MicroStrategy | 2020

Mobile SDK for Android

Core learning for: Mobile Architects

CONTENTS

3. Introduction to Android Studio

4. Mobile Architecture, Security and Configuration

5. Customize Mobile Configurations, Icons, and Splash Screen

6. Build and Deploy Your App

1

Pre-Course Setup

The first stage of this course begins by setting up the virtual MicroStrategy Cloud environment to support the exercises that you complete throughout the course.

You can reuse the steps in this chapter to reset the environment to its original state, if you elect to redo the exercises.

MicroStrategy Cloud environment

The MicroStrategy Cloud environment is composed of two servers. The Linux server contains the MicroStrategy platform and main web server. The Windows server, on the remote desktop, contains a local web server with MicroStrategy Web that can be used for development and testing purposes.

To prepare for the course, access your MicroStrategy Cloud environment and configure the connection to the remote desktop, also known as the Windows Developer Machine. The login credentials and other information you need to access your environment are included in the MicroStrategy Cloud email that you received.

Exercise 1.1: Configure the Remote Desktop connection

In this exercise, you access your MicroStrategy Cloud environment and configure a remote desktop connection.

Access the MicroStrategy platform

1. On your local machine, in the MicroStrategy Cloud email, click **Access MicroStrategy Platform**.
2. Log in using the credentials provided in the MicroStrategy Cloud email.

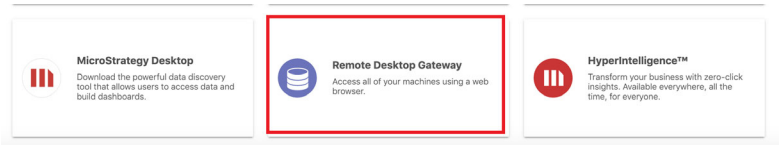
   The MicroStrategy Cloud landing page is displayed.

Access the Remote Desktop

There are two methods to connect to the remote desktop instance of your MicroStrategy Cloud environment. You can connect using the Remote Desktop Gateway or by configuring a local Remote Desktop Connection.

Option 1: Remote Desktop Gateway

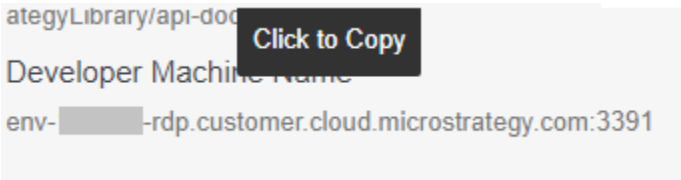
1. From the MicroStrategy Cloud landing page, scroll down to the **More Resources** section.
2. Select **Remote Desktop Gateway**.



3. Log in using the credentials provided in the MicroStrategy Cloud email.
4. Under All Connections, select **Developer Instance RDP**.

Option 2: Local Remote Desktop Connection

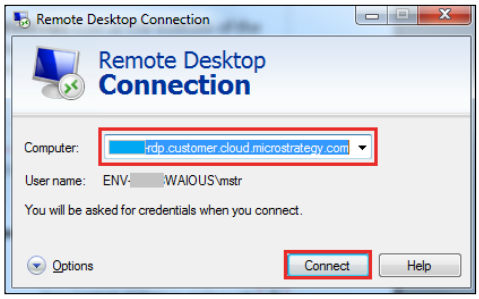
1 In the Essential Connections area, hover over **Developer Machine Name** and click **Copy** to copy the address to your clipboard. For example, the address is similar to **env-12345-rdp.customer.cloud.microstrategy.com:3391**.



2 On your local Windows machine, from the taskbar, search for **Remote Desktop**. From the search results, select **Remote Desktop Connection**.

This step may differ, depending on the version of Windows operating system on your computer.

3 In the Remote Desktop Connection window, in the **Computer** box, paste the machine name, and click **Connect**.



4 In the Windows Security window, type the username and password from the MicroStrategy Cloud email, and click **OK**.

If an identity verification message is displayed, click **Yes**.

You are now connected to the Windows machine in your MicroStrategy Cloud environment.

Preparing the environment

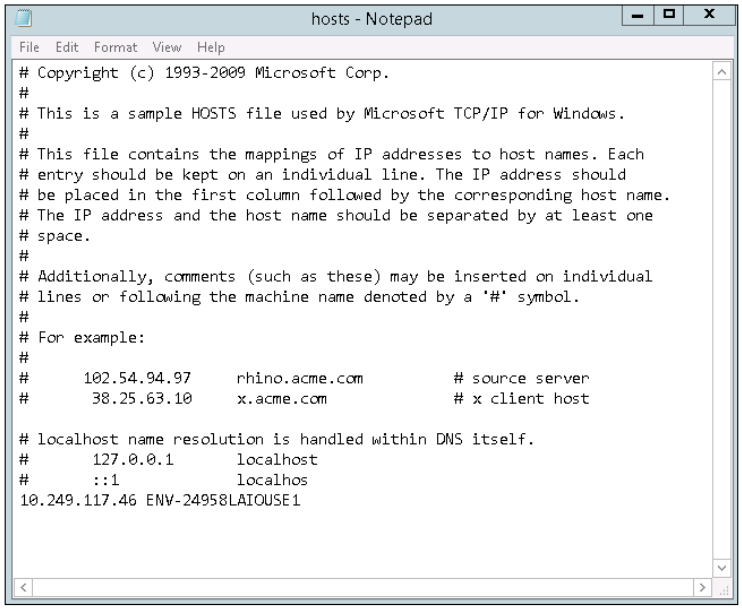
Now that you have configured the Remote Desktop application and have successfully connected to your Developer machine, you are ready to prepare the environment.

Reviewing the Windows hosts file

The hosts file cross-references specific IP addresses with user-friendly names, just like a Domain Name Server (DNS) that translates IP addresses to human-readable website names as you browse the Internet. DNS translation enables you to navigate to www.microstrategy.com instead of a forgettable series of numbers like 104.121.85.249.

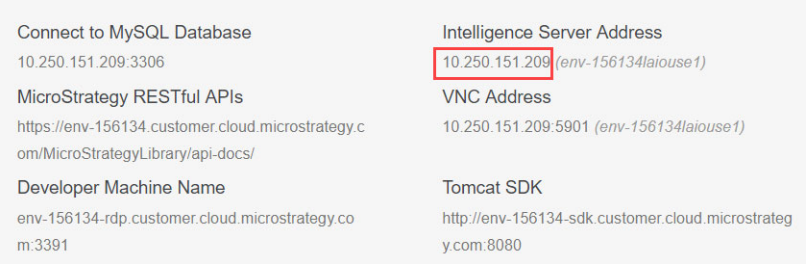
The hosts file serves a similar purpose, but it operates locally on a specific computer. We use the hosts file to map the Intelligence Server IP address to the readable machine name, as in the following example:

10.249.117.46 ENV-24958LAIOUSE1



In the hosts file, the IP address is mapped to the machine name using the following information:

- **XXX.XXX.XX.XXX**: The IP address of the Intelligence Server located in the Essential Connections area of the MicroStrategy Cloud landing page, as shown in the image below. To copy the address, hover over it and click **Copy**.



- **TAB**: A tabulation character must separate the first and third parts of the line. A space does not work.
- **ENV-XXXXLAIOUSE**: The Intelligence Server machine name, where XXXX is your environment's unique ID number located in the Essential Connections area of the MicroStrategy Cloud landing page, as shown in the image below.

Your cloud environment

The MicroStrategy Cloud environment is composed of two virtual servers. In this section, we explore the environment configuration to identify component locations and connections.

Linux server

The first server in this environment has a Linux operating system. It serves many purposes, including supporting the:

- Intelligence Server (IS)
- Web server (Tomcat)
- Mobile Server
- MicroStrategy Identity Server

Browsing to MicroStrategy Web

To access MicroStrategy Web, you use an address with the following format:

https://env-XXXXX.customer.cloud.microstrategy.com/MicroStrategy/servlet/mstrWeb

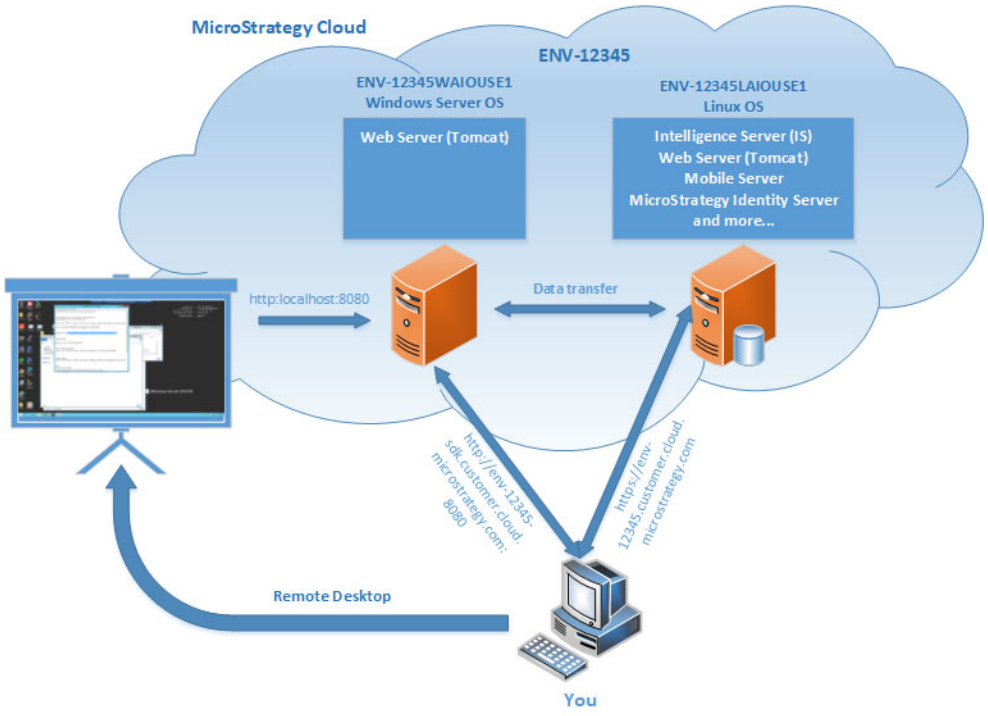
where XXXXX represents the environment number, as indicated in the MicroStrategy Cloud email. Because the address includes a "servlet", you can deduce that you are accessing a Tomcat web server.

Various requests are processed within the same physical machine using logical components like the web server, MicroStrategy Identity Server, and Intelligence Server.

Remote access

You can access the Linux server with tools like VNC, Putty, or WinSCP, depending on your need for a full remote desktop experience or a simple file transfer interface.

This diagram shows the remote connections in your MicroStrategy Cloud environment.



Windows server

For specific MicroStrategy Education classes (mainly the Administrative and SDK classes), a second server hosting a Windows operating system is provided for the following purposes:

- Web server (local second instance, and sometimes third instance of Tomcat)
- MicroStrategy Web Server (through the second instance of Tomcat)
- MicroStrategy Mobile Server (through the second instance of Tomcat)
- Java applet server (through the local Tomcat)
- Liferay portal host (for specific classes, using the third instance of Tomcat)
- Access to MicroStrategy Developer
- Access to MicroStrategy Workstation
- Access to R tools to integrate in MicroStrategy reports

Browsing to MicroStrategy Web

You can access MicroStrategy Web using the following address:

https://env-XXXXX.customer.cloud.microstrategy.com/MicroStrategy/servlet/mstrWeb

where XXXXX represents the environment number, as indicated in the MicroStrategy Cloud email.

You can access MicroStrategy data through the second Tomcat web server located on the Windows server using the following address:

http://env-XXXXX-sdk.customer.cloud.microstrategy.com:8080/MicroStrategy/servlet/mstrWeb

where XXXXX represents the environment number, as indicated in the MicroStrategy Cloud email you received. Notice the -sdk portion of the web address, which indicates that you are accessing MicroStrategy Intelligence Server through the Windows machine's Tomcat instance. Notice also that we are using a different port (8080) for this web server, which is a common Tomcat practice.

You can also access MicroStrategy Web from a browser on the Windows Developer Machine, using the following address:

http://localhost:8080/MicroStrategy/servlet/mstrWeb

Again, port 8080 is used by the Tomcat web server. You must use a browser on the Windows Developer Machine for this address to function. To access the web server on the Windows machine from a different location, use the -sdk address.

Remote access

Remote access is required to access the applications or file system on the Windows machine through one of the following options:

- A browser-based remote access application available from the Cloud landing page
- A dedicated remote desktop application, available for Windows, MacOS, and Linux

For a visual representation of the MicroStrategy Cloud environment, refer to *Appendix 1, This diagram shows the remote connections in your MicroStrategy Cloud environment.*

The Education scenario presented includes two servers. However, a typical implementation of the MicroStrategy platform can include a single server where all MicroStrategy components are installed, or multiple servers that each house distinct MicroStrategy components. The implementation in your organization depends on traffic volume considerations and specific needs like security requirements or deployment conditions.

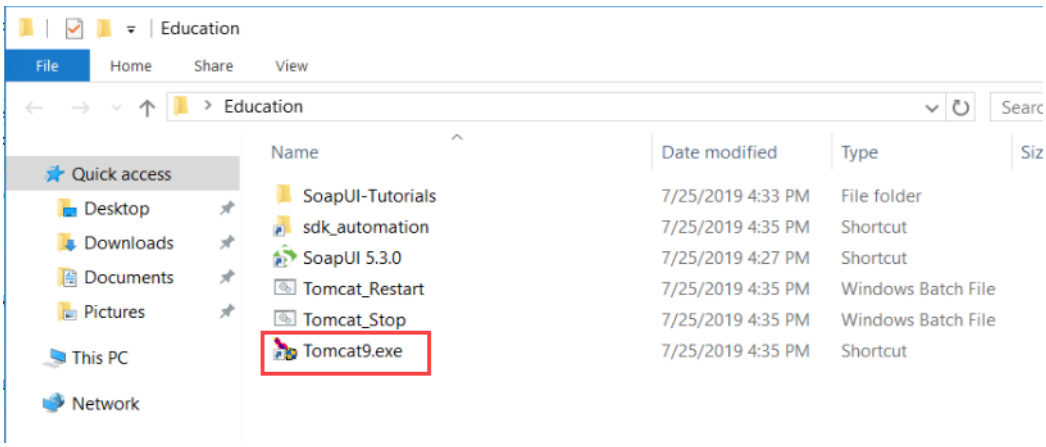
Exercise 1.2: Close Eclipse and stop your Tomcat web server

To complete the software installation for this class, you must first close the Eclipse IDE and stop the Tomcat web server on the Windows machine. To ensure proper performance on the Tomcat server, verify that an appropriate amount of memory is allocated to the Tomcat server's Java Virtual Machine.

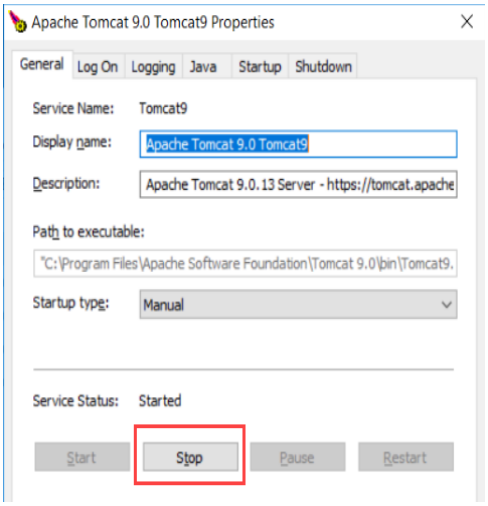
Close Eclipse and stop Tomcat

1 On the Remote desktop, double-click the **Education** folder.

2 Double-click **Tomcat9.exe**.



3 If the service is currently running, in the Service Status area, click **Stop**.



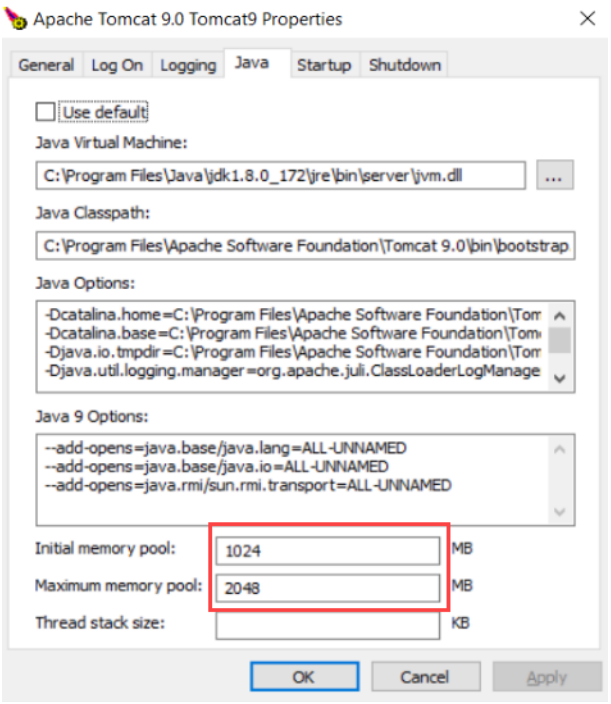
Alternatively you can use a script located on your desktop. To do this, right-click **Tomcat_Stop.bat** and click **Run as Administrator**.

Verify Tomcat's Java memory allocation

4 Click the **Java** tab and ensure the following settings are applied:

a Set the **Initial memory pool** to **1024**.

b Set the **Maximum memory pool** to **2048**.



5 If the Eclipse environment is open, save your work and close it.

Exercise 1.3: Update Internet Explorer security settings and turn off Windows Firewall

To ensure that you are able to access the required web sites for this class using, update the list of trusted sites and turn off the Windows Firewall.

Add trusted sites to Internet Explorer

1 On the remote desktop, open **Internet Explorer**.

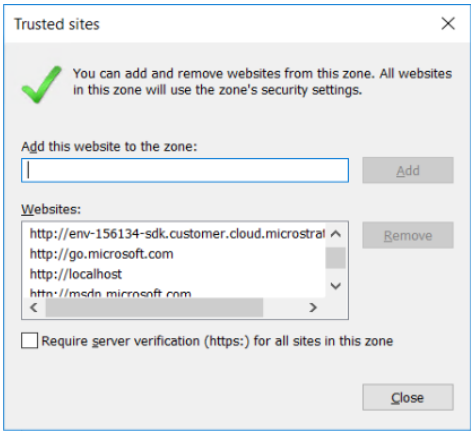
2 In the Set up Internet Explorer window, click **OK**.

3 From the **Tools** menu, select **Internet Options**.

4 On the **Security** tab, click **Trusted Sites**.

5 Click **Sites,** and add the following sites to the list:

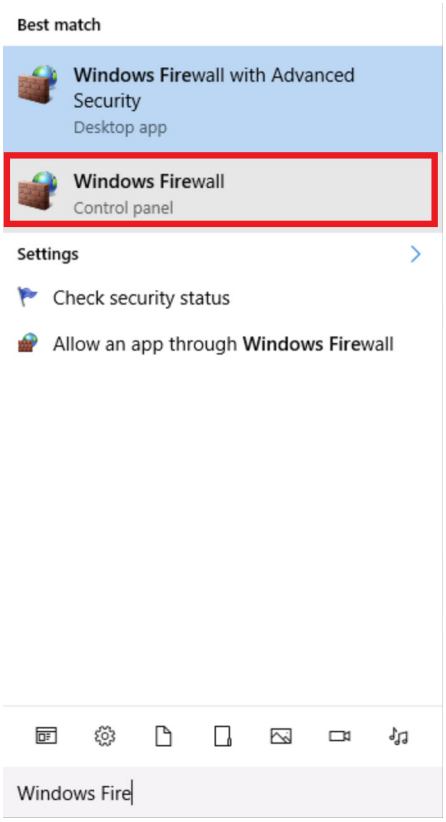
- **http://localhost**
- **http://*.microstrategy.com**
- **https://*.microstrategy.com**
- **http://env-XXXX-sdk.customer.cloud.microstrategy.com**, where XXXX is your unique environment ID.



