

Windows 10

“Project Astoria” Developer Guide

JULY 2015, V1.3

This guide provides guidance to TAP partners working with the “Project Astoria” SDK. Features and functionality described within are subject to change and should therefore be considered preliminary.

Microsoft Corporation Technical Documentation License Agreement (Standard)

READ THIS! THIS IS A LEGAL AGREEMENT BETWEEN MICROSOFT CORPORATION ("MICROSOFT") AND THE RECIPIENT OF THESE MATERIALS, WHETHER AN INDIVIDUAL OR AN ENTITY ("YOU"). IF YOU HAVE ACCESSED THIS AGREEMENT IN THE PROCESS OF DOWNLOADING MATERIALS ("MATERIALS") FROM A MICROSOFT WEB SITE, BY CLICKING "I ACCEPT", DOWNLOADING, USING OR PROVIDING FEEDBACK ON THE MATERIALS, YOU AGREE TO THESE TERMS. IF THIS AGREEMENT IS ATTACHED TO MATERIALS, BY ACCESSING, USING OR PROVIDING FEEDBACK ON THE ATTACHED MATERIALS, YOU AGREE TO THESE TERMS.

1. For good and valuable consideration, the receipt and sufficiency of which are acknowledged, You and Microsoft agree as follows:

(a) If You are an authorized representative of the corporation or other entity designated below ("**Company**"), and such Company has executed a Microsoft Corporation Non-Disclosure Agreement that is not limited to a specific subject matter or event ("**Microsoft NDA**"), You represent that You have authority to act on behalf of Company and agree that the Confidential Information, as defined in the Microsoft NDA, is subject to the terms and conditions of the Microsoft NDA and that Company will treat the Confidential Information accordingly;

(b) If You are an individual, and have executed a Microsoft NDA, You agree that the Confidential Information, as defined in the Microsoft NDA, is subject to the terms and conditions of the Microsoft NDA and that You will treat the Confidential Information accordingly; or

(c) If a Microsoft NDA has not been executed, You (if You are an individual), or Company (if You are an authorized representative of Company), as applicable, agrees: (a) to refrain from disclosing or distributing the Confidential Information to any third party for five (5) years from the date of disclosure of the Confidential Information by Microsoft to Company/You; (b) to refrain from reproducing or summarizing the Confidential Information; and (c) to take reasonable security precautions, at least as great as the precautions it takes to protect its own confidential information, but no less than reasonable care, to keep confidential the Confidential Information. You/Company, however, may disclose Confidential Information in accordance with a judicial or other governmental order, provided You/Company either (i) gives Microsoft reasonable notice prior to such disclosure and to allow Microsoft a reasonable opportunity to seek a protective order or equivalent, or (ii) obtains written assurance from the applicable judicial or governmental entity that it will afford the Confidential Information the highest level of protection afforded under applicable law or regulation. Confidential Information shall not include any information, however designated, that: (i) is or subsequently becomes publicly available without Your/Company's breach of any obligation owed to Microsoft; (ii) became known to You/Company prior to Microsoft's disclosure of such information to You/Company pursuant to the terms of this Agreement; (iii) became known to You/Company from a source other than Microsoft other than by the breach of an obligation of confidentiality owed to Microsoft; or (iv) is independently developed by You/Company. For purposes of this paragraph, "Confidential Information" means nonpublic information that Microsoft designates as being confidential or which, under the circumstances surrounding disclosure ought to be treated as confidential by Recipient. "Confidential Information" includes, without limitation, information in tangible or intangible form relating to and/or including released or unreleased Microsoft software or hardware products, the marketing or promotion of any Microsoft product, Microsoft's business policies or practices, and information received from others that Microsoft is obligated to treat as confidential.

2. You may review these Materials only (a) as a reference to assist You in planning and designing Your product, service or technology ("Product") to interface with a Microsoft Product as described in these Materials; and (b) to provide feedback on these Materials to Microsoft. All other rights are retained by Microsoft; this agreement does not give You rights under any Microsoft patents. You may not (i) duplicate any part of these Materials, (ii) remove this agreement or any notices from these Materials, or (iii) give any part of these Materials, or assign or otherwise provide Your rights under this agreement, to anyone else.

3. These Materials may contain preliminary information or inaccuracies, and may not correctly represent any associated Microsoft Product as commercially released. All Materials are provided entirely "AS IS." To the extent permitted by law, MICROSOFT MAKES NO WARRANTY OF ANY KIND, DISCLAIMS ALL EXPRESS, IMPLIED AND STATUTORY WARRANTIES, AND ASSUMES NO LIABILITY TO YOU FOR ANY DAMAGES OF ANY TYPE IN CONNECTION WITH THESE MATERIALS OR ANY INTELLECTUAL PROPERTY IN THEM.

