activity 22

## B

- build path 16

## C

- ClientLib.jar 16

## D

- Detail Activity 22
- Developing Android OData Application 15

## E

- example projects 1

## H

- Hybrid App package tutorial 1

## J

- JAR files
  - ClientLib.jar 16
  - sup-client.jar 16
  - UltraLiteJNI12.jar 16
- JDK 16

## M

- manifest file 16, 22
- mobile business object tutorial 1

## O

- Object API tutorials 1

## P

- project build path 16

## S

- samples
  - downloading 37
- SAP Control Center
  - connecting to 4
- SAP Mobile Platform
  - documentation resources 37
  - getting started 3
  - installing 3
- SAP Mobile Platform Runtime
  - installing 3
- SAP Mobile Platform services 4
- SAP Mobile Platform, enterprise edition, cloud
  - version
    - getting started 9
- SAP Mobile SDK
  - installing 3
- simulator 16
- sup-client.jar 16

## T

- tutorials 1
  - downloading 37

## U

- UltraLiteJNI12.jar 16
- User interface 22

## V

- virtual devices 16