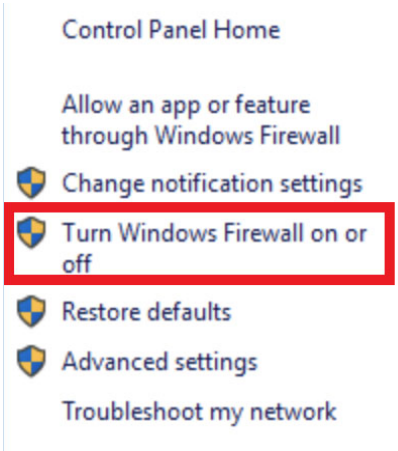
6 Click **Close**.

7 Click **OK** to close Internet Options.

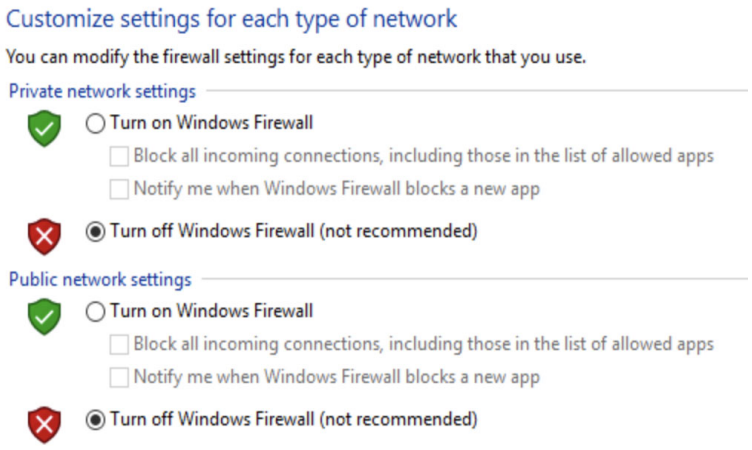
8 In the **Search Windows** text box, search for and select **Windows Firewall**.



9 On the left, select **Turn Windows Firewall on or off**.



10 Turn off the Windows Firewall for both the Private and Public network settings.



11 Click **OK** and close the Control Panel window.

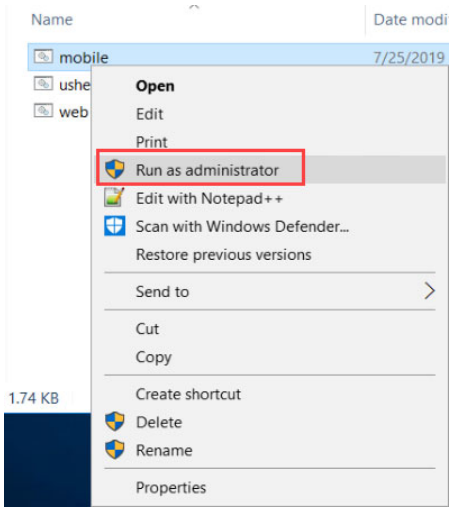
Exercise 1.4: Run the final installation script

The exercises in this class require additional files and software to be installed. In this section, install the files using an automated script.

Run mobile.bat

1 In File Explorer on the remote desktop, navigate to the **C:\sdk_automation\scripts** folder.

2 Right-click **mobile.bat** and select **Run as administrator**. In the User Account window, click **Yes**.



Run the script only once, as the script deletes all of your progress if it is executed twice. If you would like to revert the environment to a brand-new state, you can run the script again.

Once the script ends, the command prompt window closes automatically. This might take a few minutes.

Exercise 1.5: Configure MicroStrategy Web

You ran the installation script and you are ready to verify that the installation is complete. To do this, restart the web server and log in to the local version of MicroStrategy Web, then connect your local version of Web to the Intelligence Server.

Validate your established connections

1 From the remote desktop, right-click the **Tomcat_Restart.bat** script and click **Run as administrator**. In the User Account window, click **Yes**. The Tomcat Web server restarts after a few minutes.

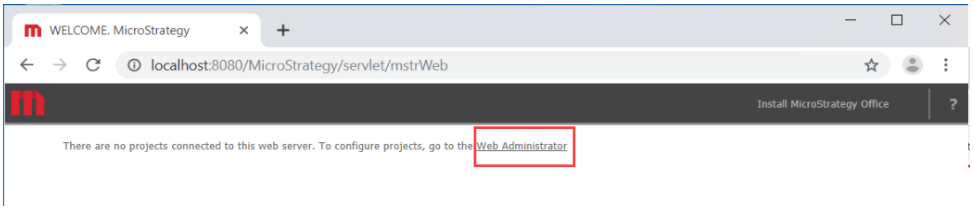
2 From the Windows desktop, double-click **Google Chrome**.

If you want to make Chrome the default browser on the Windows machine, complete the steps displayed in the browser.

3 Navigate to the following URL to validate that Tomcat and MicroStrategy Web are running correctly:

http://localhost:8080/MicroStrategy/servlet/mstrWeb

4 Click the **Web Administrator** link.



5 In the Sign in window, enter the following username and password:

- Username = **admin**
- Password =**sdkws**

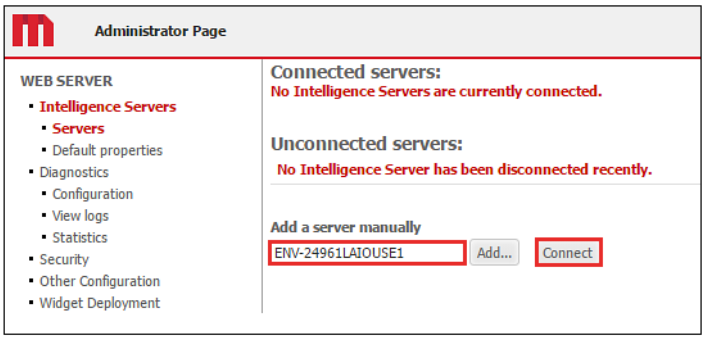
The MicroStrategy Web Administrator page opens.

Connect your local instance of MicroStrategy Web to the Intelligence Server

6 On the remote desktop, open **hosts.txt** in **Notepad**.

7 At the bottom of the hosts.txt file, copy the Intelligence Server machine name, for example, env-24961laiouse1.

8 On the Administrator Page, in the **Add a server manually** box, paste the copied Intelligence Server machine name and click **Connect**.



9 Click the **MicroStrategy Web Home** link.

A list of connected projects is displayed. For all subsequent exercises in this course, access MicroStrategy Web using the Chrome browser on the Windows Developer Machine. This ensures that you are able to access custom files used in this course.

10 **Minimize** the MicroStrategy Web window.

Optional Class Hardware

To have the best experience in this class, you should have an Android phone or Android tablet at your disposal. If you want to test your application in real-world scenarios, use an actual Android phone or tablet. Any serious application should be tested on an actual device. Emulators are fine for development, but cannot be used as the only testing ground for your applications.

Development tools

Several development tools are required to create and test mobile applications. Some of these tools are installed on the remote desktop, while others are required on your local computer.

Tools on the Remote Desktop

For this class, your MicroStrategy Cloud environment is already set up with the following development tool. If you want to perform the exercises on a different device, install the following software:

- **Apache Tomcat 8**

  https://tomcat.apache.org/download-90.cgi

Tools on Local computer

A computer with the following software is required to develop Android mobile applications:

- **Android Studio integrated development environment** (use the version recommended by your instructor)

  https://developer.android.com/studio/install.html

  The Android Studio installation installs Android Studio as well as the Android SDK and simulators for various Android devices.

Class files

Your instructor will provide an archive of files needed to complete the exercises in this course.

Exercise 1.6: Extract the class files

The exercise files for this class are in an archive that need to be available on both the remote desktop and your local computer.

Extract the class files

1. Place the **Android_Mobile_SDK.zip** folder on the desktop of your local computer.
2. Right-click **Android_Mobile_SDK.zip** and select **Extract All**.
3. In the Select a Destination window, click **Extract**. A folder named **Android_Mobile_SDK** is automatically created.

The Android_Mobile_SDK folder contains the following folders:

- **Exercises:** Resources required for the class exercises.
- **Logos:** The logo and splash screen image used in the customization.
- **MicroStrategy SDK:** Where you will paste the Mobile SDK files you download.
- **My Exercises:** Where you will save your applications you create in class.

Your environment is configured, and you are ready to start exploring the MicroStrategy Mobile SDK for Android. Let's get started.

2

Discovering and Installing MicroStrategy Mobile SDK

In this course, you develop an understanding of the MicroStrategy Mobile Application Programming Interface (API) and Software Development Kit (SDK). You also learn to customize the MicroStrategy Mobile application and change its behavior.

You gain insights on how to rebrand, preconfigure, and customize your app. You also learn how to take advantage of MicroStrategy's security and authentication functionality and features.

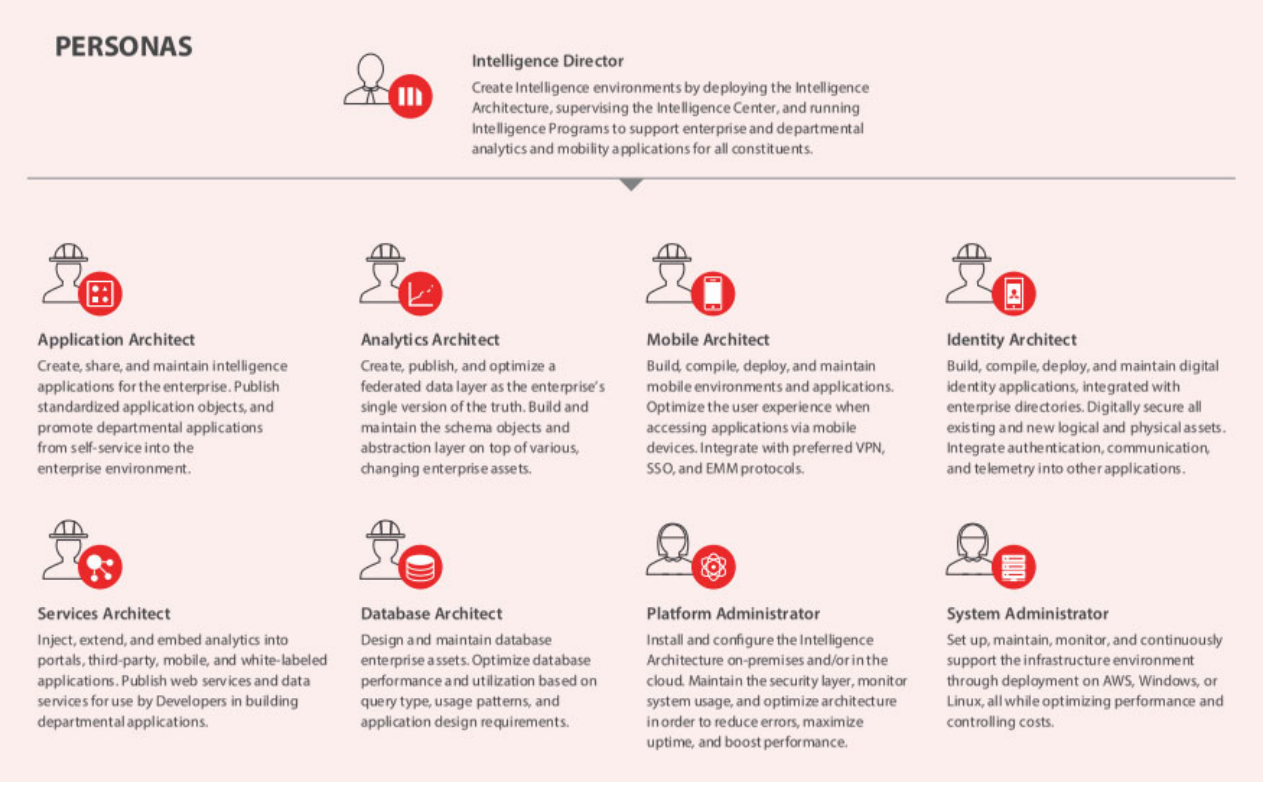
By the end of this chapter, you will be able to:

- Discuss development in an intelligent enterprise.
- Describe the MicroStrategy Mobile SDK.
- Explain the three target scenarios MicroStrategy supports with the MicroStrategy SDK.
- Explore the levels of customization that can be achieved with MicroStrategy SDK.
- Install the MicroStrategy Mobile SDK on your computer.

Developing in an Intelligent Enterprise

MicroStrategy provides self-service Business Intelligence (BI) with MicroStrategy Desktop, and Enterprise BI with data governance, enterprise-level security, user administration, and big data support, all centered around the Intelligence Server. The Intelligence Server delivers world-class monitoring, reporting, and analysis on one integrated platform, offering next-generation BI capabilities for the full range of BI applications. MicroStrategy's enterprise platform is the core of the Intelligent Enterprise.

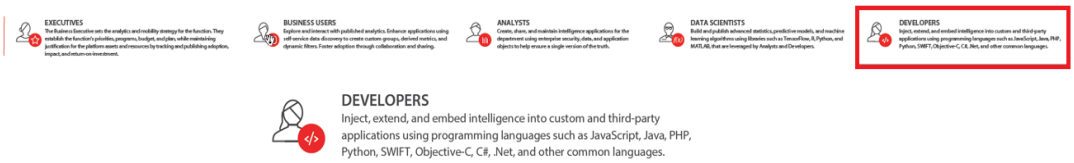
Supporting the Intelligent Enterprise is the Intelligence Center, comprised of a team of expert architects who define, develop, and provide guidance across the enterprise.



The Services Architect is responsible for creating and implementing the guidelines and best practices to inject, embed, and extend the MicroStrategy analytics capabilities into custom applications and other development efforts. White labeling and web services are also under his purview. Any development initiative must follow the standards established by the Services Architect.

As a developer, you already know why standards are important. Even a small customization can become a nightmare to maintain and support if coded

haphazardly. Every effort made to support the Intelligent Enterprise's best practices is a step in the right direction.



The scope of a customization can take on a life of its own. It starts as a change made for a department, and then suddenly other departments want to use it. After a while, the customization attracts the attention of a CXO and that small customization is destined to become an enterprise standard. The small customization good for a few users now becomes adopted in a completely different scale, enterprise wide. Considerations such as federated data, performance, and security may have slipped through the cracks when built for just a few people.

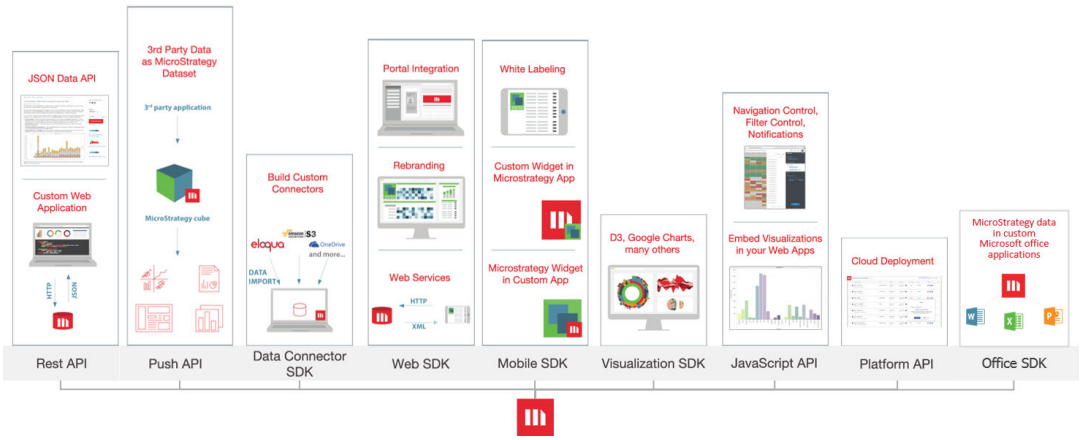
Using the standards set in place by the Services Architect of the Intelligence Center, even a developers small project can have the appropriate structure to evolve organically without much effort during its evolution from your desk to the CEO's desk.

Whether your organization is small or large, the Intelligent Enterprise is a model to pursue. It offers the opportunity to get the most out of the MicroStrategy platform and contribute to giving you that intelligence edge to make your enterprise as successful as it can be.

MicroStrategy SDK

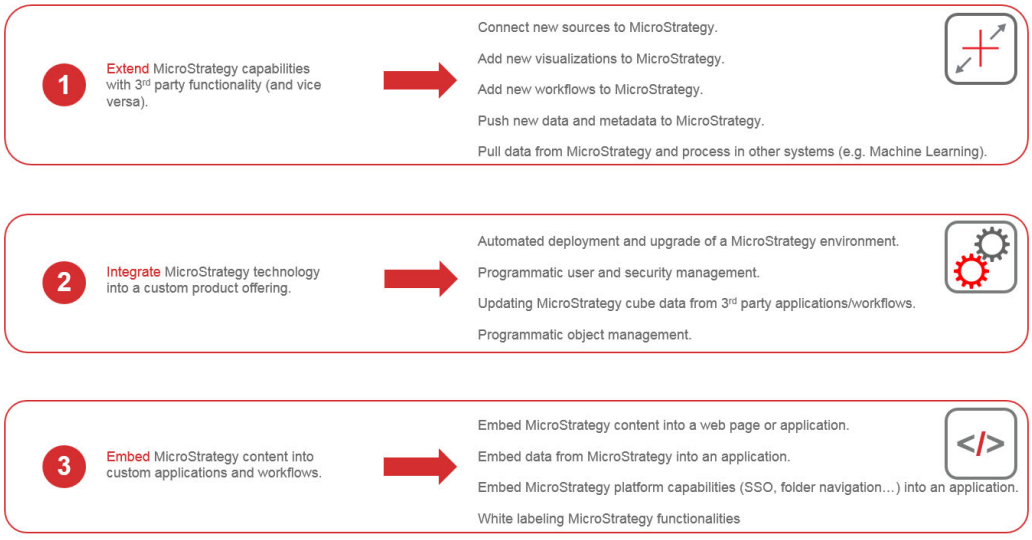
The analytics market is diverse, growing, and full of innovations. MicroStrategy offers a set of SDKs and APIs that empowers users and organizations to implement highly customized, functional, and powerful solutions that tap into the heart of the platform to deliver the information when, where, and how you want it. MicroStrategy SDK is an umbrella term that represents a large family of

SDKs and APIs. This chapter, provides a high-level tour of the various SDKs and APIs that MicroStrategy offers.



Target scenarios behind the MicroStrategy SDK

The key target scenarios that are the constant driving force behind the evolution of the MicroStrategy SDK and API suite are:



This three-sided approach (extend – integrate – embed) is at the center of the SDK and API offerings. As an independent enterprise analytics vendor, MicroStrategy is in a unique position to have integration points and SDKs that allow integration across an enterprise ecosystem.

The SDKs and APIs allow partners to push data (for example, after data wrangling) into the MicroStrategy platform and pull data (for example, to visualize in another tool or app) from the same platform.

Further, the visualization plug-in API has been opened and enables REST APIs to embed MicroStrategy content into other applications.

Extend

The suite aims to extend MicroStrategy capabilities with third-party functionality (and vice-versa). This can be seen in connecting to new data sources, adding new visualizations, following new workflows, pushing new data and metadata into MicroStrategy, and even pulling data from MicroStrategy to process it in other systems, like in machine learning.