4. If You are an entity and (a) merge into another entity or (b) a controlling ownership interest in You changes, Your right to use these Materials automatically terminates and You must destroy them.

5. You have no obligation to give Microsoft any suggestions, comments or other feedback ("Feedback") relating to these Materials. However, any Feedback you voluntarily provide may be used in Microsoft Products and related specifications or other documentation (collectively, "Microsoft Offerings") which in turn may be relied upon by other third parties to develop their own Products. Accordingly, if You do give

Microsoft Feedback on any version of these Materials or the Microsoft Offerings to which they apply, You agree: (a) Microsoft may freely use, reproduce, license, distribute, and otherwise commercialize Your Feedback in any Microsoft Offering; (b) You also grant third parties, without charge, only those patent rights necessary to enable other Products to use or interface with any specific parts of a Microsoft Product that incorporate Your Feedback; and (c) You will not give Microsoft any Feedback (i) that You have reason to believe is subject to any patent, copyright or other intellectual property claim or right of any third party; or (ii) subject to license terms which seek to require any Microsoft Offering incorporating or derived from such Feedback, or other Microsoft intellectual property, to be licensed to or otherwise shared with any third party.

6. Microsoft has no obligation to maintain confidentiality of any Microsoft Offering, but otherwise the confidentiality of Your Feedback, including Your identity as the source of such Feedback, is governed by Your NDA.

7. This agreement is governed by the laws of the State of Washington. Any dispute involving it must be brought in the federal or state superior courts located in King County, Washington, and You waive any defenses allowing the dispute to be litigated elsewhere. If there is litigation, the losing party must pay the other party's reasonable attorneys' fees, costs and other expenses. If any part of this agreement is unenforceable, it will be considered modified to the extent necessary to make it enforceable, and the remainder shall continue in effect. This agreement is the entire agreement between You and Microsoft concerning these Materials; it may be changed only by a written document signed by both You and Microsoft.

©2015 Microsoft. All rights reserved.



Contents

Introduction	6
Prerequisites	6
Change history for the documentation.....	6
Set up your development environment	7
Set up your development computer.....	7
PC – Install pre-requisites for the wconnect tool	11
Configure your phone to install Windows Insider Preview builds.....	11
Deploy and run your app	12
Connect to your phone by using the wconnect tool	12
To connect to your phone over USB (PC).....	12
To connect to your phone over Wi-Fi (Mac or PC)	13
Deploy using adb.....	15
Deploy and run your app from the IDE.....	15
Configure your app to use Microsoft services instead of Google Play services.....	16
Add a Windows product flavor that uses the Microsoft Services interop library	16
Add the configuration file for Microsoft services	18
Configure specific services	19
Configuring Analytics	19
Configuring Maps.....	22
Configuring Ads (without Ad Network Mediation)	23
Configuring Ads (with Ad Network Mediation)	26
Configuring Cloud Messaging	29
Configuring Location Services	30
Platform differences between Android and “Project Astoria”	31
Activity stack management.....	31
Standard behavior.....	31
Single top behavior	32
New task behavior	34
Clear top behavior.....	35
Switching between activity stacks	37
Appendix A: Troubleshooting.....	39

Trouble connecting to the device?	39
Appendix B: wconnect tool reference	41
wconnect command line options.....	41
wconnect errors	41
Appendix C: Using adb commands with "Project Astoria"	42
adb errors.....	42
Appendix D: Enabling developer mode on the phone.....	43
Appendix E: Getting the IP address of the phone	46


Introduction

“Project Astoria” (also called “Project A”) enables you to leverage your Android™ code and toolset to build apps for devices running Windows 10 Mobile. Using your existing tools like IntelliJ and Android Studio™ “Project Astoria” enables you to:

- Build Windows apps with a great user experience for phones and other mobile devices with few code changes
- Use a Microsoft interop library to replace most Google Play™ services with Microsoft services with very little effort
- Test and debug your app with your preferred IDE

Using this guide, you’ll be able to do the following:

1. Install the “Project Astoria” SDK on your Windows or Mac development computer.
2. Deploy and run your Android app on a phone running Windows 10 Mobile.
3. Configure an app that uses Google Play services to use Microsoft services instead. Google Play services are not available on Windows devices.

 **Note:** *If your app does not use Google Play services, you typically do not need to modify or rebuild your app project to work with “Project Astoria”.*

Prerequisites

The “Project Astoria” SDK requires a Windows PC or Mac that meets the following requirements:

- Windows 8.1 or Windows 10 (PC), or OS X Yosemite (Mac).
- The Java JDK (1.7 or later) must be installed and the JAVA_HOME environment variable set.
- IntelliJ or Android Studio must be installed.
- The app project must be targeted at a `minSdkVersion` less than or equal to 19 (KitKat 4.4).