Integrate

The MicroStrategy SDK supports the integration of MicroStrategy technology into a custom product offering. Examples include, automated deployment and upgrade of a MicroStrategy environment, programmatic user and security management, update of a MicroStrategy data cube from a third-party application or workflow, and programmatic object management.

Embed

As embedded analytics is a highly sought feature, the SDKs and APIs assist you in reaching this objective. You can embed MicroStrategy content into a web page or application, embed MicroStrategy platform capabilities (SSO, folder navigation, and so on) into an application, and even white label MicroStrategy functionalities.

MicroStrategy Mobile overview

Deployment of your BI with the MicroStrategy Mobile app extends report access to mobile devices. This transformational mobility enables much more range and flexibility in how and where users review vital information, monitor KPIs, and share data in and out of the workplace. With your mobile device, you can easily pass a dossier around a meeting, make on-site reviews of off-site facilities, and create geolocation and bar code reader prompts for reports.

MicroStrategy Mobile works seamlessly with the MicroStrategy platform to minimize the amount of work necessary to prepare a dynamic dossier for mobile deployment. No new skills are required to construct attractive and stable apps for the iPhone, iPad, or Android. MicroStrategy Mobile provides additional report and

document manipulation options as well as mobile device-specific widgets that better suit the medium.

Mobile deployment consists of the following two main components:

- **The MicroStrategy Mobile Server**: The Mobile Server functions analogously to the web server that supports MicroStrategy Web. It connects to the MicroStrategy Intelligence Server to retrieve BI data and pass it on to MicroStrategy Mobile clients.

  The Mobile Server is already pre-configured in your MicroStrategy Cloud installation. A complete setup is done in the Linux Server, while a basic setup was completed in the Windows server in a previous chapter. Further modifications of this server are part of the learning experiences of this class.

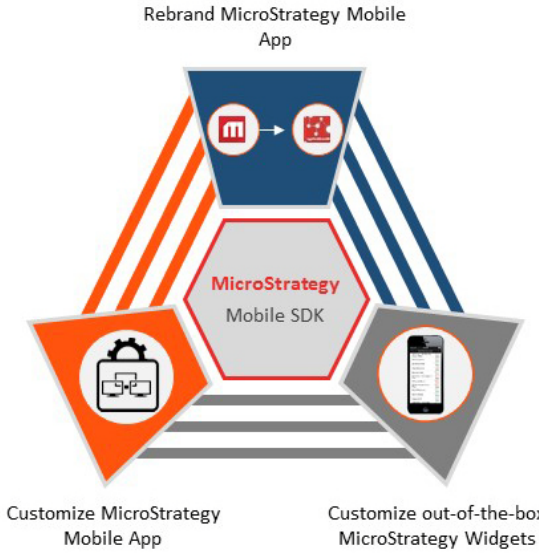
- **The MicroStrategy Mobile app**: MicroStrategy Mobile apps are clients installed on mobile devices that enable you to access and interact with the MicroStrategy BI data. They serve as viewers.

  The generic Mobile app is a canvas you use and adapt to your business needs. This can be a single app doing everything or multiple copies each with their own modifications to respond to specific needs or audiences.

Mobile SDK

The MicroStrategy Mobile SDK is a member of the comprehensive family of SDKs that MicroStrategy offers. The Mobile SDK provides resources to customize MicroStrategy Mobile on Android and iOS mobile devices, and the MicroStrategy Mobile Server. Using the Mobile SDK, you can change the look and feel of the

MicroStrategy Mobile app, and create custom widgets and apps that use MicroStrategy BI data.

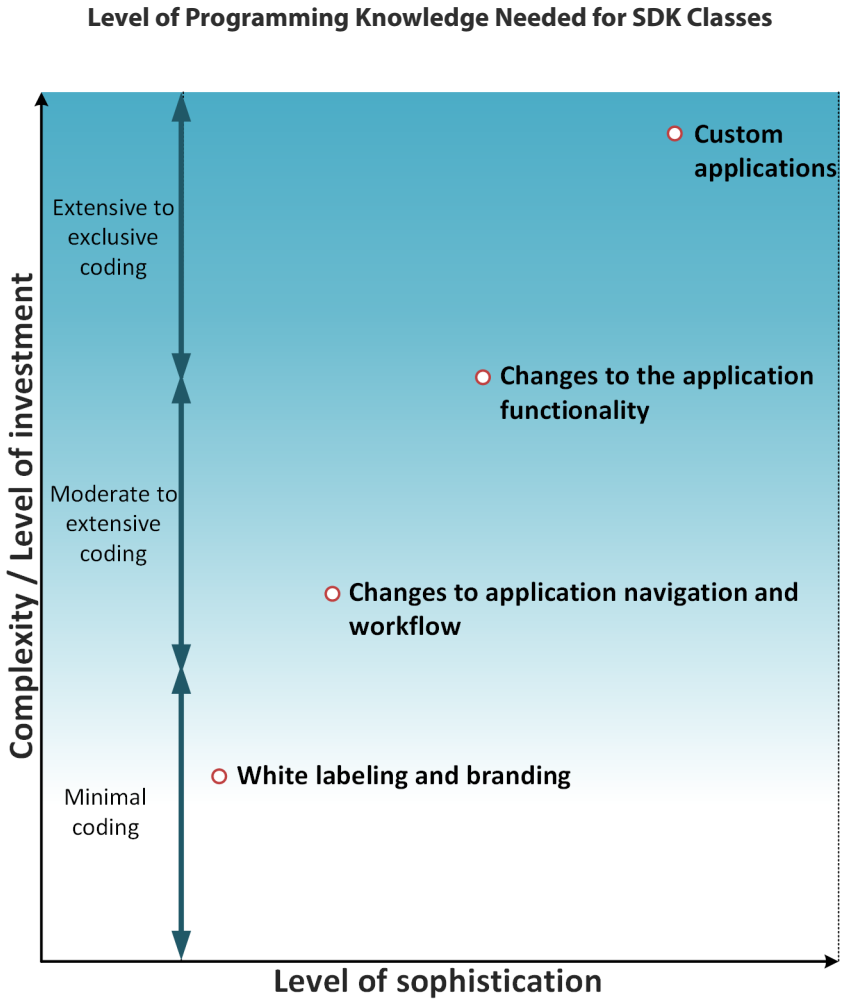


The image above depicts using Mobile SDK to achieve the following common customization goals:

- Customize mobile apps built using the MicroStrategy Mobile app platform, for Android and iOS mobile devices.
- Rebrand and preconfigure your app, customize it to use custom help files, or perform actions such as disabling software encryption and performing specific checks before launching.
- Use MicroStrategy's wide array of out-of-the-box widgets, as well as create custom widgets that extend MicroStrategy functionality.

Levels of customizations

You can customize the MicroStrategy platform in many ways and at different levels of sophistication. Balance this with the level of ease to perform those customizations and decide what is possible for you and your organization.



Minimal coding

You can customize many aspects of the MicroStrategy platform with minimal coding knowledge. For example, you can:

- Add a Logo to Report Services documents and interactive dossiers
- Use a new application icon and splash screen
- Display different headers and footers

MicroStrategy Web and MicroStrategy Mobile are targets for such scenarios. This class discusses customization of MicroStrategy Mobile. If you are interested in MicroStrategy Web, enroll in the *11.221 SDK for Customizing Analytical Applications* class to start your journey in MicroStrategy Web SDK.

Moderate to extensive coding

If you have some programming knowledge (or the desire to learn a minimum), more customization scenarios are within your reach. Examples include:

- Adding links or buttons to Report Services documents and interactive dossiers
- Adding or removing toolbar buttons
- Create custom workflows, such as automatic prompt answering
- Add new elements to the user interface

In this class, we explore some of those scenarios, with a focus on MicroStrategy Mobile, and introduce a little bit of code.

Extensive to exclusive coding

If coding is second nature to you, the sky is the limit. For example, you can:

- Enable new data sources
- Create a new analytical application
- Embed MicroStrategy applications
- Use MicroStrategy as a data provider

By using the documentation and samples provided, you can create deep changes in functionality or workflow in the architecture, or create your own custom applications to leverage the power of MicroStrategy. This level of customization is covered in advanced classes.

Now that you have a better understanding of the MicroStrategy Mobile SDK, install it on your local computer so you can begin customizing your application.

Exercise 2.1: Install the Mobile SDK

In this exercise, you locate, download, and install the MicroStrategy Mobile SDK for Android on your local computer.

Install the Android Mobile SDK

1 On your computer, open a browser and navigate to the following address:

https://community.microstrategy.com/s/products

A login page opens.

2 Perform the following depending on whether or not you already have a MicroStrategy login:

- If you already have a MicroStrategy login, enter your name and password.
- If you do not have a MicroStrategy login:

  a Click **Sign up**.

  b Enter the following information:

  – First and last name

  – Password: Use at least 8 characters including one special character: ! @ # $ % & / = ? _ . , : ; + ^ -

  – Email address

  – Company Name

  – Job Title

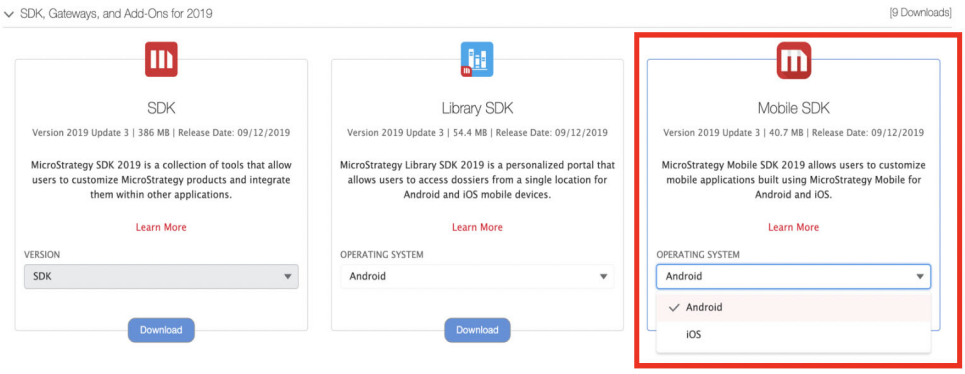
  – Telephone

  – Country

  – State/Province

  c Click **Create account**.

3 Under **Mobile SDK**, select **Android**, then click **Download**.



Your browser opens a new window with a license agreement.

4 Accept the license and click **Download**.

The file is automatically saved in the **Downloads** folder.

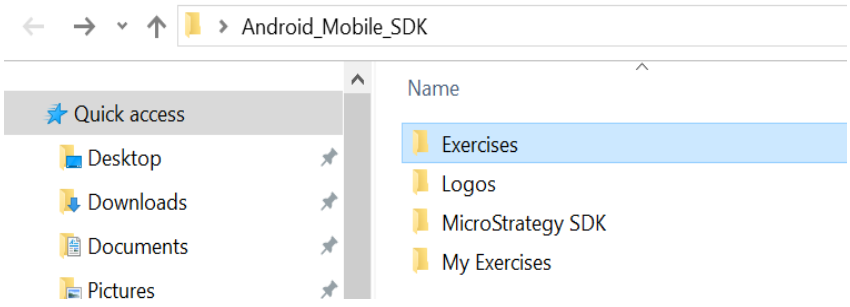
5 Open the **Downloads** folder. Double-click the archive you just downloaded.

6 Select all of the contents in the folder and copy them.

7 Open the **Android_Mobile_SDK/MicroStrategy SDK** folder on the desktop.

8 Paste the Android Mobile SDK files you copied in the MicroStrategy SDK folder.

Your Android Mobile SDK folder structure should look like the image below.



Activity 2.2 Quiz

Answer the following questions.

1 Name one typical use of the MicroStrategy Mobile SDK.

_____

2 Name the two components of a mobile deployment with MicroStrategy.

_____

_____

Summary

In this chapter, you were introduced to the MicroStrategy SDK and API family.

We then turned our focus on a specific component of the MicroStrategy SDK: the Mobile SDK. We saw that it allows you to work in the following areas:

- Customize mobile applications built using the MicroStrategy Mobile platform.
- Rebrand, preconfigure it, and customize your mobile app.
- Use and/or customize custom widgets for the MicroStrategy Mobile app to help you enhance the display of data.

We also discussed the balance you have to juggle between the level of customization you desire and the amount of programming skills needed to achieve the desired type of customization.

Then, you installed the MicroStrategy SDK on your computer to further prepare you to begin your own customization of the MicroStrategy Mobile app.

The next chapter, introduces the Android Studio development environment.

3

Introduction to Android Studio

In this class, you use Android Studio to customize, pre-configure, and rebrand your MicroStrategy Android mobile application. Android Studio is a powerful Integrated Development Environment (IDE) for Android app development. One of the many features of Android Studio is a unified environment where you can develop for various Android devices.

By the end of this chapter, you will be able to:

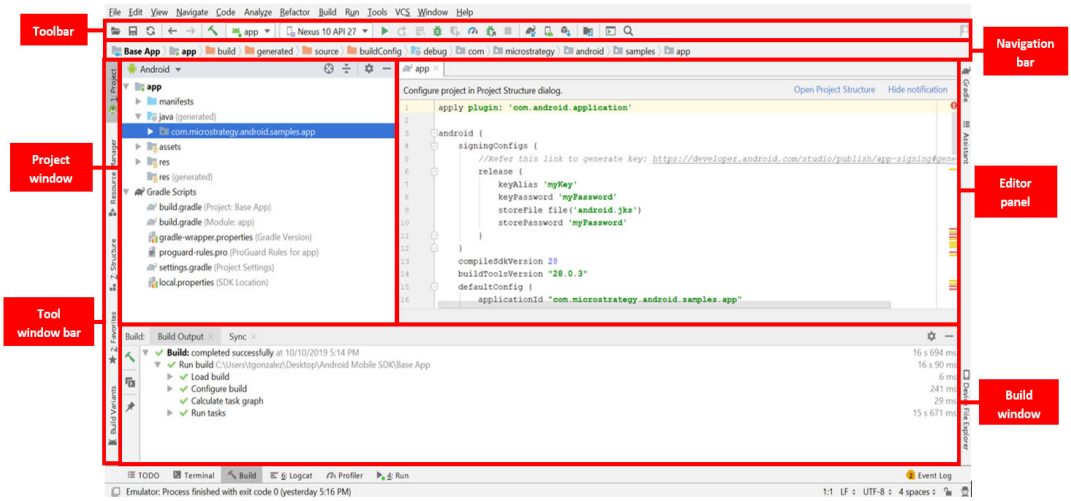
- Identify the main areas of the Android Studio environment.
- Download and install Android Studio IDE
- Create a simple app using Android Studio.
- Build and deploy the app to an emulator.

Android Studio overview

Android Studio is an IDE used for Android app development. It was created by IntelliJ and has a powerful code editor, as well as developer tools. Android Studio also offers features that enhance your productivity when building Android apps, such as a flexible Gradle-based build system, feature-rich emulators, and an image asset creator.

Gradle is an open-source build automation system used to automate the building, testing, and deployment of Android applications. Every Android project needs a gradle for generating an Android Package kit (apk) from the .java and .xml files in the project. APK is the file format Android uses to distribute and install apps.

An emulator is a virtual device that allows you to test your app on your computer. The emulators in Android Studio can simulate various Android phone, tablet, Wear OS, and Android TV devices. Although emulators are fine for development, they should not be the only testing ground for your application. With Android Studio, you can deploy your app to a physical Android device connected to your computer.



The image above highlights the main sections of the Android Studio interface. You can re-organize the main window to give yourself more screen space by moving or hiding the toolbars and tool windows. Once you have your layout the way that you want, it can be set as the default layout by selecting **Store Current Layout as Default** from the Window menu.

Navigation bar

The navigation bar provides a more compact view of your project structure visible in the Project window. You can use the navigation bar to open project files for editing.

Tool window bar

The toolbar window runs around the outside of the Android Studio window and contains buttons that allow you to expand or collapse the individual tool windows. Tool windows give you access to specific tasks like project management or version control.

Exercise 3.1: Install Android Studio

To customize the MicroStrategy Mobile app for Android, you need to install a development environment. Follow these steps to install Android Studio on your local computer.

Download Android Studio

1. On your computer, open a browser and navigate to the following address:

   https://developer.android.com/studio/index.html#downloads

2. Download the latest released version for Mac or Windows. Do not download a preview version for an upcoming operating system.

3. The Terms and Conditions window opens. Select **I have read and agree with the above terms and conditions**.

4. Click **Download.**

Set up Android Studio

The install process of Android Studio may vary with different versions. The installation is similar whether you run the setup on a PC or on a Mac.

1. Launch the install keeping the suggested installation path. Click **Next**.

2. Keep the suggested Start Menu Folder option. Clear the **Do not create shortcuts** check box. Click **Install**.

3. Click **Next**, then click **Finish** to start Android Studio.

4. When prompted, select **Do not import settings**, and click **OK**.

5. On the Welcome screen, click **Next**.

6. Select **Standard**, then click **Next**.

7. Select either the dark or light theme, and click **Next**.

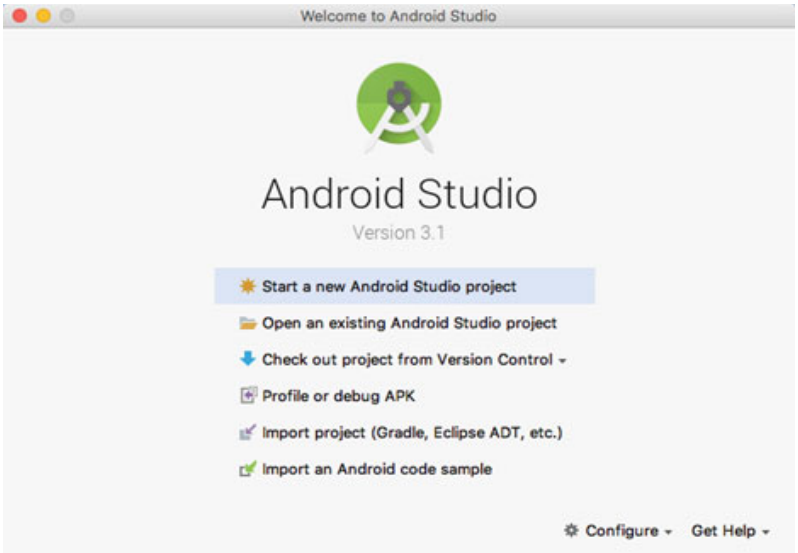
8. Click **Finish**.

   A component install begins. This can take some time.

9 If necessary, approve requests if they display.

10 Click **Finish**.

The Welcome to Android Studio screen opens, as illustrated below.



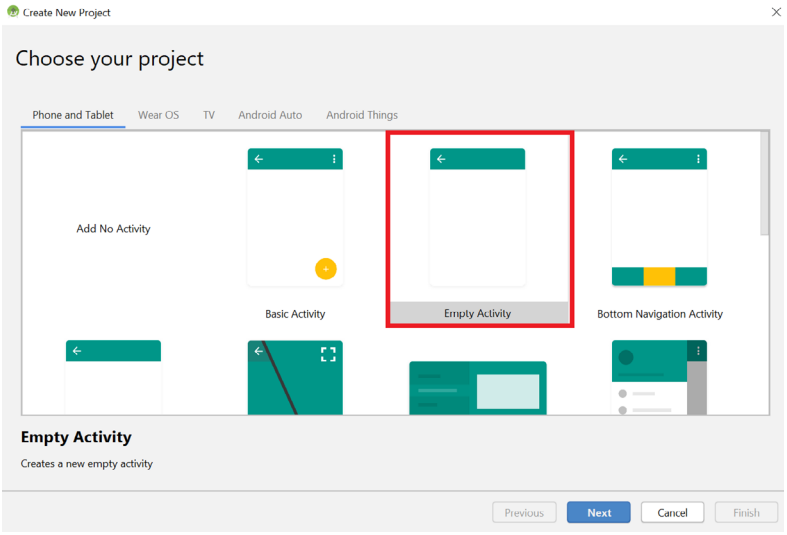
Exercise 3.2: Start a new project in Android Studio

This exercise shows you how to start and deploy a new Android Studio project to a virtual device. It also describes some of the files used in the project.

Start a new project

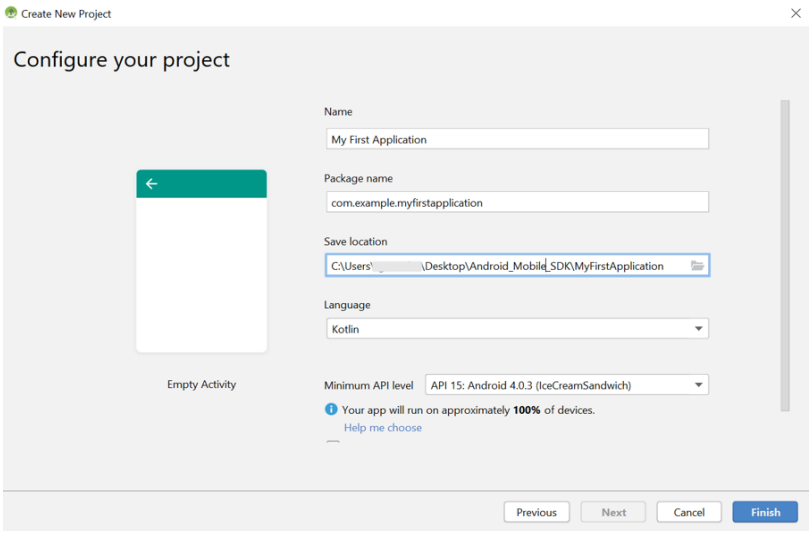
1 From the Welcome to Android Studio screen, select **Start a new Android Studio project**.

2 From the Choose your project window, choose **Empty Activity**. Click **Next**.



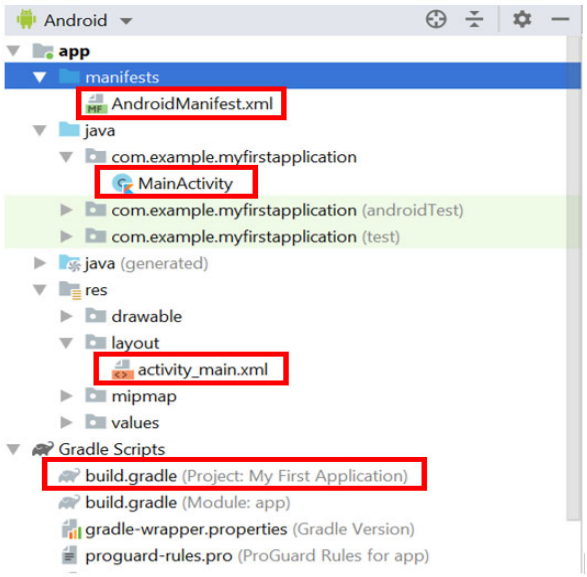
3 In the Configure your Project window, type **My First Application** in the name box. Notice the package name automatically updates to match the name of the application.

4 Browse to the **Android_Mobile_SDK/My Exercises** folder on your desktop and select it as the Save location.



5 Leave the defaults for the remaining options and click **Finish**. After some processing time, the Android Studio main window opens.

The Project window helps you navigate through your project and open files for editing. When you start a new project in Android Studio, it creates the necessary structure for all your files and makes them visible, as shown below.



- **app > manifests > AndroidManifest.xml:**The manifest file describes the fundamental characteristics of the app and defines each of its components.
- **app > java > com.example.myfirstapplication > MainActivity:** This folder is the entry point for your app. When you build and run your app, the system launches an instance of this activity and loads its layout.
- **app > res > layout > activity_main.xml**: This XML file defines the layout for the activity's user interface.
- **Gradle Scripts > build.gradle:** There are two files with this name, one for the project and one for the app module. Each module has its own build.gradle file and is used to control how the Gradle builds the app.

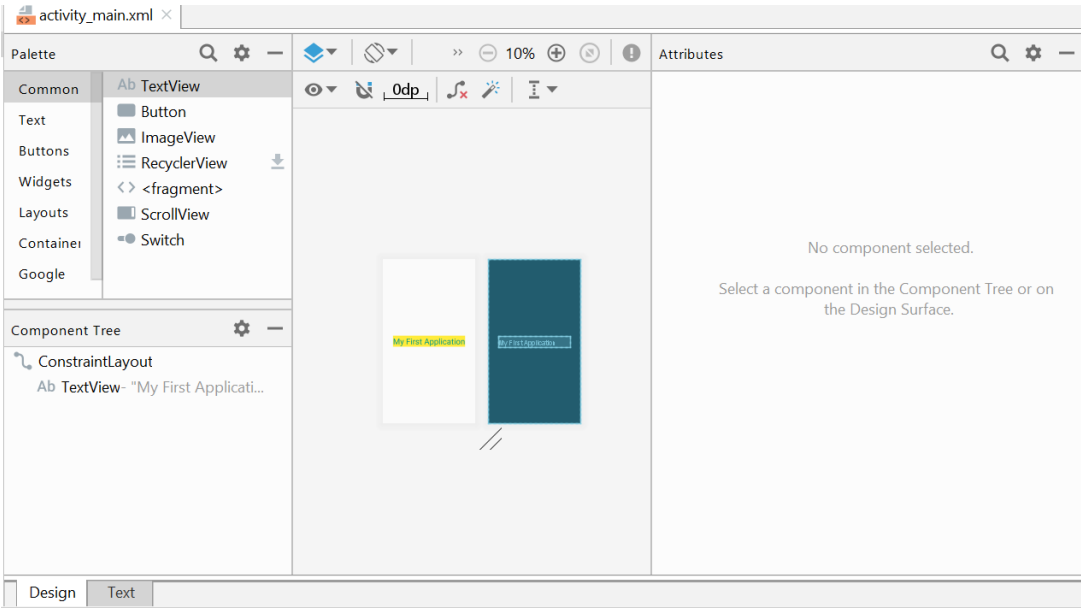
Exercise 3.3: Build and run the app

In this exercise, create a simple user interface (UI) and run your My First Application in an emulator.

Create the user interface

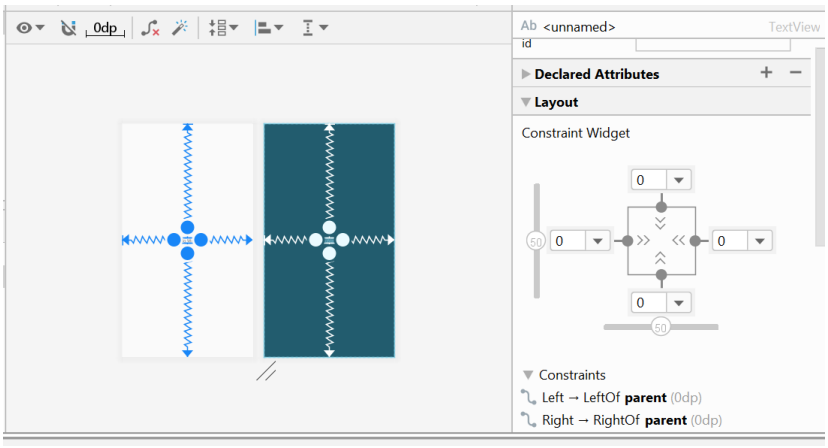
1 From the Project window, double-click the **activity_main.xml** file. The file opens in the editor panel displaying the Layout editor.

The editor window is where you create and modify code. The editor window changes based on the file type. For example, when viewing a layout file, the editor displays the Layout Editor as shown below.



2 From the Design tab, select the **Hello World** TextView that is on the layout.

The properties for this object are displayed in the Attribute panel on the right. The blue lines on each side of the textView defines its layout constraints.



3 Expand **Declared Attributes**, and change the text field to **My First Application.**

4 Scroll down on the Attributes panel and expand **All Attributes**.

5 Within All Attributes, change the following properties:

- **background**: Use the color picker next to background to select a color of your choice.
- **textColor**: Select a color that pairs well with your background color.
- **textSize**: Set the text size to **36sp**.

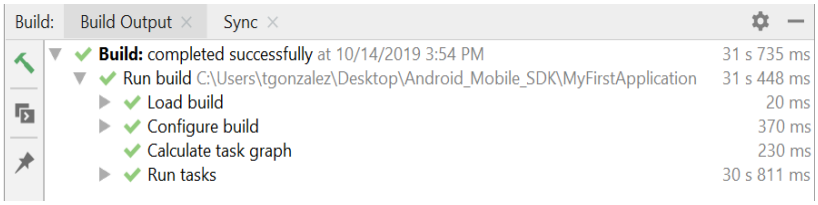
6 Save your changes by clicking **Save All** from the File menu.

Test the application

1 Click the **Make Project** icon on the toolbar to build the project.

The build window displays the tasks that Gradle executes to build the app. The executed Gradle tasks are displayed as a tree, where each node represents

either a build phase or a group of task dependencies. Once your build has completed, your Build window will look like the one below.



If you receive build-time or compile-time errors, inspect the tree and select an element to read the error output.

2 After compilation without error, click the arrow next to **No devices** and select **Open AVD Manager**.

The Select Deployment Target window opens. This is where you select the emulator to use to run the app. The Android Emulator simulates various Android phone, tablet, Wear OS, and Android TV devices on your computer.

3 Click **Create Virtual Device**. The new hardware window opens.

4 Click **Tablet** on the left under Category.

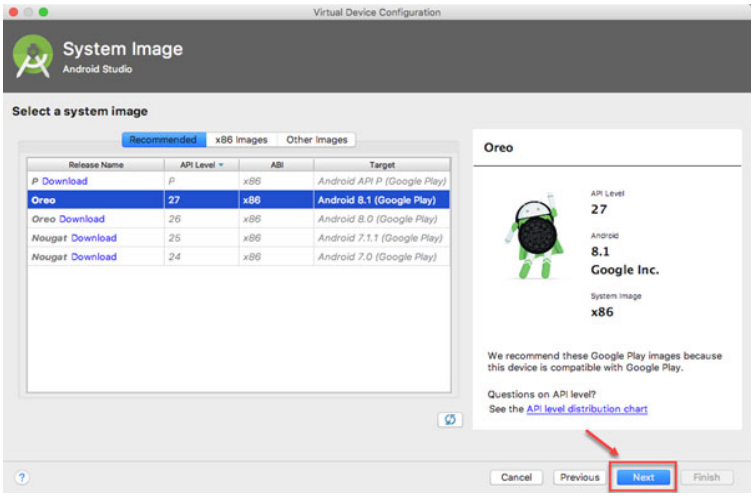
5 Select **Nexus 10**.

6 Click **Next**.

7 In the Android 8.1 row, click **Download**.

8 Follow the instructions onscreen to complete the download and configuration process of the device. Once completed, click **Finish**.

Your screen will look like the image below.



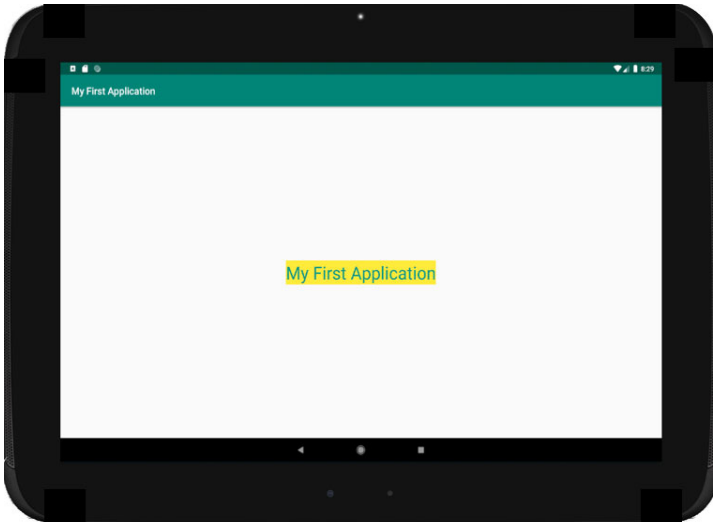
9 Click **Next**. You can now review the configuration.

10 Click **Finish**. The device selection screen is displayed.

11 Click **OK**.

12 Open your app in the emulator by clicking the **Run app**  icon on the toolbar.

Your app opens in the emulator similar to the image below. To test your app on a phone, you can follow the steps above to create an Android phone virtual device.



Congratulations! You just created your first app in Android Studio and deployed it to an emulator to test.

4

Mobile Architecture, Security and Configuration

Before deploying enterprise applications, Intelligent Enterprises identify and implement solutions to run mobile apps successfully without security or connectivity issues. Understanding the importance of each component of MicroStrategy's mobile architecture and its functionality is key to deploying a successful mobile application, as well as, learning how to take advantage of MicroStrategy's security and authentication functionality and features.

At the end of this chapter, you will be able to:

- Understand the MicroStrategy Mobile Architecture and its components.
- Explain different login credentials for server and project authentication.
- Create mobile configurations using the MicroStrategy Mobile Server Administrator page.
- Embed the configuration details directly in the app.

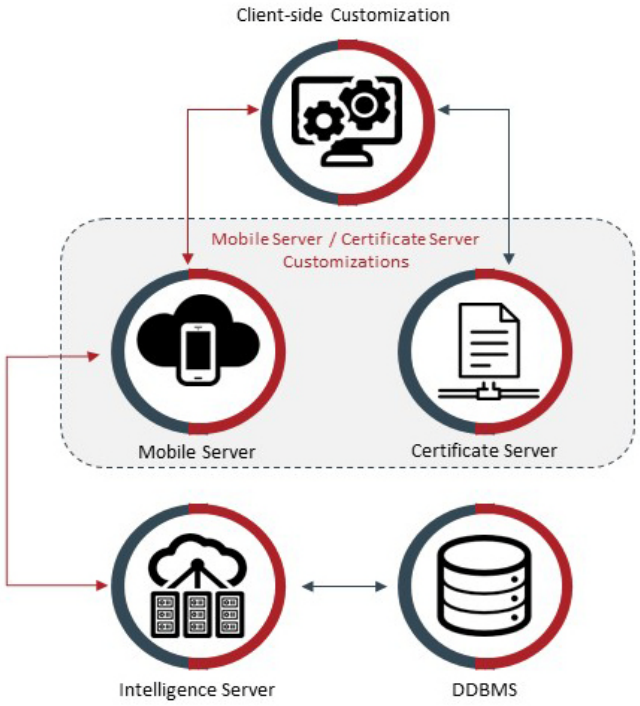
MicroStrategy Mobile architecture

There are two main components in a MicroStrategy Mobile deployment:

- The MicroStrategy Mobile client application installed on an iPhone, iPad, or Android device.
- MicroStrategy Mobile Server that maintains the configurations for devices and connects the devices to projects on the Intelligence Server.

The mobile architecture is comprised of the Mobile Server, Intelligence Server, Certificate Server, database, and client-side devices.

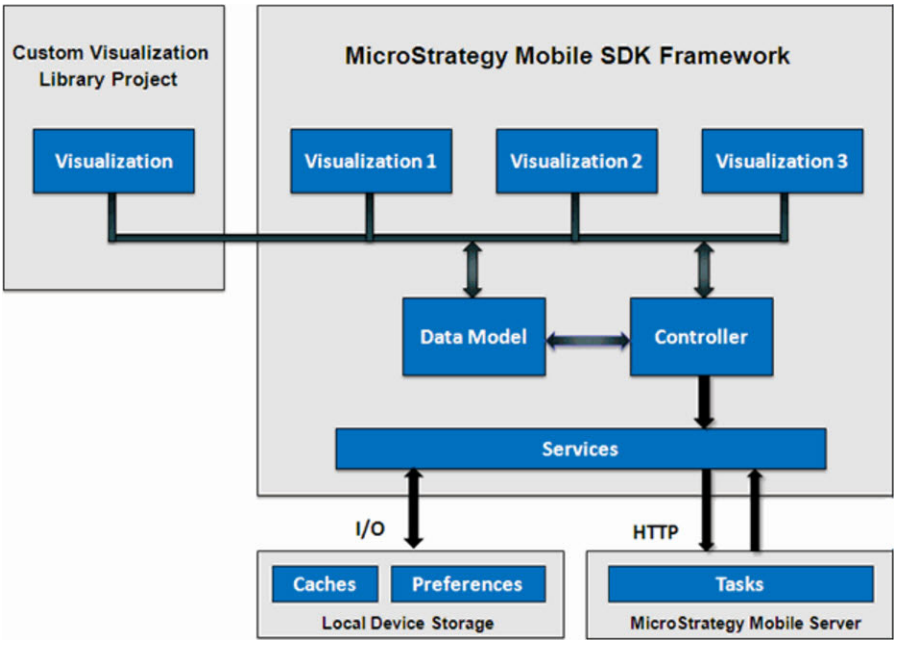
MicroStrategy Mobile Architecture



Client-side architecture

The architecture for MicroStrategy Mobile is based on the Model–View–Controller (MVC) paradigm, as shown in the diagram below. The services layer

communicates with information stored on the local device and communicates with Mobile Server using the MicroStrategy Web Task infrastructure.



Visualizations

A view object—called a visualization or widget—is what users see in the mobile application. The primary function of a view object is to display data from the data model object and to allow that data to be edited. View objects communicate with the controller about changes in the data model, both sending and receiving information about these changes.

Controller

The controller acts as an intermediary between the data model and the view, allowing them to remain independent of one another and channeling information in both directions. The controller sends changes from the view to the data model, and it also modifies the view based on changes to the data model. For example, user actions in the view layer that creates or modifies data are communicated through the controller object and results in the creation or updating of the data model object. When the data model object changes as a result of this communication, it notifies the controller object, which updates the appropriate view objects.

The controller sits on top of a services layer, which communicates with the Web Task infrastructure, the preferences, and report lists that are stored on the local

mobile device. The Web Task infrastructure is Web's Service-Oriented Architecture (SOA). Each Web task is a discrete chunk of functionality packaged as a web-accessible service that can be called as a service from anywhere using any protocol. These web-accessible services, and the packaged functions they represent, perform actions such as data retrieval, session management, caching, and so on. Examples of tasks include ReportDataService, which gets data for a report, and PromptsService, which answers prompts on a report.

Data model

The data model holds data and provides it to various views. The data model object also defines the logic and computation used to manipulate and process that data. Data model objects communicate with the Controller about changes in views, both sending and receiving information about these changes.

The data model holds data and provides it to various views (for example, widgets displayed in a mobile app). MSIModelData represents the entire data for one report. It provides the list of all MSIHeader objects in the report, as well as methods to access report data row-by-row. Each row in the data is a collection of cells, and each cell is a MSIHeaderValue object.

Security and authentication

As a component of the MicroStrategy platform, the mobile app has access to all the data and functionality the platform offers. Like other platform components, Mobile must be authenticated to the environment, at each level that is configured in a given environment, to gain access to MicroStrategy data and functionality. This can include certificate server authentication, server authentication, and project authentication.

The MicroStrategy SDK lets you customize authentication on both the server side and client side. For example, you can customize the look and feel of the authentication prompts (login screens) on mobile devices, as well as the workflow of the authentication process in the mobile app. You can also customize the credentials validation process on the Mobile Server by creating a custom mobile login task.