Change history for the documentation

The following table lists the change history for this documentation.

Release	Change history
May	First release.
June	Added the following new sections: <ul style="list-style-type: none">• Activity stack management• Appendix B: wconnect tool reference• Appendix C: Using adb commands with “Project Astoria”
July	Updated the following sections:

™ Android, Android Studio, and Google Play are trademarks of Google Inc.

- [Set up your development environment](#): Added new instructions for installing the SDK using SDK Manager in the IDE and for configuring a retail phone so that it can run Android apps.
- [Deploy and run your app](#): Updated the instructions to make the process easier to follow.
- [Configure your app to use Microsoft services instead of Google Play services](#): Documented how to configure your app project to build a Windows of the APK that uses the Microsoft services interop library and described the new Microsoft services configuration file syntax.
- [Configure specific services](#): Updated these sections with new details for using the new Microsoft services configuration file syntax.

Set up your development environment

This section provides instructions for setting up your development computer and phone to use with the “Project Astoria” SDK.

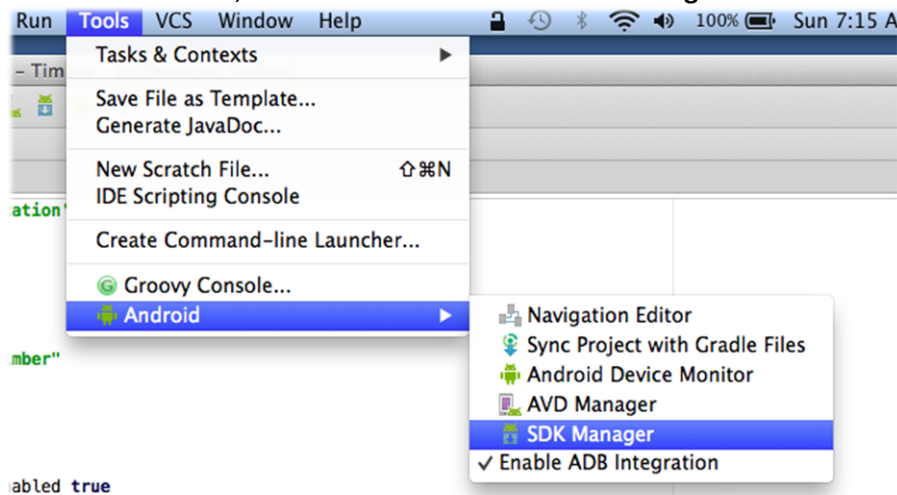
Set up your development computer


The “Project Astoria” SDK is now distributed via the SDK Manager in your IDE. Before installing the latest “Project Astoria” SDK, first uninstall any previous version of the SDK.

To install the SDK for the first time on your development computer, the SDK Manager must be configured with a custom URL. When Microsoft releases subsequent TAP releases of the Microsoft Services SDK, you can use SDK Manager to install updates to the SDK package. For more information about using SDK Manager, see <http://developer.android.com/tools/help/sdk-manager.html>.

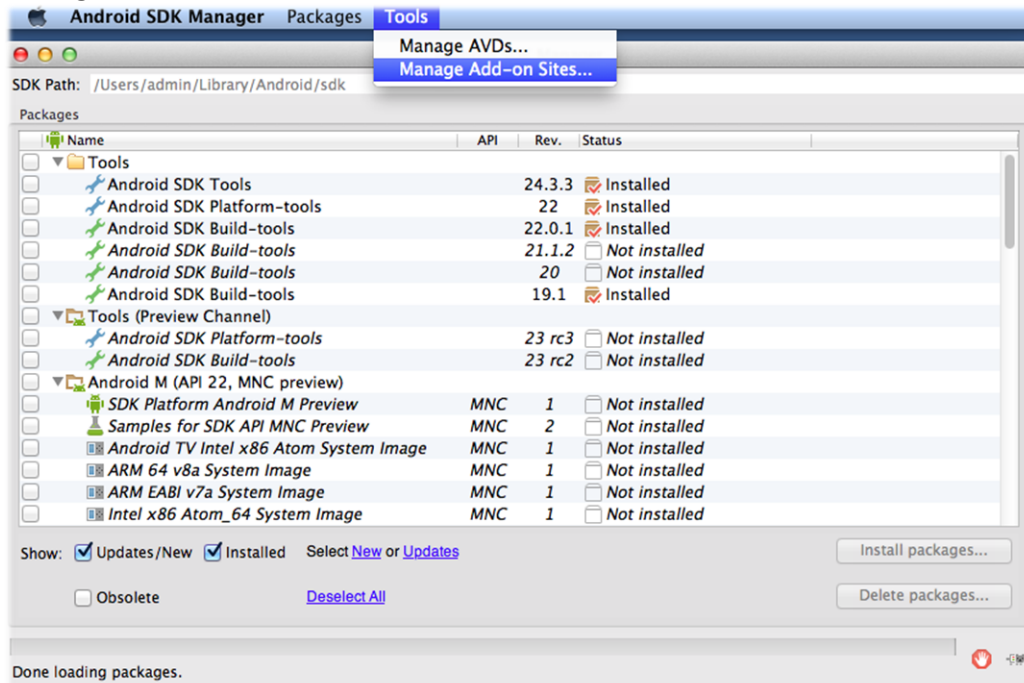
To install the SDK for the first time:

1. Open Android Studio or IntelliJ.
2. On the **Tools** menu, choose **Android** and then **SDK Manager**.

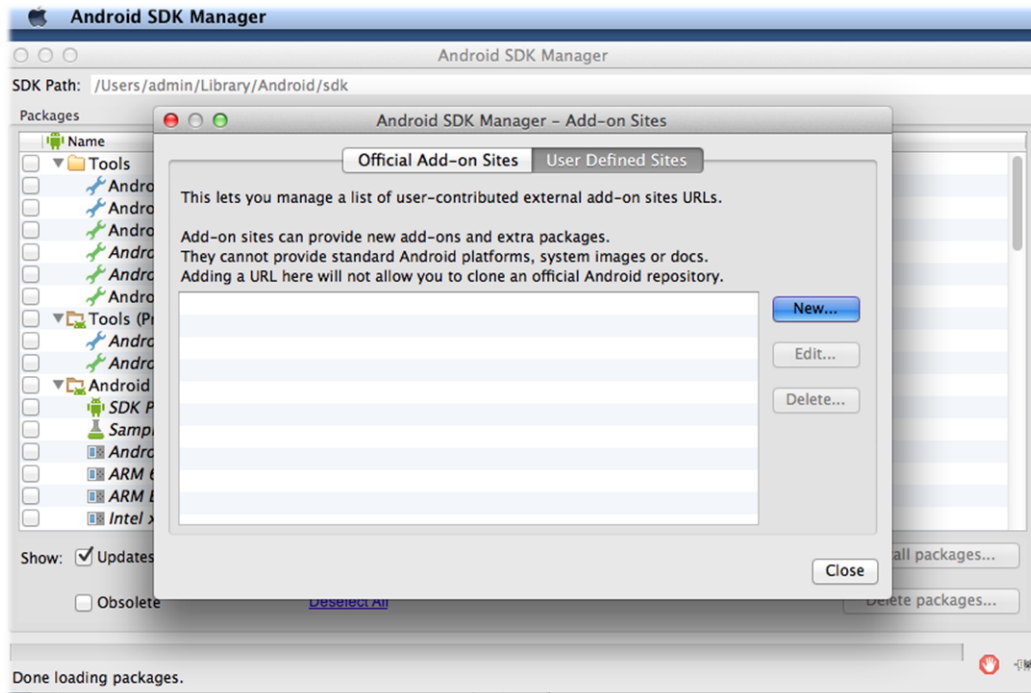


 **Note:** On a Mac, you may need to Command-Tab to the IDE and Command-Tab back to the SDK Manager to get the **SDK Manager** menu to appear.

3. In the **SDK Manager** window, select **Tools > Manage Add on Sites** to open the **Android SDK Manager – Add-on Sites** window.