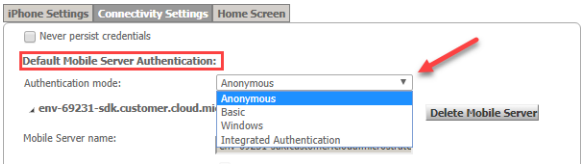
Types of authentication

The mobile app and custom app that embed the mobile framework must log on to MicroStrategy and be authenticated before they can access MicroStrategy data and platform functionality.

The app performs the following types of authentication:

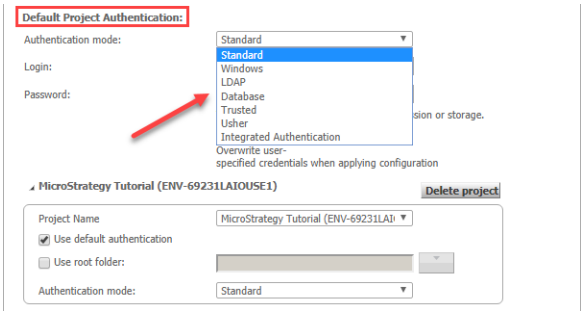
- **Server authentication**

  The server authentication process is handled by the container (for example, Tomcat or IIS). MicroStrategy does not take part in this process and none of the SDK APIs are used. Authentication choices include Windows, Basic, and Anonymous, which requires no credentials.



- **Project authentication**

  The project authentication process is handled by the Intelligence Server. The authentication authority used to verify that the login credentials are valid depends on how user credentials are stored within a company's environment. Companies typically store user credentials in a database, a Windows domain, or a directory server. Depending on the Intelligence Server configuration, authentication choices include Standard, Windows, LDAP, and Trusted.

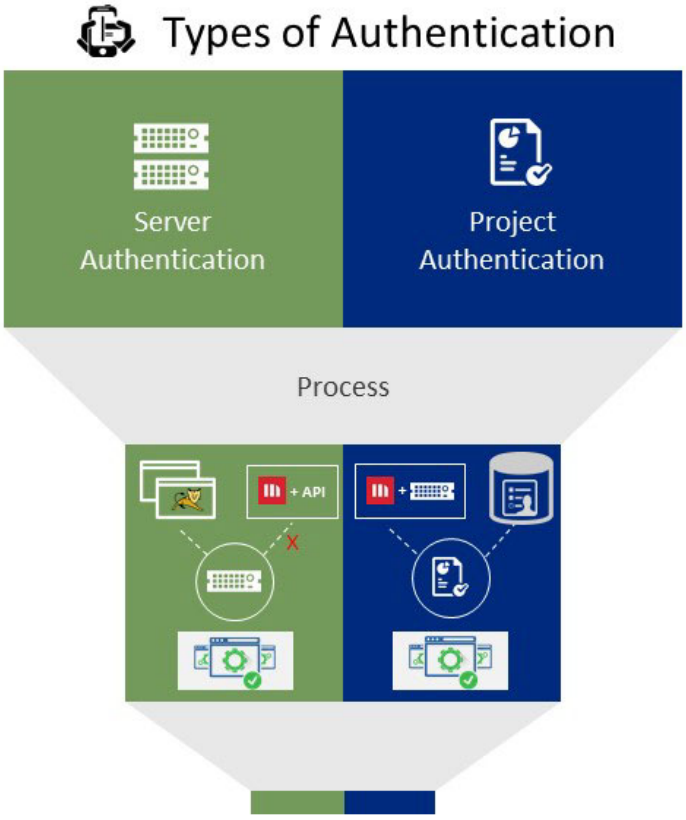


MicroStrategy Mobile performs project authentication using the mobile login task. This task is responsible for obtaining the credentials passed by the mobile app and sending them to the Intelligence Server for validation. If this process is successful, the Intelligence Server creates a session and passes the corresponding session ID back to the client.

Tasks are discussed in detail in the Web SDK section of the MSDL under **Tasks and Service-Oriented Architecture**.

You may need to change the authentication workflow of your mobile app. For example, you may need to map one user ID to another user ID, create users on the fly, validate credentials against another server, or validate additional input parameters. You can achieve custom workflows like these by creating a custom

mobile login task that includes the code to handle the necessary authentication scenarios.



Exercise 4.1: Set up the local Mobile Server

In the MicroStrategy Cloud environment you received for this class, a MicroStrategy Mobile Server is already set up (on the Linux server) to provide content to different devices. It takes a few steps to use what is in place to associate the MicroStrategy Mobile app and this server. However, during your setup, you installed a local MicroStrategy Mobile Server on the Windows server. That one is blank and waiting for you to prepare it to be used in your app.

You complete two Mobile configurations in this exercise, for an Android phone and the other for an Android tablet. Complete both, even if you brought a single device to class.

Create an Android phone mobile configuration

1 Connect to the remote machine, using the credentials provided in the Welcome to MicroStrategy Cloud email.

2 On the desktop, right-click the **RestartTomcat.bat** file and select **Run as administrator**. In a security alert window, click **Yes**.

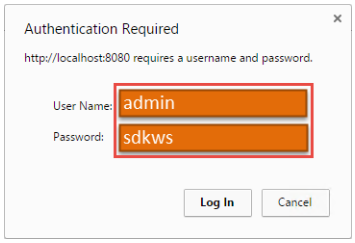
This restarts the Tomcat server.

3 Open a browser, and navigate to the following address:

http://localhost:8080/MicroStrategyMobile/servlet/mstrWebAdmin

4 In the security pop-up, login with the following credentials:

- user name: **admin**
- password: **sdkws**

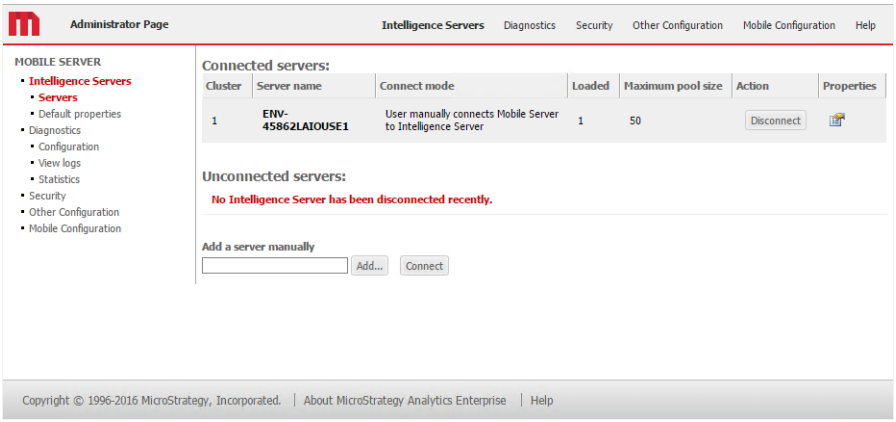


The MicroStrategy Mobile Server Administrator page is displayed.

5 In the **Add a server manually** box, enter ENV-**XXXX**LAIOUSE1, which is the name of the Intelligence Server, and where **XXXX** is your environment number.

6 Click **Connect**.

Your screen should resemble the following image:



7 In the left pane, under Mobile Server, select **Mobile Configuration**, then click **Define New Configuration**, as shown below.

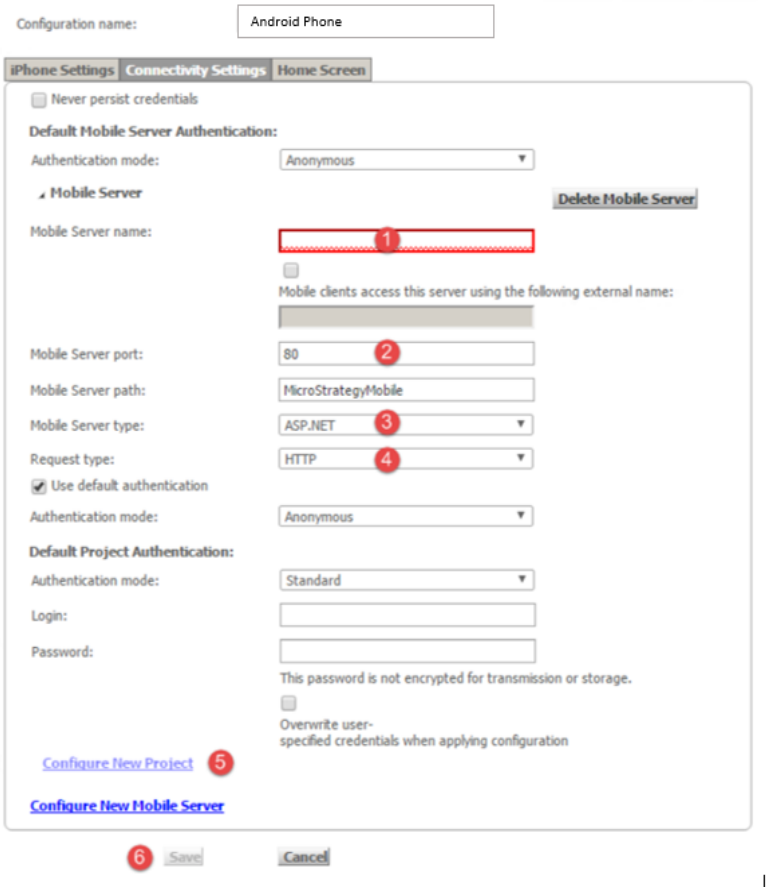
8 In the pop-up window, ensure that the **Android Phone** device is selected, and click **OK** as shown below.

The Mobile Configuration screen opens so you can modify the details for the new configuration.

9 Select the **Connectivity Settings** tab.

10 In the Configuration name text box, type **Android Phone**.

11 Click **Configure New Mobile Server**. The screen expands and should look like the picture below.



12 Under Default Mobile Server Authentication, provide all the necessary information as described below:

- **Mobile Server name**, enter the following server name:

  env-XXXX-sdk.customer.cloud.microstrategy.com

  where **XXXX** is the environment number.

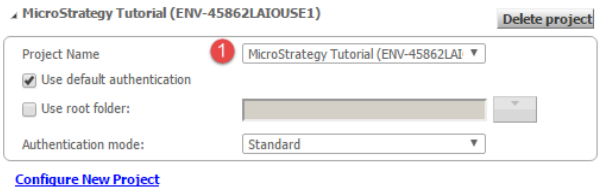
- **Mobile Server port**, enter **8080**.

  This number is the port of the local Tomcat web server running on the Windows server. The port number can vary if you are using a different web server or decide to use a non-standard port for security, as well as if you are using https. For example, the mobile configuration on the Linux server uses https and port 443.

- From the **Mobile Server type** drop-down list, select **J2EE**.

- From the **Request Type** drop-down list, select **HTTP**.

13 Under Default Project Authentication, click **Configure New Project**. The bottom of the page expands as shown below.



The page uses the information you entered to connect to the server to retrieve a list of projects. If a successful connection is established, this is almost immediate. If you see a turning wheel, it can mean the requested server is not available or some details entered are erroneous. If this occurs, click **Delete Mobile Server** and start from scratch with the connectivity information.

14 In the **Project Name** drop-down list, select **MicroStrategy Tutorial**.

Tutorial allows you to verify the connectivity while learning how to complete the process.

15 Click **Save**.

The main Mobile Configuration page is displayed, as shown below.

Your local Mobile Server now has its Android Phone configuration. You need to create another one for an Android tablet.

Create an Android tablet mobile configuration

1 Click **Define New Configuration.** In the pop-up window, ensure that the **Android Tablet** is selected, and click **OK**.

2 Select the **Connectivity Settings** tab.

3 In the Configuration name textbook, type **Android Tablet.**

4 Click **Configure New Mobile Server**.

5 Under Default Mobile Server Authentication, provide all the necessary information as described below:

- **Mobile Server name**, enter the following server name:

  env-XXXX-sdk.customer.cloud.microstrategy.com

  where **XXXX** is the environment number.

- **Mobile Server port**, enter **8080**.

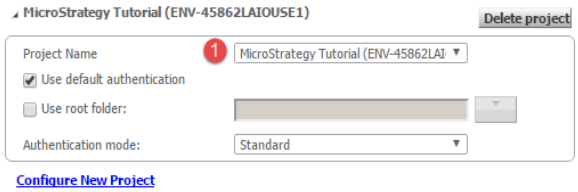
  This number is the port of the local Tomcat web server running on the Windows server. The port number may vary depending on if you are using a different web server, or decide to use a non-standard port for security, as well as if you are using https. For example, the mobile configuration done on the Linux server uses https and port 443.

- From the **Mobile Server Type** drop-down, select **J2EE**.

- From the **Request Type** drop-down list, select **HTTP**.

6 Under Default Project Authentication, click **Configure New Project**.

The bottom of the page expands as shown below.

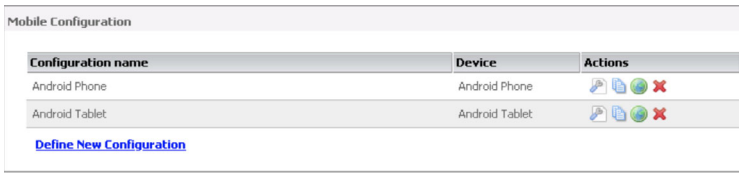


The page has uses the information you entered to connect to the server to retrieve a list of projects. If a successful connection is established, this is almost immediate. If you see a turning wheel appear, it can mean the requested server is not available or some details entered are erroneous. If this occurs, click **Delete Mobile Server** and start from scratch with the connectivity information.

7 In the **Project Name**, select **MicroStrategy Tutorial.**

8 Click **Save**.

The main Mobile Configuration page is displayed, as shown below.



Congratulations! The local Mobile Server now has configurations for the devices we want.

Pre-configuring the mobile app

In the previous exercise, you defined configuration profiles for specific devices. However, at this point there is a disconnect in the setup in the mobile application. On one hand, you have the Mobile Server with information on connectivity to data. On the other hand, you have a mobile app that points to the MicroStrategy demo data.

So how do we bridge the communication gap? How do we tell the mobile app to "point" to, or be aware of, our Mobile Server for content?

To resolve this situation, we can:

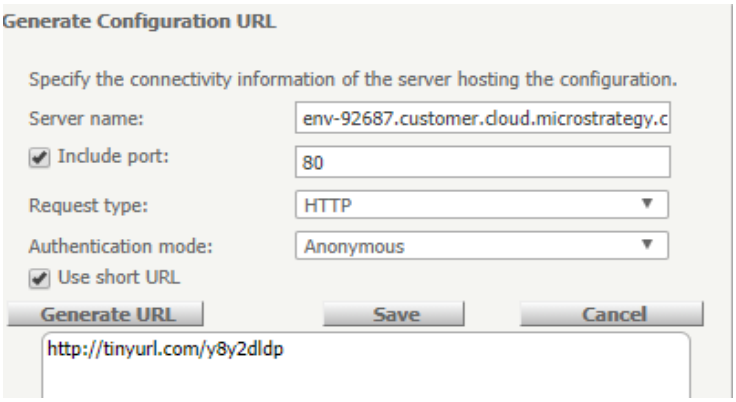
- Generate a specific URL link and distribute it to users in an email or on a web page.
- Embed the configuration file's content in the mobile app.

Tap to configure: Configuring using a URL

To make a mobile app aware of the MicroStrategy Mobile Server and its specific content, the developer or system administrator can rely on the Mobile Server to generate a URL that triggers a dialog between the mobile app and the Mobile Server. In the end, the mobile app "configures itself" and once the process is complete, the mobile app accesses the Mobile Server content and remembers this configuration at every launch.

By default, tapping on a configuration link using an Android device performs a search rather than automatically configuring the MicroStrategy Mobile for

Android application. When generating the URL for Android devices, you must select **Use short URL** to configure an Android app.



This configuration method can also be used to update an existing mobile application. This avoids developers having to embed a new configuration file and force an update of the mobile application.

Pros and cons

The following are pros and cons for using the tap to configure method:

Pros

- You do not have to force users to update their app.
- Simple to do; no programming skills involved.
- Quickly done and distributed.
- Simple to revert to previous settings, using the previous URL.

Cons

- Harder to control the change to all users, as you do not control the users clicking the link or following the URL sent by email.
- Less secure, as the link can be read.

Pre-configured app: Embedding the configuration in the app

The second method of configuring the mobile app removes any participation from the end user as the information becomes part of the mobile application

users install. When a mobile configuration is created, the settings are saved in an xml file named mobileConfig-<Identifier>.xml, where <Identifier> is a Global Unique Identifier (GUID) composed of 30 hexadecimal characters in groups of 8-4-4-4-12 values. Below is an example of an actual configuration file name:

mobileConfig-902d5e49-c11a-44b6-aa72-109fa83b5858.xml

For Android, generate the appropriate JSON and text files through the getMobileConfiguration task to add the configuration to the MicroStrategy Mobile app. This way, the app uses the defined configuration you created, and not the default one.

Pros and cons

The following are pros and cons for using the pre-configured app method:

Pros

- Every user receives the change as the app is updated.
- Easier to force users to update.
- Simpler support as everyone is on the same code and configuration.
- More secure as the configuration is harder to locate.

Cons

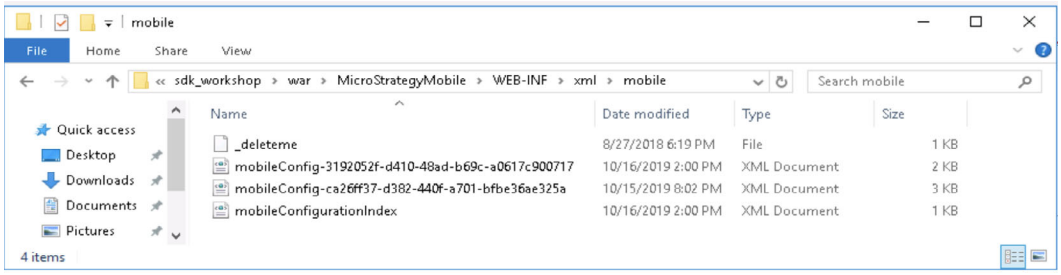
- All users are using the same configuration, so it is hard to make on the fly adjustments for special circumstances
- Harder to do, as programming skills are involved. Need to plan the resources for the change.
- Longer to implement and distribute.
- Harder to revert to previous setting, need to re-deploy the entire app.

Exercise 4.2: Create the generated URL and JSON objects

In this exercise, retrieve the mobile configurations you created in the previous exercise and use them to create the generated URL and JSON objects needed to pre-configure the app.

Retrieve the mobile configuration IDs

1 On the remote desktop, navigate to the **sdk_workshop/war/MicroStrategyMobile/WEB-INF/xml/mobile** folder to locate the XML configuration files created in the previous exercise.



2 Open both mobileConfig XML files by right-clicking each file and selecting **Edit with Notepad++**.

3 Copy only the **cid** (configurationID) value from each file and paste it in a text file. Make sure to label the values for the correct device as shown below.

Make the getMobileConfiguration task available

By default, only public tasks are visible in and accessible from the Task Administrator application. The getMobileConfiguration task is private. Follow

these steps to make this task visible so you can access the task to add the configuration to your app.

1 Navigate to the **sdk_workshop/war/MicroStrategyMobile/WEB-INF/xml** folder.

2 Locate the **taskProcessorControllerConfig.xml** file.

3 Right-click the file and select **Edit with Notepad++**.

4 Locate the line that defines the getMobileConfiguration task as private:

```
<taskProcessorController>
...
<privateTasks show="false">
...
<privateTask taskID="getMobileConfiguration" />
```

5 Rewrite the line as shown below:

```
<!--<privateTask taskID="getMobileConfiguration" />-->
```

This comments the task out, so it is not used, but leaves it in place in case it needs to be changed back.

6 Save the **taskProcessorControllerConfig.xml** file. Click **Yes** to launch Notepad++ in Administrator mode and save again.

7 Right-click the **RestartTomcat.bat** file and select **Run as administrator**. In a security alert window, click **Yes**.

Tomcat must restart so the changes are integrated.

Create a JSON object based on the configuration URL

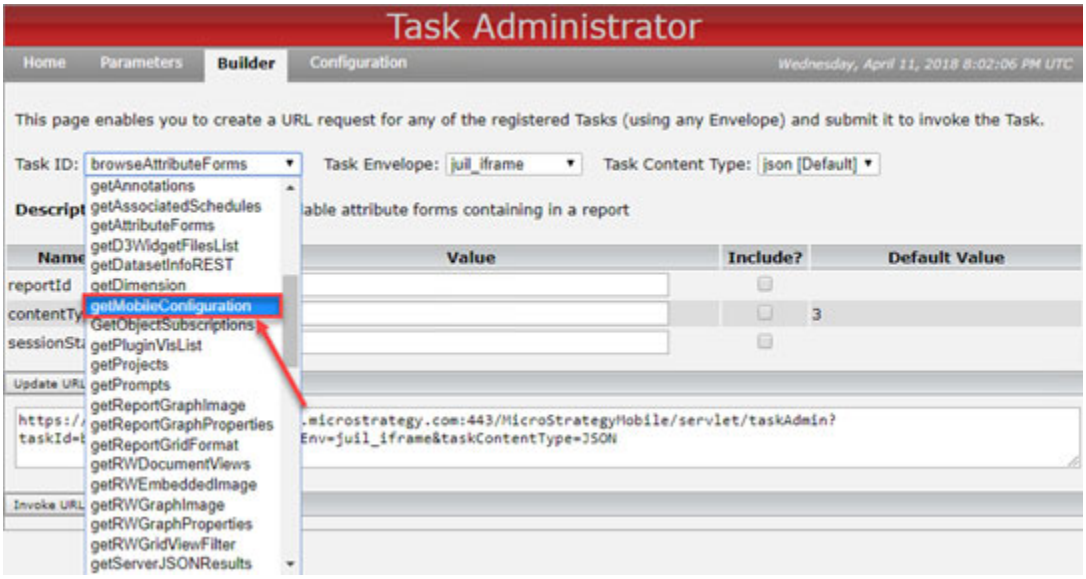
1 Open a Chrome browser on your remote desktop, and access the Task Administrator application using the URL below.

http://localhost:8080/MicroStrategyMobile/servlet/taskAdmin

2 If prompted, login with the following credentials:

- user name: **admin**
- password: **sdkws**

3 On the **Builder** tab, select **getMobileConfiguration** from the **Task ID** drop-down list as shown below.



4 Select **xhr** from the **Task Envelope** drop-down list. When a task is invoked, the caller must specify an envelope. This is a predefined form that defines how the contents generated by the task are packaged and returned to the caller. When you use an xhr envelope, a default content type cannot be inferred. You must specify the desired content type (JSON or XML).

5 Select **JSON** from the **Task Content Type** drop-down list.

6 In the **Value** box for **configurationID**, type the cid value for **phone** that you saved in your text file. Select the **Include?** check box next to it.

7 Enter **flatten** in the **Value** box for **blockTransform**. Select the **Include?** check box next to it.

8 Enter **1** in the **Value** box for **blockVersion**. Select the **Include?** check box next to it.

9 Click **Update URL** to generate the URL. The generated URL is displayed in the text box below the button as shown below.

10 Copy the generated URL and paste it under the phone configuration ID in your text file.

11 Click **Invoke URL** to generate the Task Response. The response is generated as a JSON object in the text box below the button.

12 Copy the URL JSON object and paste it below the generated URL in your text file.

13 Repeat steps 6 - 12 above to create the generated URL and JSON object for the tablet.

14 Paste both the generated URL and JSON object in your text file below the tablet configuration ID.

15 Save your text file.

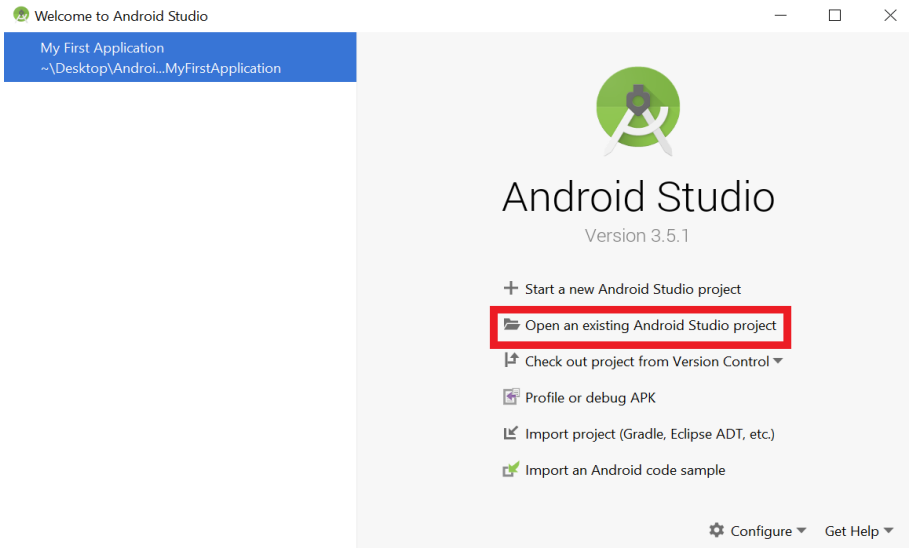
Your file should look similar to the one shown below.

Exercise 4.3: Embed the Mobile Configuration in the app

Embedding the mobile configurations directly within the app makes the implementation and distribution process easier. In this exercise, you embed the generated URL and JSON objects created in the previous exercise in the mobile app.

Build the Base App

1. On your local computer, open the **Android_Mobile_SDK folder**.
2. Copy the **MicroStrategy SDK** folder and paste it inside the **My Exercises** folder.
3. Rename the new folder **BaseApp**.
4. Open Android Studio and close any open projects.
5. From the Welcome to Android Studio screen, select **Open an existing Android Studio project**.



6. Navigate to **Desktop/Android_Mobile_SDK/My Exercises/Base App** and click **OK**.

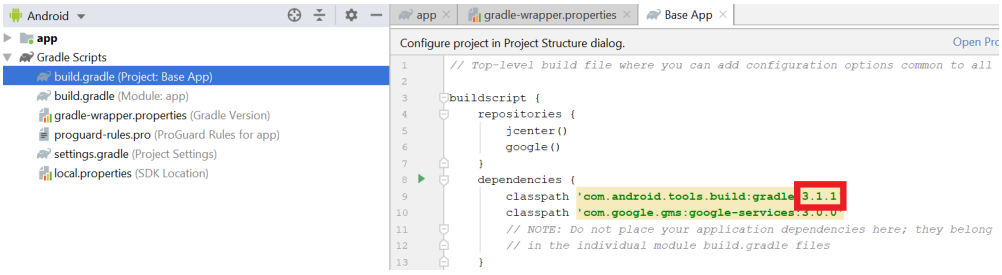
   The out-of-the-box MicroStrategy mobile app opens in Android Studio.

Best Practice

As a best practice, you should make sure you have a working build of the app before starting customizations like pre-configuring or re-branding.

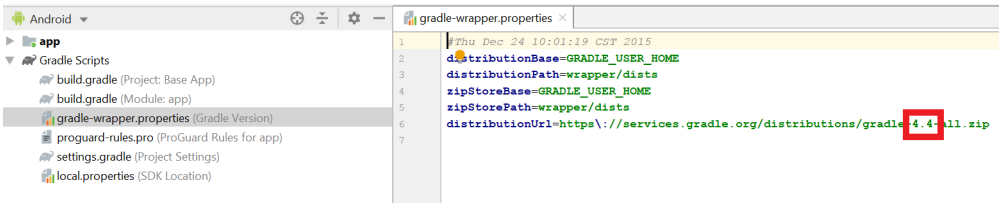
Build the Base app

7 In the Project window, expand **Gradle Scripts** and double-click **build.gradle (Project: Base App)**. The file opens in the Editor window.



8 Change 3.1.1 to **3.2.1**.

9 Double-click **gradle-wrapper.properties**.



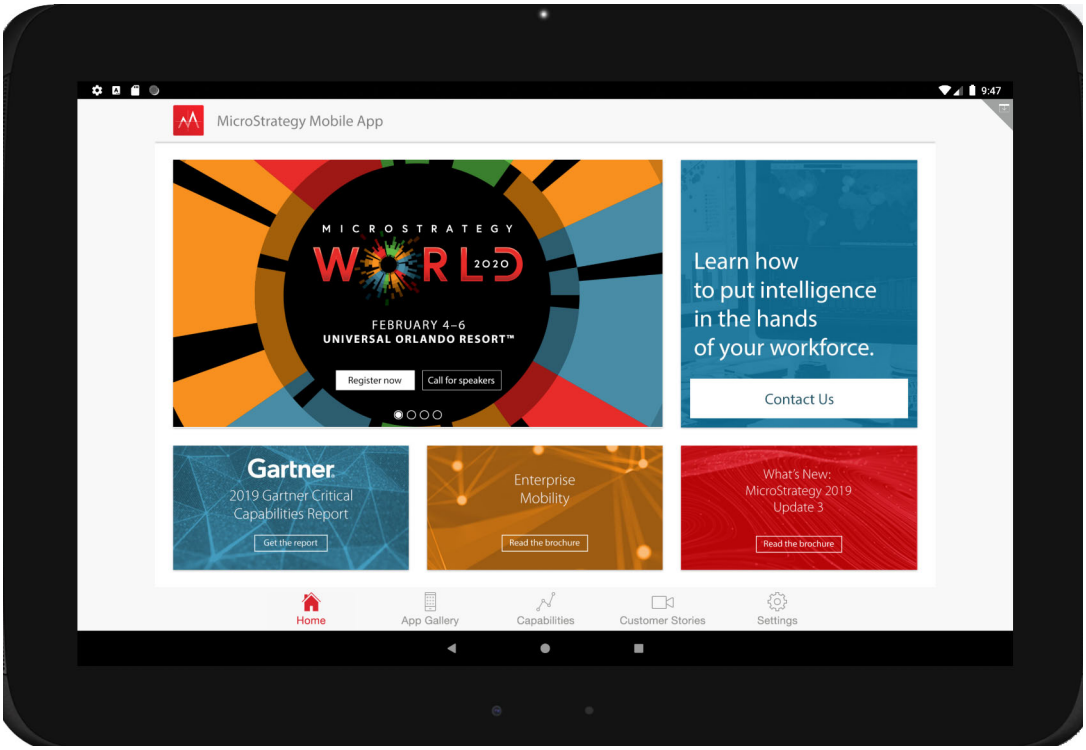
10 Change 4.4 to **4.6**.

11 From the **File** menu, click **Save All**.

12 Click the **Make project** icon on the toolbar to build the app.

13 After compilation without error, open your app in the emulator, by clicking the **Run app** icon in the toolbar.

After syncing with the Base App, the MicroStrategy Mobile app is deployed on the emulator as shown below.



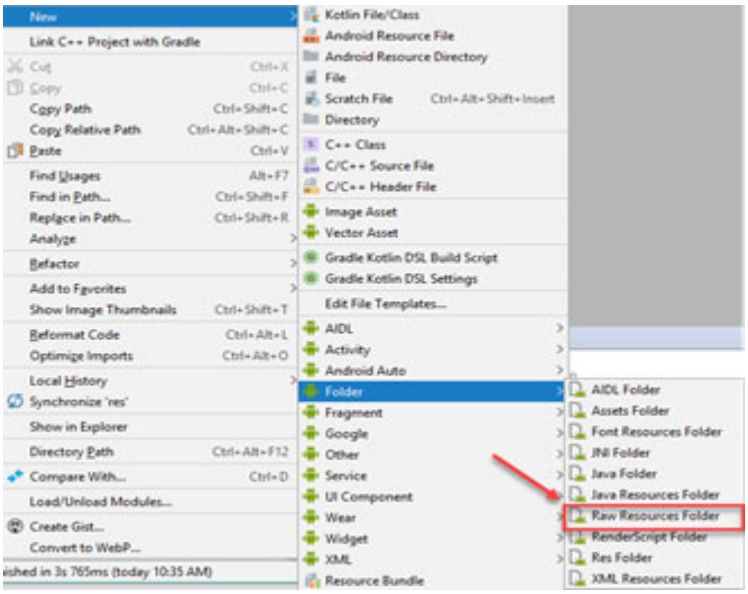
Your development environment setup is valid and the base MicroStrategy Mobile app is running, but is currently pointing to the MicroStrategy Demo server.

Embed the generate URL and JSON object in the app

Follow the steps below to pre-configure the base app to point to your MicroStrategy Cloud environment.

1 In the Android Studio Project window, expand the **app** folder.

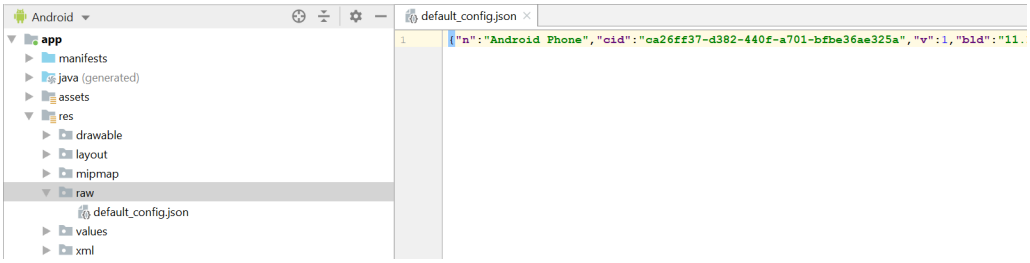
2 Right-click the **res** folder and create a new folder called **Raw Resources** as shown below.



3 Click **Finish** in the folder window. The Project window now displays the raw folder.

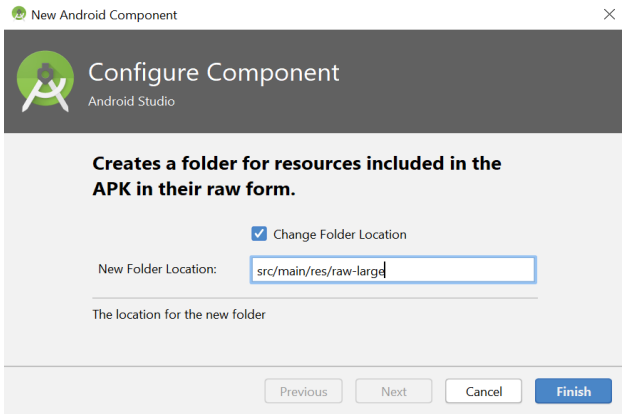
4 Right-click the **raw** folder, and create a file named **default_config.json**.

5 Paste the **phone JSON object** from the text file created previously.



6 Save the file.

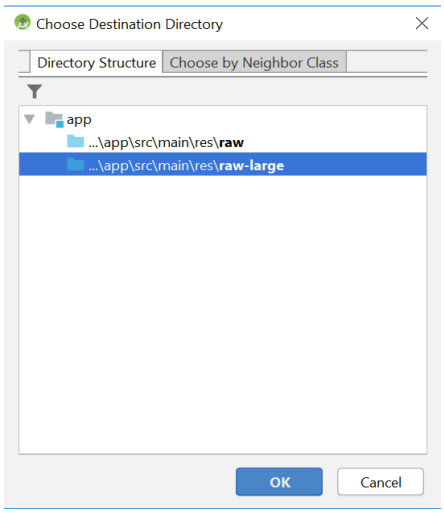
7 Right-click the **res** folder and create a second Raw Resources folder named **raw-large**. To name the folder, select **Change Folder Location** in the New Android Component window.



The raw-large folder contains the project configuration file for tablets. The raw-large folder is only visible in File Explorer, not in Android Studio.

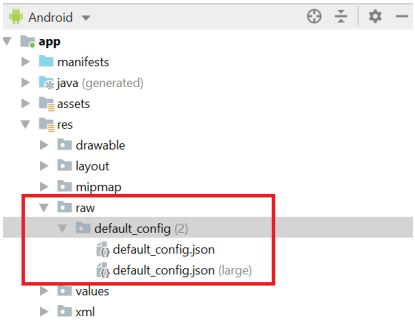
8 Right-click the **res** folder and create a new file.

9 Select the **raw-large** folder and name the file **default_config.json**.



10 Paste the **tablet JSON object** from the text file created previously. Save the file.

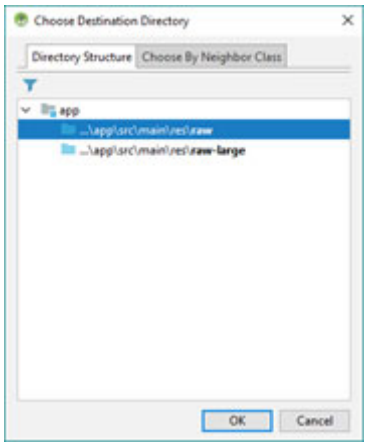
After completing these steps, your Project window should look like the image below.



Replace the default configuration URL used for configuration syncing

When you use the MicroStrategy Mobile base project to build your own custom mobile application, you need to change the default configuration URL stored in the base project. This configuration URL is used by the Mobile Server to determine whether a configuration has been updated. If you do not replace the default URL with your own URL, the automatic configuration syncing will not work correctly on the device.

1 Right-click the **raw** folder and create a text file named **default_config_url.txt**. When adding the file, choose the destination folder as shown below.



2 Paste the **phone generated URL** saved in your text file created previously.

3 Right-click the **raw** folder and create a second text file named **default_config_url.txt**.

4 Select the **raw-large** folder from the **Choose Destination Directory** window.

5 Paste the **tablet generated URL** saved in your text file created previously.

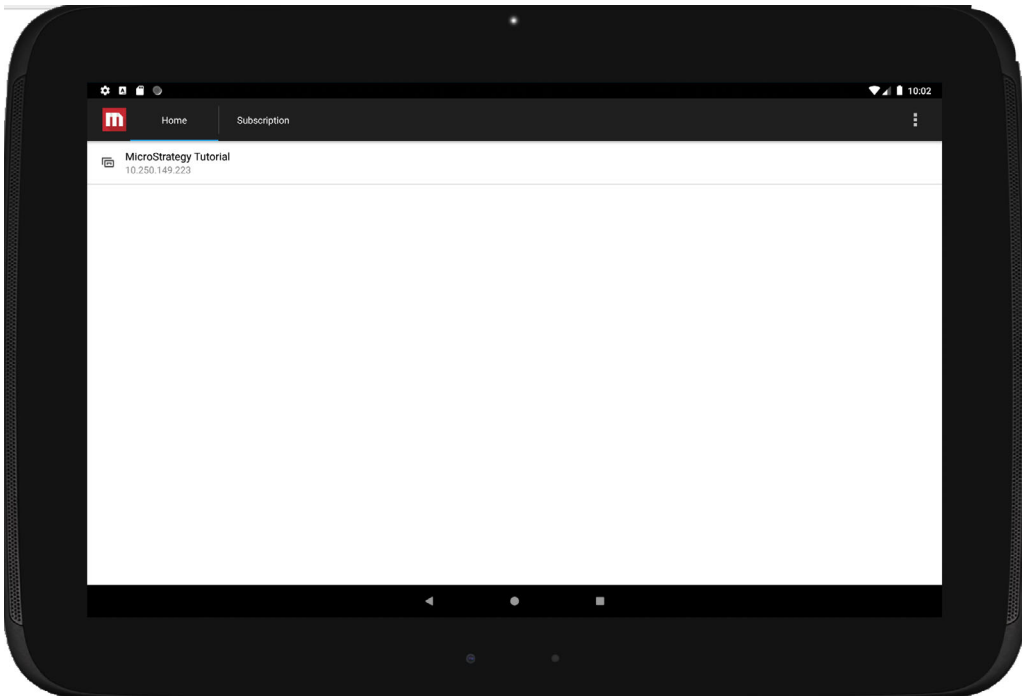
6 From the **File** menu, select **Save All**.