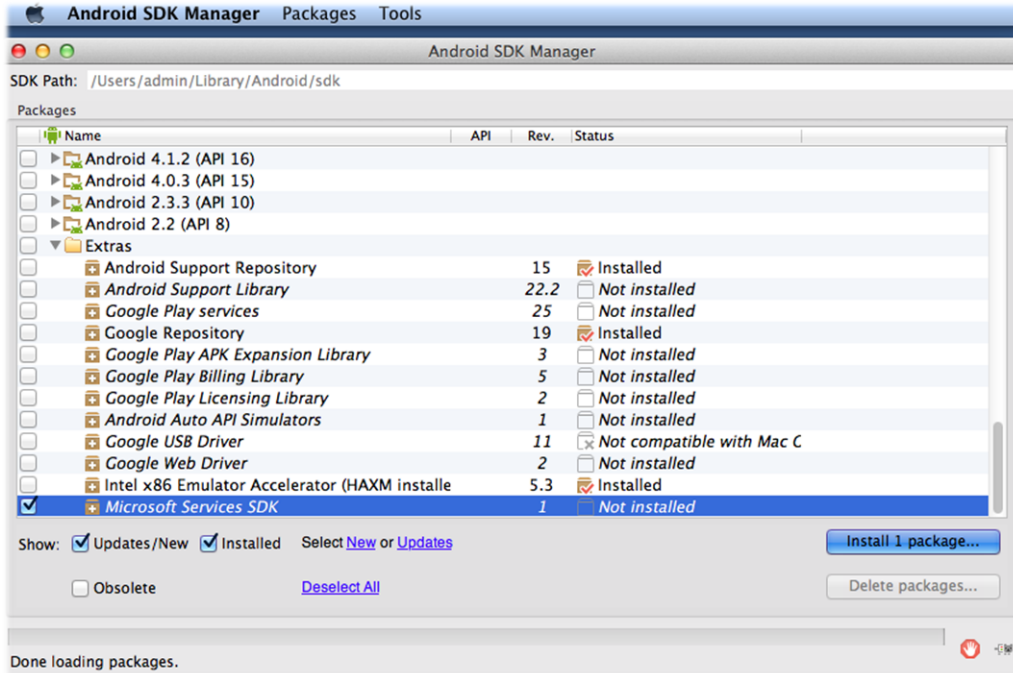


4. In the **Android SDK Manager – Add-on Sites** window, open the **User Defined Sites** tab. Click **New...**, paste the following URL in the URL field, and click **OK**:
 - a. PC:
 - b. Mac:




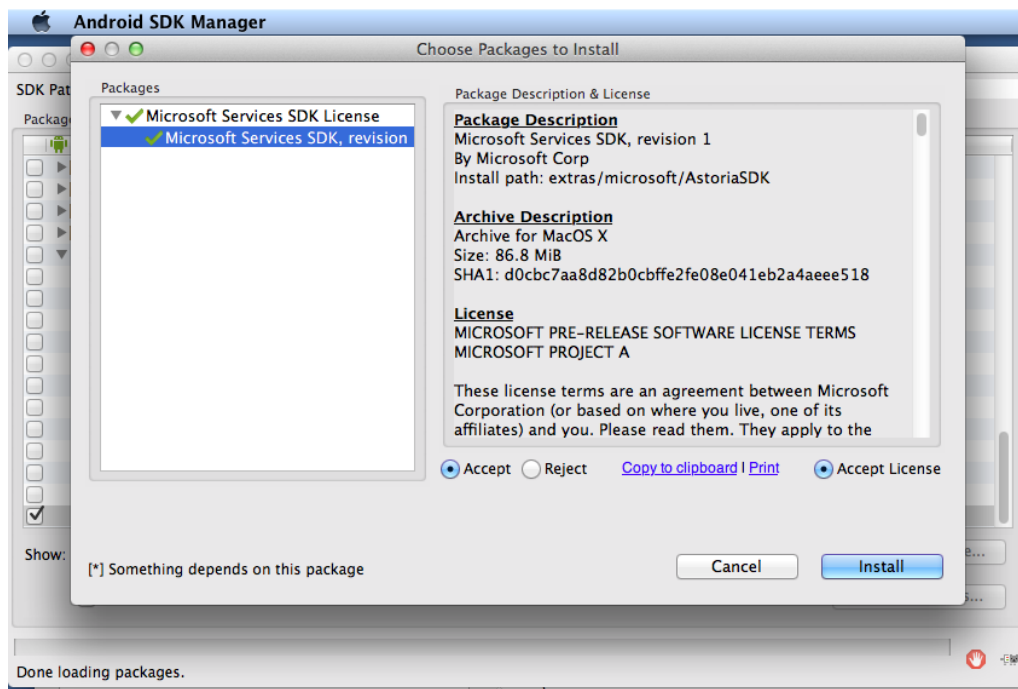
5. In the **Android SDK Manager – Add-on Sites** window, click **Close**.

6. In the **SDK Manager** window, select **Deselect All**. Scroll to the bottom of the list of packages, and under the **Extras** group, check the box next to **Microsoft Services SDK** and click **Install**.



7. In the **Choose Packages to Install** window, select **Microsoft Services SDK**. Then click **Accept License** and **Install**. The SDK will be installed to your Android SDK path under \extras\Microsoft\AstoriaSDK.

 **Note:** On a Mac, you must select the child **Microsoft Services SDK** node.



PC – Install pre-requisites for the wconnect tool

For PCs only (these steps are not required for Macs): Perform the following steps to install several components that are necessary for establishing device connections.

- a. Open Windows Powershell as an administrator.
- b. Enter the following commands:

```
$policy=get-executionpolicy  
set-executionpolicy unrestricted
```

- c. Type **yes** or **y** in response to the question and press Enter.
- d. Enter the following command to change the directory to the extras\Microsoft\AstoriaSDK\tools folder in your Android SDK path. The default Android SDK path is %LOCALAPPDATA%\Android\sdk.

```
cd ${env:LOCALAPPDATA}\Android\sdk\extras\Microsoft\AstoriaSDK\tools
```

- e. Enter the following command to run a script that installs several necessary components:

```
.\install.ps1
```

- f. After the script is finished running, enter the following command:

```
set-executionpolicy $policy
```

Configure your phone to install Windows Insider Preview builds

“Project Astoria” supports the following retail Windows Phone devices:

- Nokia Lumia 920
- Nokia Lumia 925
- Nokia Lumia 929 (icon)
- Nokia Lumia 830
- Nokia Lumia 930
- Nokia Lumia 1020
- Nokia Lumia 1520
- Nokia Lumia 635
- Nokia Lumia 730
- Nokia Lumia 820
- Nokia Lumia 435
- Nokia Lumia 928

To deploy and run Android apps on one of these phones, do the following:


1. Restore your retail device image using [Nokia Software Recovery Tool](http://www.microsoft.com/en-my/mobile/support/software-recovery/). For more information, see <http://www.microsoft.com/en-my/mobile/support/software-recovery/>.
2. Join the Windows Insider Program and install the Windows Insider app on your phone. For more information, see <http://windows.microsoft.com/en-us/windows/preview-download-phone>.


3. In the app, choose **Get preview builds** and **Insider Fast** to receive the necessary OS update.

Deploy and run your app

With “Project Astoria” we want your development experience to be as seamless as possible. To that end, we support debugging your app from your existing IDE while the app is running on a phone that is running a Windows Insider build of Windows 10 Mobile. The following steps walk you through the procedure to establish a connection with the phone, deploy your app to the phone, and run or debug your app using Android Debug Bridge (adb) or the IDE.

Debugging works only while the phone is unlocked and the screen is on. We recommend that you increase the time after which the screen times out and a password is required to the maximum values. To change these options on the phone, open **Settings**, tap **Personalization**, tap **Lock screen**, and at the bottom of the screen change **Screen times out after** to **Never**.

 **Note:** Apps that use Google Play services must be rebuilt using the “Project Astoria” SDK. For details about modifications required for services, see [Configure your app to use Microsoft services instead of Google Play services](#).

 **Note:** Deploy and debug is only possible on a phone that is running an Insider Preview build of Windows 10 Mobile. Retail phones and the emulator are not supported at this time.

Connect to your phone by using the wconnect tool


Android developers typically use the `adb connect` command to establish a connection to a device before deploying or debugging an app. With “Project Astoria”, you must instead use the **wconnect** tool to establish the device connection. After you create a device connection using **wconnect**, you can use `adb` commands or your IDE to perform tasks such as installing and debugging your app.

The **wconnect** tool supports connections over Wi-Fi or USB. Follow the instructions in the following sections to establish a device connection with the **wconnect** tool. If you have difficulty establishing a connection, see [Appendix A: Troubleshooting](#).

 **Note:** In the current release of the “Project Astoria” SDK, USB connectivity is supported only on PCs. If you are on a Mac, you must use a Wi-Fi connection.

To connect to your phone over USB (PC)

1. Enable developer mode on the phone and get a pairing code. For instructions, see [Appendix D: Enabling developer mode on the phone](#).

 **Note:** After you enable developer mode and get a pairing code, make sure that you leave the Pair device screen open on the phone until you complete the remaining steps to establish a device connection.

2. Open a command prompt, and change the directory to the `extras\Microsoft\AstoriaSDK\tools` folder in your Android SDK path. The default Android SDK path is `%LOCALAPPDATA%\Android\sdk`.

```
cd %LOCALAPPDATA%\Android\sdk\extras\Microsoft\AstoriaSDK\tools
```

3. Start **wconnect** with the **usb** parameter:

```
wconnect.exe usb
```

4. When asked to provide a key, type the pairing code that is currently displayed on the **Pair device** screen on the device. If the pairing code is no longer visible, follow the instructions in [Appendix D: Enabling developer mode on the phone](#) to obtain a pairing code again.

```
C:\Users\Administrator>cd C:\Users\Administrator\AppData\Local\Android\sdk\extras\microsoft\AstoriaSDK\tools
C:\Users\Administrator\AppData\Local\Android\sdk\extras\microsoft\AstoriaSDK\tools>wconnect 192.168.1.2
Creating session...


Please enter the pin for the device to be paired and connected: 1234
emulator-5554 on 192.168.1.2 connected.
```