Test your modification

1 Click the **Make project** icon to build the app.

2 Open your app in the emulator using the **Run app** icon.

Your app is now pre-configured to your MicroStrategy Cloud environment and opens like the image below.



3 Select **MicroStrategy Tutorial** and enter the credentials you received in your MicroStrategy Cloud email.

The app opens to the Shared Reports folder.

Adding credentials to the configuration

Users have to enter their credentials inside the mobile application once. After that, the application can remember them. There are parameters to force users to enter their credentials again, for example, if the app moved to the background for

a long period of time. Per most companies' security policies, it is not a recommended practice to embed credentials in a Mobile configuration.

However, during training, as the password for MicroStrategy Cloud is robust and complex, we embed the credentials to save time for each launch of the application.

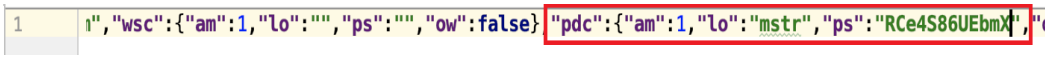
Exercise 4.4: Embed credentials in mobile configuration

To simplify the login process in this class, embed the credentials in the mobile app so that anytime you deploy it in the simulator, you do not have to enter the password.

Embed credentials in default_config file

Embed the credentials for the tablet. The same steps can be followed to embed the credentials for the phone as well.

1. From the Project window in Android Studio, expand the **app/res/raw** folders.
2. Double-click the **default_config(large)** file. The file opens in the Editor panel.
3. Locate the following section of code:



4. Enter your username between the quotes next to "lo":.
5. Enter your password between the quotes next to "ps":.
6. Save the file.
7. Build and deploy your app in the emulator.

Congratulations! Your app is now pre-configured to your MicroStrategy environment and no longer asks for credentials every time you deploy it to the emulator.

5

Customize Mobile Configurations, Icons, and Splash Screen

In the previous chapters, you were introduced to the MicroStrategy Mobile SDK, Android Studio, and the MicroStrategy Mobile application. This chapter focuses on one of the typical usages of the Mobile SDK: re-branding the Mobile app to reflect your company's identity. Re-branding seems like a simple affair, but in organizations, it can trigger numerous discussions on identity, image, message, and even purpose.

Follow the lead of the Services Architect and the Mobile Architect. Their role is to put in place standards, guidelines, and best practices for customizations and re-branding of the mobile application.

In this chapter, you re-brand the mobile app by applying the following customizations:

- Change the application ID to add a unique identifier for the app.
- Change the name of the app to match corporate style guides and better identify the app for users.
- Generate a new icon set at appropriate resolutions using the Image Asset Studio.

- Change the application icon so users' first view of the app is based on the enterprise style, typically using a corporate logo.
- Modify the copyright according to the enterprise's copyright information.
- Adjust the version number so designers and users are aware of what version number they are using.
- Customize the splash screen to match corporate color schemes.

Customizing MicroStrategy Mobile

For many organizations, re-branding the MicroStrategy Mobile application is their core use of the Mobile SDK, allowing them to enjoy the benefits the MicroStrategy platform offers. For the remainder of this course, your focus is re-branding and building the mobile application for the fictitious company, SportsEQ.

Meet SportsEQ

SportsEQ is a leading provider of sports equipment for retailers around the world. Currently, SportsEQ is in the process of becoming an Intelligent Enterprise and needs the ability to provide each member of their sales team with access to the latest data, even when in the field. Since SportsEQ already adopted MicroStrategy as their Business Intelligence platform, they are able to build and deploy a custom mobile application that fully integrates with their current environment. This means that, they are able to use existing dossiers and documents to create their mobile application.

Re-brand the mobile app SportsEQ wants to deploy by starting with the Base app you pre-configured in the previous chapter. Each exercise in this chapter covers

one aspect available for re-branding the MicroStrategy Mobile app. Once these concepts are clearer, you become empowered with the tools and knowledge to do the same in your own customization of the MicroStrategy Mobile app.

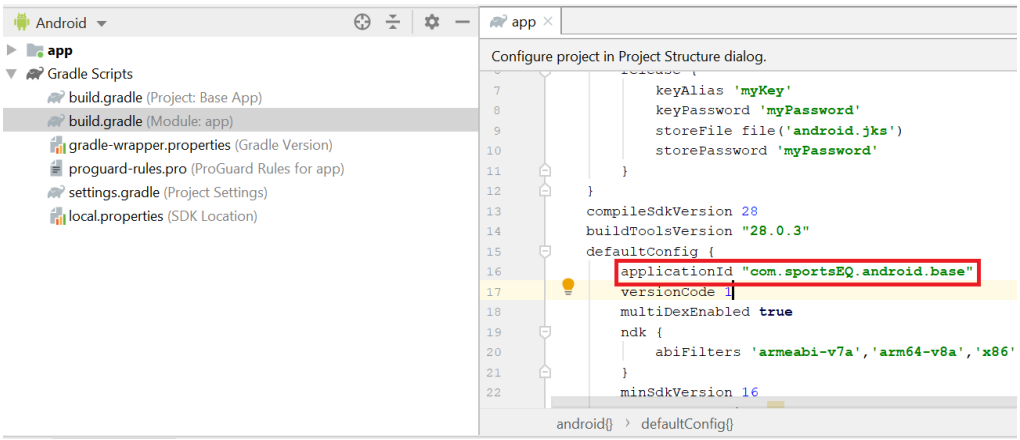
Exercise 5.1: Change the Application ID

The Application ID on Android applications represents a unique identifier. Every Android app has a unique application ID that looks like a Java package name, such as com.example.myapp. This ID uniquely identifies your app on the device and in Google Play Store.

By default, the Application ID is com.microstrategy.android.samples.app defined with the applicationId property in your module's build.gradle file.

Change the application ID

1. Navigate to the **Desktop/Android_Mobile_SDK/My Exercises** folder and duplicate the BaseApp folder.
2. Rename the new folder **SportsEQ**.
3. Open this app in Android Studio.
4. Expand **Gradle Scripts** in the Project window.
5. Double-click **build.gradle (Module: app)**. The file opens in the Editor window as shown below.



6. Locate the **defaultConfig** block of code.
7. Change the applicationID to **com.sportsEQ.android.base**. Keep the quotes in place.
8. Locate the line **version code 1**.

9 Press **Enter**, then add the following line under version code 1:

```
versionName "1.0"
```

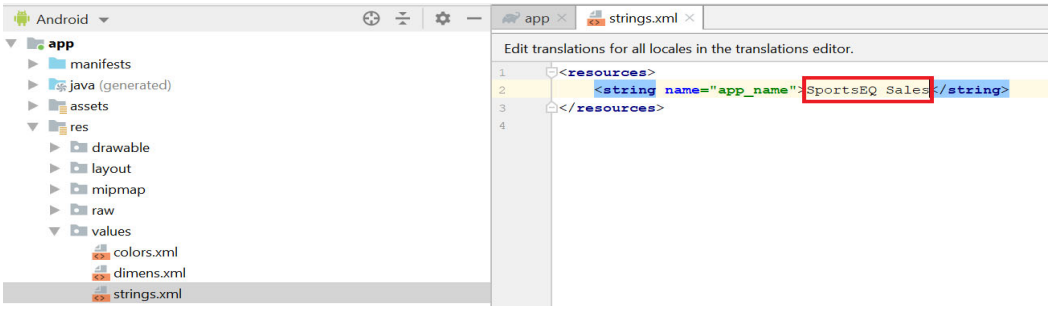
Exercise 5.2: Change the name of the app

The name of the app is displayed in application lists on mobile devices and under the icon of the app. The Mobile Architect may require that the app name reflect both the enterprise and the app functionality.

Modify the strings.xml file

Since this is an app for the sales team, change the name of your app to SportsEQ Sales.

1 In the Project window, expand **app**/**res**/**values**, as shown below.



2 Double-click **strings.xml**. The file opens.

A string resource provides text strings for your application with optional text styling and formatting.

3 Locate the string **"app-name"** parameter.

4 Type **SportsEQ Sales** as the value.

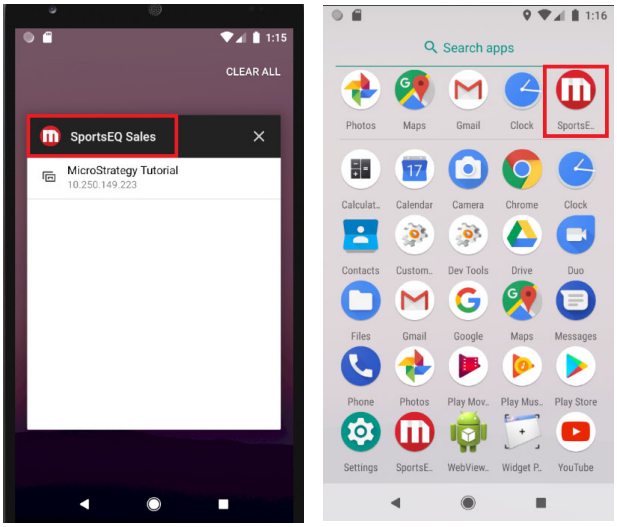
5 Click **File**, then **Save All** to save your changes.

Test your modification

1 Click the **Make project** icon to build your app.

2 Click **Run app** to deploy your app in the emulator.

3 Minimize the app and open the apps menu by clicking the **arrow** icon on the Android home screen. Your application icon shows the new name, and the new name is displayed when viewing all open apps.



Generating app icons with Image Asset Studio

The first thing users see for any mobile app is the app icon. From the start, all enterprise apps should include the corporate logo to strengthen corporate app branding. Android Studio includes a tool called Asset Studio that helps you generate your own app icons from a custom image or text string.

Asset Studio generates a set of icons at the appropriate resolution for each pixel density that your app supports. The newly generated icons are placed in density-specific folders in the resource directory in your project. To change the icons within the MicroStrategy base app, you replace the existing PNG files (keeping the names the same) in the drawable folders and update the existing colors.xml file.

Exercise 5.3: Change the application icon

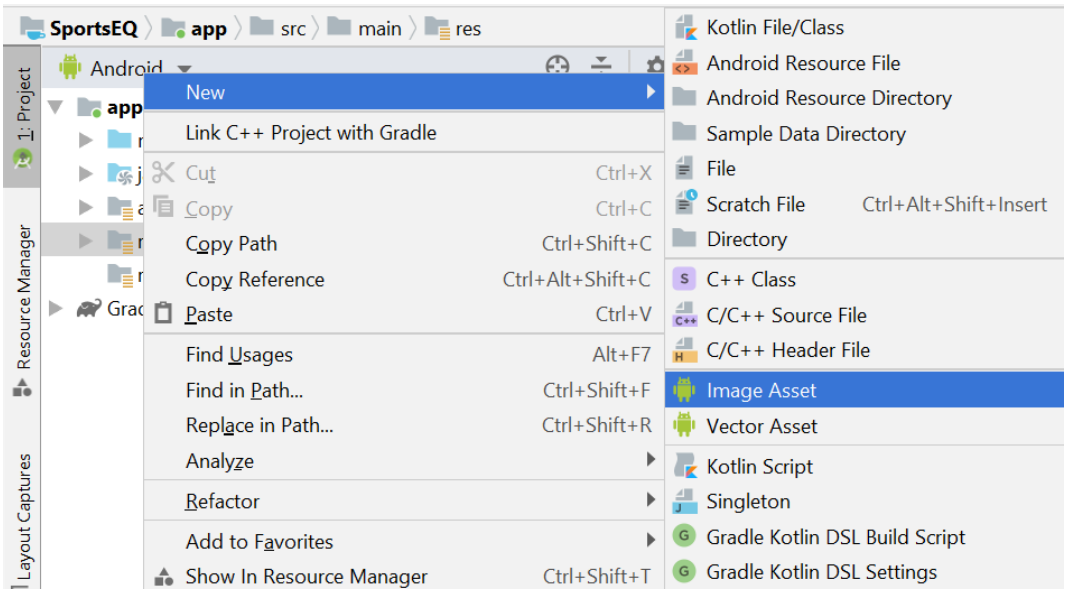
Generate a set of SportsEQ icons using Image Asset Studio. Then use the newly generated icons to replace the default MicroStrategy icon throughout the application and within the Android OS when needed.

Generate the SportsEQ icon set

1 Open the **SportsEQ app** in Android Studio.

2 From the Project window, expand the **app** folder.

3 Right-click the **res** folder, point to **New,** and select **Image Asset**.

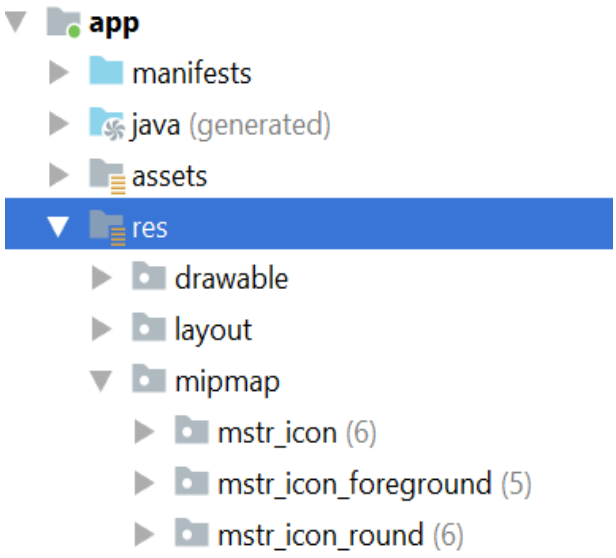


4 From the Configure Asset Image window, make the following changes:

- Icon Type: **Launcher Icons** (Adaptive and Legacy)
- Name: **mstr_icon**
- Asset Type: **Image**
- Path: **Desktop/Android_Mobile_SDK/Logos/sports_eq logo.png**
- Trim: **No**

5 Click **Next**. On the Confirm Icon Path screen, click **Finish**.

The new icons have been created and placed in the appropriate mipmap folders in the res directory as shown below.



Replace the default icons with the newly generated icons

To complete the icon customization process, the newly generated icons need to be placed in their respective drawable folders in the resource directory. Image Asset Studio generates three types of icons: Launcher icons, Action bar and tab icons, and Notification icons. Each icon type is used in different places within the mobile app and the Android OS.

1 Open a File Explorer window and navigate to **Desktop/ Android_Mobile_SDK/My Exercises/SportsEQ/app/src/main/res.**

2 Copy and paste the newly generated SportsEQ icons from the mipmap folders to their respective drawable folders outlined in the table below:

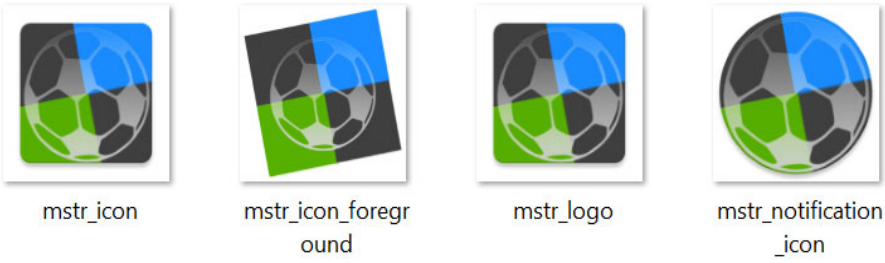
Mipmap Folder	Drawable Folder
mipmap-hdpi	drawable-hdpi
mipmap-mdpi	drawable-mdpi
mipmap-xhdpi	drawable-xhdpi
mipmap-xxhdpi	drawable-xxhdpi

3 In each of the drawable folders, make the following changes:

a Rename the **mstr_icon_round.png** image to **mstr_notification_icon.png**. This file will be used for notifications that the app will deliver to the user.

b Duplicate the **mstr_icon.png** image. Rename it **mstr_logo.png**. This icon is viewed in the action bar of the application.

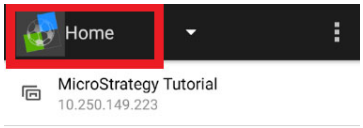
Each of your drawable folders should contain the four icons shown below.

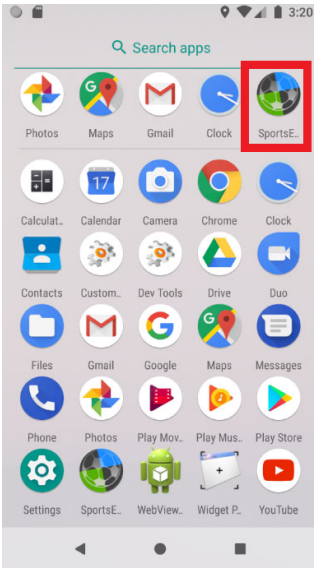


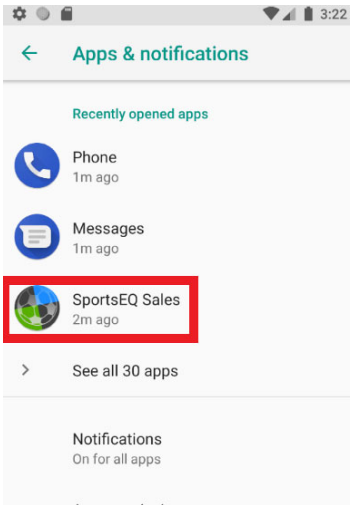
Test your modification

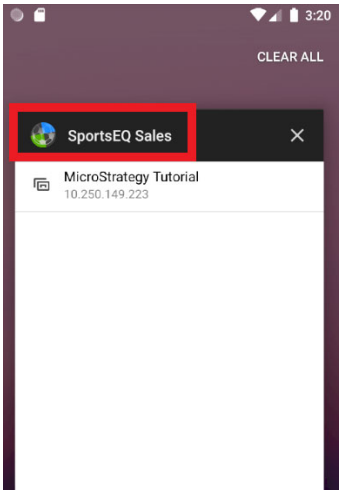
1 Build and run your app in the emulator.

2 Explore the emulator to see the new icon in the locations shown below:









Locations to view the new icon:

- When viewing the currently opened app
- In the Applications list
- When reviewing the apps notifications settings
- When opening the Action Bar inside the app

Customizing the Splash Screen for the application

After launching the app, your Mobile Architect may require that designers add a splash screen as the app loads, standardized according to corporate branding. Splash screens are important for marketing purposes as they display the company that designed the app, copyright, and legal information. Splash screens also improve user experience by allowing users to view something new while the app content loads.

You can customize the splash screen by:

- Replacing the entire splash screen with a custom image.

  Provide two splash screens, one for portrait mode and one for landscape mode. Include copyright and legal information on each custom image.

- Replacing the splash screen icon with a custom image and customizing the text on the splash screen.

  Customize the splash screen icon, background color, text, font size, and font color for the copyright and legal information.