After **wconnect** succeeds in establishing a connection, the phone appears as an emulator in the list of devices shown by running the **adb devices** command. If **wconnect** reports “Connected on 127.0.0.1:xxxx”, the device will appear in the list as **emulator-xxxy**, where $y=x-1$. For example, in the following screenshot, the device appears as **emulator-5554**.

```
Creating session...
Please enter the pin for the device to be paired and connected: 1234
Connected on 127.0.0.1:5555 ← 5555 means it's
                             emulator-5554, see below
F:\wconnect
> adb devices
List of devices attached
emulator-5554 device ←
```

To connect to your phone over Wi-Fi (Mac or PC)

1. Connect your computer and phone to the same Wi-Fi network, and retrieve the IP address of the phone by following the instructions in [Appendix E: Getting the IP address of the phone](#).
2. Enable developer mode on the phone and get a pairing code. For instructions, see [Appendix D: Enabling developer mode on the phone](#).

 **Note:** After you enable developer mode and get a pairing code, make sure that you leave the **Pair device** screen open on the phone until you complete the remaining steps to establish a device connection.

3. Do the following on your development computer.

From a PC:

- a. Open a command prompt, and change the directory to the `extras\Microsoft\AstoriaSDK\tools` folder in your Android SDK path. The default Android SDK path is `%LOCALAPPDATA%\Android\sdk`.

```
cd %LOCALAPPDATA%\Android\sdk\extras\Microsoft\AstoriaSDK\tools
```

- b. Start **wconnect** using the IP Address you obtained for your phone. For example:

```
wconnect.exe 10.60.204.204
```

From a Mac:

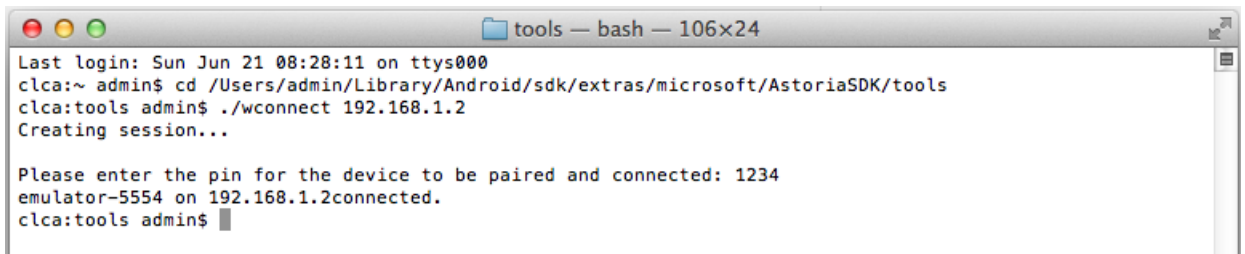
- a. Open a terminal window. Change directory to the extras/microsoft/AstoriaSDK/tools folder in your Android SDK. The default Android SDK path is ~/Library/Android/sdk.

```
cd ~/Library/Android/sdk/extras/microsoft/AstoriaSDK/tools
```

- b. Start **wconnect** using the IP Address you obtained for your phone. For example:

```
./wconnect 10.60.204.204
```

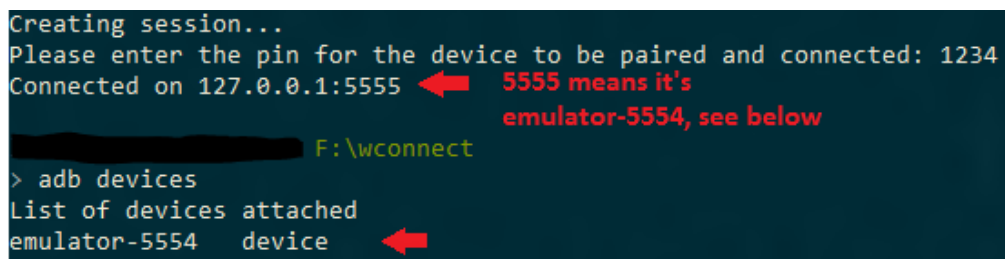
5. When asked to provide a pin, type the pairing code that is currently displayed on the **Pair device** screen on the device. If the pairing code is no longer visible, follow the instructions in [Appendix D: Enabling developer mode on the phone](#) to obtain a pairing code again.



```
tools — bash — 106x24
Last login: Sun Jun 21 08:28:11 on ttys000
clca:~ admin$ cd /Users/admin/Library/Android/sdk/extras/microsoft/AstoriaSDK/tools
clca:tools admin$ ./wconnect 192.168.1.2
Creating session...

Please enter the pin for the device to be paired and connected: 1234
emulator-5554 on 192.168.1.2connected.
clca:tools admin$
```

After **wconnect** succeeds in establishing a connection, the phone appears as an emulator in the list of devices shown by running the **adb devices** command. If **wconnect** reports “Connected on 127.0.0.1:xxxx”, the device will appear in the list as **emulator-xxxxy**, where $y=x-1$. For example, in the following screenshot, the device appears as **emulator-5554**.



```
F:\wconnect
> adb devices
List of devices attached
emulator-5554 device
```

5555 means it's emulator-5554, see below

Deploy using adb

After a connection is available, you can deploy your app using the `adb install` command. To install an APK on your machine, do the following.