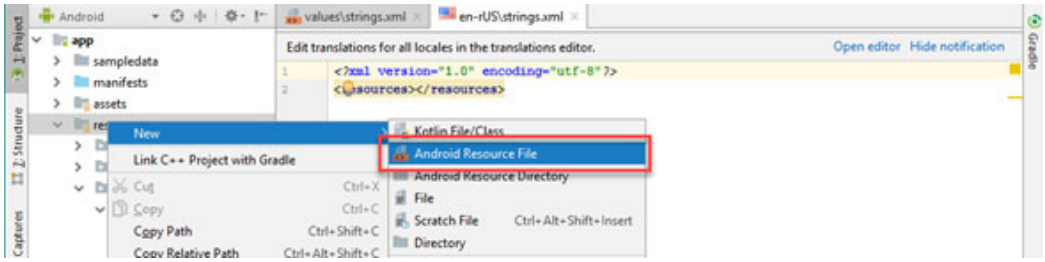
Exercise 5.4: Modify the copyright information

Your Mobile Architect requires that all enterprise apps contain and display the appropriate copyright information to comply with SportsEQ's corporate legal terms.

Change copyright information

1 In the Project window, expand the **app** folder.

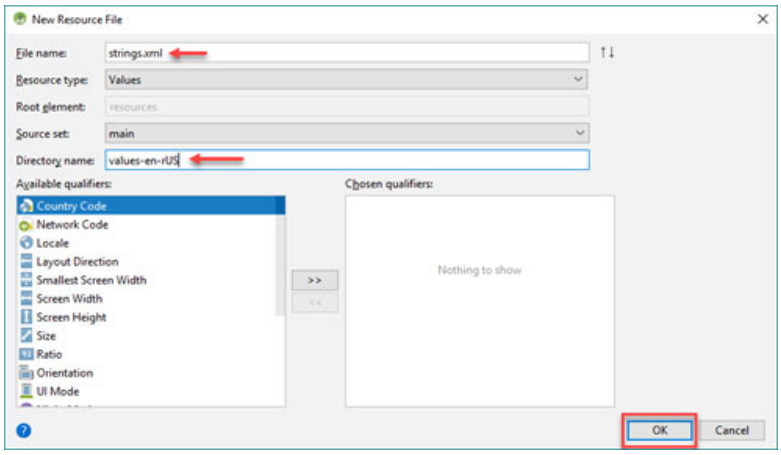
2 Right-click the **res** folder, select **New**, and then select **Android Resource File**, as shown below.



3 Enter the following information in the window. Leave the other fields as is.

- File Name: **strings.xml**
- Directory Name: **values-en-rUS**

Your window should match the image below.



4 Click **OK**.

You created a folder called **values-en-rUS**. This folder is a sibling of the values folder inside res. The file will display below the first instance of strings.xml. Notice the added path (en-rUS), reflecting the copyright is in English and for the United States.

5 In the new strings.xml folder, type the following line between the <resources> and </resources> tags.:

```
<string name="SPLASH_COPYRIGHT">Copyright SportsEQ
Sales 2019</string>
```

6 **Save** the file.

Exercise 5.5: Display the version number

Best Practice

The version number is displayed on the splash screen below the copyright information. Your team should always update the version number with each app upgrade and deployment. However, the version number is not displayed by default. You can add your own string of characters to the splash screen by modifying the strings.xml file.

Add the version number to the splash screen

1 In the same strings.xml file you created in the previous exercise, type the following line of code between: </string> and </resources>.

```
<string name="SPLASH_LEGAL">11.2</string>
```

The file should look like the picture below:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="SPLASH_COPYRIGHT">Copyright SportsEQ Sales 2019</string>
    <string name="SPLASH_LEGAL">11.2</string>
</resources>
```

2 **Save** the file.

Exercise 5.6: Replace the splash screen icon with a custom image

In this exercise, you customize SportsEQ's splash screen by replacing the current splash screen icon with a custom image. You also set the background and text colors using their corporate colors. The new splash screen will use a light gray as the background and a dark gray for the text, along with the company's approved splash screen icon.

Add the splash screen image to the drawable folders

1. Open File Explorer and navigate to **Desktop\Android_Mobile_SDK\Logos**.
2. Copy **sports_eq splash.png**.
3. In File Explorer, navigate to **Desktop/Android_Mobile_SDK/My Exercises/SportsEQ/app/src/main/res.**
4. Inside each of the four drawable-XXX folders, paste **sports_eq splash.png** and rename the image **mstr_splash.png**.

   To adjust to the size of the devices resolutions, you may want to use larger images. For our purpose, the image will be resized for us.

Adjust the background and text colors of the splash screen

1. In the Project window in Android Studio, expand **app**/**res**/**values**.
2. Double-click the **colors.xml**.
3. Locate "**splashscreen_background_color**" in the code in the editor, click the red square next to it and change the following values:
   - R: **175** G: **175** B: **175**
4. Locate "**splashscreen_text_color**" in the code in the editor, click the white square next to it and change the following values:

- R: **63** G: **63** B: **36**

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="splashscreen_background_color">#AFAFAF</color>
    <color name="splashscreen_text_color">#3F3F3F</color>
    <color name="mstr_icon_bg">#FFD9222D</color>
    <color name="notification_tint_color">#FFD9232E</color>
</resources>
```

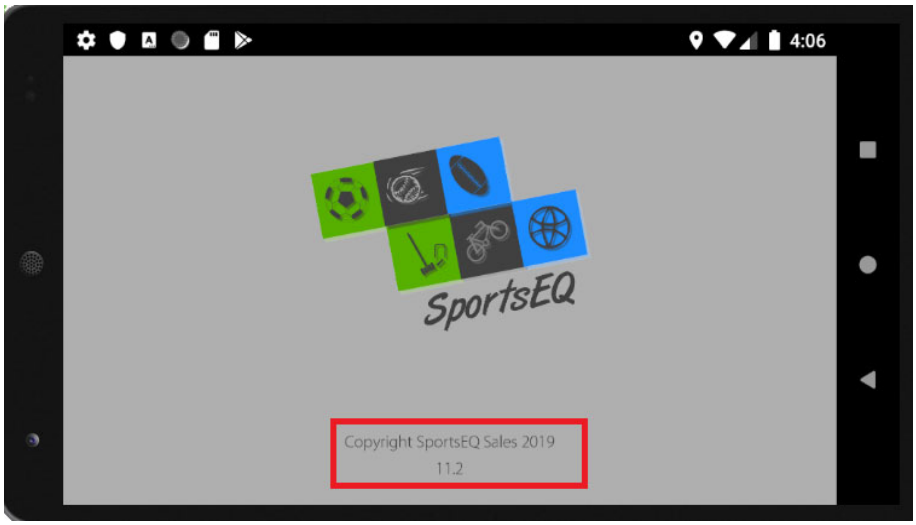
As an alternative, you can also change the hexadecimal values in the code to the ones shown above. Android Studio can understand color in many formats. The one used above is #AARRGGBB, where each pair represents a hexadecimal value from 0(00) to 255(FF). There is a pair for each color component: RR for red, GG for green, and BB for blue.

5 **Save** the file.

Test your modification

1 Build and run your app in the emulator.

Below is the customized splash screen in both portrait and landscape modes. Notice the copyright and version number at the bottom of each.



6

Build and Deploy Your App

In the previous chapters, you started with a clone of the MicroStrategy Mobile app and turned it into a customized application for SportsEQ. These customizations represent the basic branding that is most commonly applied. Now, we can turn our attention to the content within the application itself.

In this chapter, you:

- Install a custom font and use it on a document.
- Pass attribute element values using a selector
- Link multiple documents to build SportsEQ Sales mobile app.

Custom fonts

Fonts can carry a message, mood, or even an atmosphere, and can be an integral part of the corporate brand. In your enterprise, the marketing department probably puts great effort into defining which fonts to use for specific purposes. Fonts are part of the image your company wants to convey.

Using Mobile SDK, a custom font that MicroStrategy Mobile does not support out-of-the-box can be added and viewed on a mobile device. For Android devices, the font file must be added to the MicroStrategy Mobile project and

configured in the FontsMapping.xml file for the project. In addition, a plug-in needs to be installed in MicroStrategy Web to use the font in a document.

Using fonts in documents

A document built with MicroStrategy Web can include labels that use any font in the system where the document is created. However, to have that font display properly in Web, the target PC must have the font installed too.

The same is true for mobile consumption. In Android and iOS you can display various fonts using a similar approach. The process is slightly different depending on whether the font is supported by the operating system by default. Android supports two fonts by default, Roboto-Regular.ttf and DroidSansMono.ttf. Whether the font is supported by default in the operating system or not, you need to install a plug-in for MicroStrategy Web to recognize and make it available in a document.

Exercise 6.1: Install the font plug-in

In this exercise, you add a font plug-in to MicroStrategy Web to use in a document.

Install the font plug-in

1 On the remote desktop, navigate to the **Android_Mobile_SDK/Exercises** folder.

2 Copy the **FontPlugin** folder, and paste it in **sdk_workshop/war/MicroStrategy/plugins**.

3 From the plugins folder, navigate to **FontPlugin/WEB-INF/xml/config**. Open the **fontNamePicker.xml** file in Notepad++.

4 In the fontNamePicker.xml file, change each instance of **Zapfino** to **KaushanScript.**

Your updated file should look like the one below:

```
<shortcut-list name="fontPicker">
  <shortcut desc="KaushanScript" name="pkrKaushanScript">
    <attributes>
      <attribute name="style" source="const" value="font-family:KaushanScript"/>
      <attribute name="value" source="const" value="KaushanScript-Regular"/>
    </attributes>
  </shortcut>
</shortcut-list>
```

5 **Save** and close the fontNamePicker.xml file.

6 On the desktop, right-click the **RestartTomcat.bat** file and select **Run as administrator**. In the security alert window, click **Yes**.

The Tomcat server needs to restart for the changes to take effect.

Exercise 6.2: Use the custom font in a document

In this exercise, you create the landing page that is used in the SportsEQ Sales mobile app. The landing page document will use the custom font installed in the previous exercise.

Log in to MicroStrategy Web on the remote desktop

1. Log in to the **remote desktop** of your MicroStrategy Cloud environment.
2. From a Chrome browser on the remote desktop, navigate to:

   http://localhost:8080/MicroStrategy/servlet/mstrWeb
3. Select **MicroStrategy Tutorial** and login with your credentials from the MicroStrategy Cloud email.
4. Click **Go to MicroStrategy Web**. The Shared Reports folder opens.

Create the Landing page document

1. In the Shared Reports folder, create a new folder named **Android App**.
2. Click **Create**, point to **New Document**, and select the **08 One Content Top Two Contents Bottom** template.
3. In the document editor, right-click **Panel Stack 1** and select **Properties and Formatting**.
4. In the Properties and Formatting window, select **General** in the list of properties on the left and clear the **Show Title Bar** check box.
5. Select **Color and Lines** in the list of properties. Set the **Fill Color** to **Gray-80%** and **Borders** to **None**.
6. Click **OK**.

Insert the document title

7. From the **Insert** menu, select **Text**.
8. Use your cursor to draw a text box on Panel Stack 1.

9 In the text box, type **SportsEQ Sales**.

10 Right-click the text box and select **Properties and Formatting**.

11 In the Properties and Formatting window, select **Color and Lines** on the left.

12 Change the **Fill Color** to **No Fill** and **Borders** to **None**.

13 Select **Font** on the left and make the following changes:

- Font: **KaushanScript**
- Size: **48**
- Color: **white**

The newly added custom font displays at the bottom of the font list. The text will display as Times New Roman in MicroStrategy Web since the font file has not been added, but will display correctly on the mobile device once the font file has been added to the mobile project.

14 Select **Alignment** on the left and set the **Text Alignment** to **Center.**

15 Click **OK.**

16 Save the document as **SportsEQ Sales** in the **Android App** folder.

17 Return to **Design Mode**.

Insert SportsEQ's logo

1 From the **Insert** menu, select **Image**.

2 Use your cursor to draw the image on **Panel Stack 1**.

3 From the Properties and Formatting window, click **Browse** and navigate to **Desktop/Android_Mobile_SDK/Logos.**

4 Select **sports_eq logo** and click **Open.**

5 Click **Color and Lines** on the left, and select **None** under Borders.

6 Click **OK** to close the Properties and Formatting window.

7 Resize and position the SportsEQ's logo on the left side of the title, as shown below.

8 **Save** the document.

Exercise 6.3: Add a font file to the mobile project

To complete the custom font installation, the font file must be added to the MicroStrategy Mobile project and configured in the FontsMapping.xml file for the project.

Add font file to the mobile project

1. On your local computer, navigate to **Android_Mobile_SDK/Exercises.**
2. Copy the **KaushanScript-Regular.otf** file.
3. On your local computer, navigate to **Android_Mobile_SDK/SportsEQ/app/src/main/assets/fonts** and paste the font file.
4. From the current fonts folder, open the **FontsMapping.xml** file in a text editor.
5. Add the following block of code inside of the **<fonts> </fonts>** tag:

```
<font>

<custFontName>KaushanScript</custFontName>

<custFontFile>KaushanScript-Regular.otf
</custFontFile>

</font>
```

6. Save and close the **FontsMapping.xml** file.

Test the custom font

1. If not already open, open your **SportsEQ** app in Android Studio.
2. Build your SportsEQ project and run it in the emulator.
3. Once your app opens in the emulator, select the **MicroStrategy Tutorial** project.
4. From the Shared Reports folder, open your **Android App** folder and run your **SportsEQ Sales** document.

Your document should look similar to the one below. Notice that the custom font displays on the mobile device since the font file was added to the mobile project.

Exercise 6.4: Complete the SportsEQ Sales Landing Page

In this exercise, complete the SportsEQ Sales app by adding buttons and a selector to the landing page. The buttons link to the two most highly used documents by the SportsEQ Sales team and the selector allows each user to customize their view of the data.

Add a dataset for the selector

To pass selector values from the landing page document to the target document, both the landing page and the target must contain the same selector or dataset object. Adding the dataset used in the SportsEQ document to the landing page allows you to configure the Customer State selector.

1. On the remote desktop, open **MicroStrategy Web** from a browser.
2. From the MicroStrategy Tutorial project, navigate to the **Shared Reports/ Android App** folder and open your **SportsEQ Sales** document.
3. In the SportsEQ Sales document, click **Add a dataset** on the left.
4. Select the **Customer State Summary** report from Shared Reports/ MicroStrategy Platform Capabilities/MicroStrategy Report Services/ MicroStrategy Widget Library/Datasets**.**
5. Click **OK**. The dataset is added to the document.
6. From the **Insert** menu, point to **Selector** and select **Drop-down**.
7. Draw the selector below the title on Panel Stack 1.
8. Right-click the selector and select **Properties and Formatting**.
9. Click **Selector** from the list of properties on the left.
10. Verify **Action Type** is **Select Attribute Element**.
11. From the **Source** drop-down list, select **Customer State**.
12. To set the target for the selector, select **Click here to enable manual control of targets for this layout**.
13. Click **OK** on the warning stating you will need to manually maintain targets for all selectors in this layout.

14 Select **Panel Stack 2** from the **Available** list and click the **right arrow** to move it to the Selected list. Click **OK**.

15 **Save** the document.

Add Customer Sales by State button

1 From the **Insert** menu, select **Button** and **Caption only**.

2 Use your cursor to draw the button on Panel Stack 2.

3 Double-click the button and type **Customer Sales by State**.

4 Right-click the button and select **Properties and Formatting**.

5 From the Properties and Formatting window, select **Button** from the list of properties on the left.

6 On the bottom of the Button window, click the **Configure actions on this button**.

7 Under When this link is clicked, select **Run this report or document**.

8 Click the **(browse button)** to select the target document.

9 From Shared Reports, navigate to **MicroStrategy Platform Capabilities/ MicroStrategy Report Services/MicroStrategy Widget Library** and select the **Sales by Customer State** document. Click **OK**.

10 From the **Pass all selector values** drop-down list on the bottom of the Links Editor, select **Match selectors by source attribute**. Click **OK**.

When configuring a link that passes selector values, you can choose to match the selector's value by either the selector name or the source object. In this step, the link is configured to match the selector's value by the source attribute Customer State.

11 Click **Close** on the Properties and Formatting window.

12 **Save** the document.

Add the Financial Statements button

1. From the **Insert** menu, select **Button** and **Caption only**.
2. Use your cursor to draw the button on Panel Stack 3.
3. Double-click the button and type **Financial Statements**.
4. Right-click the button and select **Properties and Formatting**.
5. From the Properties and Formatting window, select **Button** from the list of properties on the left.
6. On the bottom of the Button window, click the **Configure actions on this button**.
7. Under When this link is clicked, select **Run this report or document**.
8. Click the **(browse button)** to select the target document.
9. From Shared Reports, navigate to **MicroStrategy Platform Capabilities/ MicroStrategy Report Services/MicroStrategy Widget Library** and select the **Financial Statements** document. Click **OK**.
10. Click **OK**, and then **Close**.

Format the button panel stacks

1. Right-click **Panel Stack 2** and select **Properties and Formatting**.
2. In the Properties and Formatting window, select **General** in the list of properties on the left. Clear the **Show Title Bar** check box.
3. Select **Color and Lines** from the list of properties on the left. Set the **Fill Color** to **No Fill** and **Borders** to **None**.
4. Click **OK**.
5. Right-click **Panel Stack 3** and select **Properties and Formatting**.
6. In the Properties and Formatting window, select **General** in the list of properties on the left. Clear the **Show Title Bar** check box.

7 Select **Color and Lines** on the left. Set the **Fill Color** to **No Fill** and **Borders** to **None**.

8 Click **OK**.

9 **Save** and **close** your document.

Test your app in the emulator

1 On your local computer, open the SportsEQ app in Android Studio.

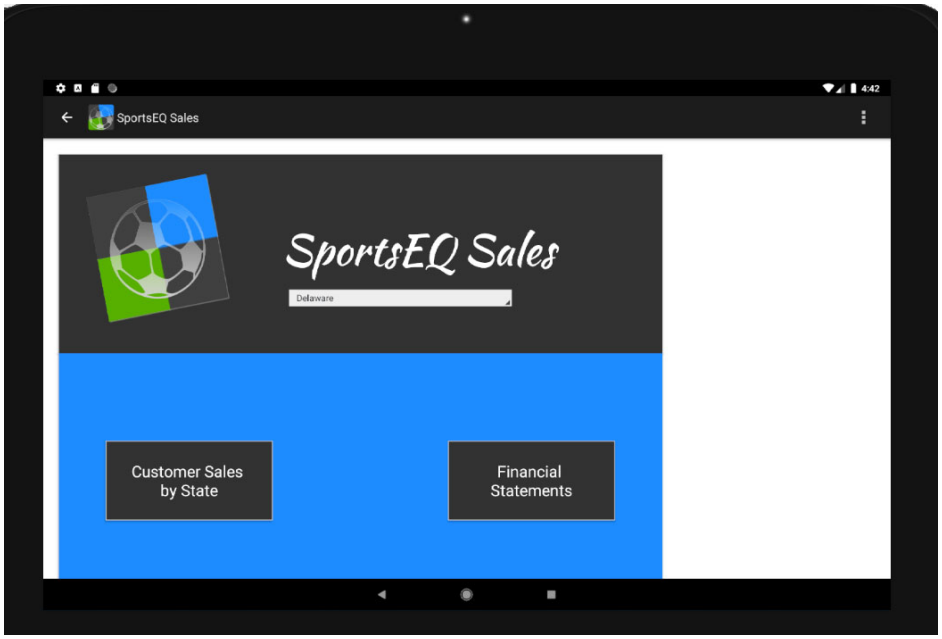
2 Build your app and run it in the emulator.

3 Once your app opens in the emulator, select the **MicroStrategy Tutorial** project.

4 From the Shared Reports folder, open your **Android App** folder and run your **SportsEQ Sales** document.

5 Interact with your document by testing your selector and buttons.

Your completed document should look similar to the one below.



Congratulations! SportsEQ Sales now has a custom mobile application that has been pre-configured and re-branded to their corporate standards.

Copyright Information

Trademark Information