PC:

1. Open a command prompt, and change the directory to the platform-tools directory in your Android SDK path.

```
cd %LOCALAPPDATA%\Android\sdk\platform-tools
```

2. Run the following command:

```
adb install myapp.apk
```

Mac:

3. Open a command prompt, and change the directory to the platform-tools directory in your Android SDK path.

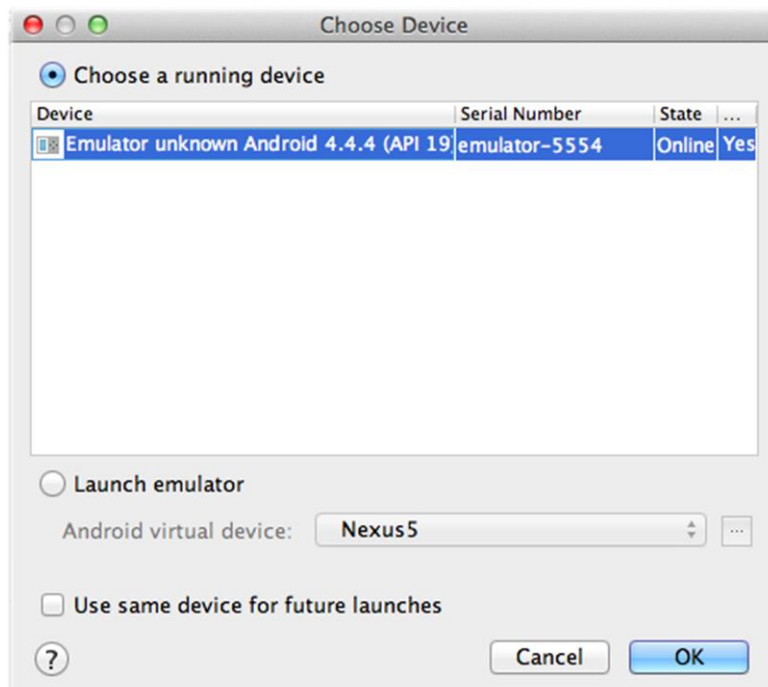
```
cd ~/Library/Android/sdk/platform-tools
```

4. Run the following command:

```
./adb install myapp.apk
```

Deploy and run your app from the IDE

Alternatively, you can deploy and debug your app using your IDE. From your IDE, you should see a device connected to adb. Select this device and deploy/debug as you normally would.




Configure your app to use Microsoft services instead of Google Play services

Because Google Play Services are not available on Windows devices, Microsoft provides an interop library that maps your existing service API calls to Microsoft services. In most cases, you can update your app to use this library by adding a configuration file to your project, without changing any code.

The following sections provide the following instructions:


- How to add a Windows product flavor to your project so that it builds Android and Windows versions of the APK:
 - The Android APK uses Google Play Services, and you can continue to submit this version to the Google Play Store.
 - The Windows APK uses Microsoft services, and you can submit this version to the Windows Store.
- How to add the required configuration file for Microsoft services to your project.
- How to configure your project to use specific Microsoft services.

 **Note:** The guidance in this section will expand as additional services become available through the interop library.

Add a Windows product flavor that uses the Microsoft Services interop library

Follow these steps to configure your project so that it builds a Windows version of the APK that uses the Microsoft Services interop library, along with the Android version of the APK.

1. Open the project in your IDE of choice and open the build.gradle file for the app module (the app\build.gradle file in the project).

 **Note:** Gradle changes are only required in the app module. The build.gradle file in the root of the project does not need changes.

2. Add the following text to the top of the build.gradle file. Make sure that you revise the URI path to include the path of the “Project Astoria” SDK on your computer.

PC:

```
buildscript {
    repositories {
        maven {
            url uri('C:\\Users\\<user>\\AppData\\Local\\Android\\sdk\\extras\\Microsoft\\
                AstoriaSDK\\gradlePlugin')
        }
    }
    dependencies {
        classpath 'com.microsoft.services:astoriaPlugin:1.0'
    }
}
apply plugin: 'com.microsoft.services'
```


Mac:

```
buildscript {
    repositories {
        maven {
            url uri('/Users/admin/Library/Android/sdk/extras/Microsoft/AstoriaSDK/
                gradlePlugin')
        }
    }
    dependencies {
        classpath 'com.microsoft.services:astoriaPlugin:1.0'
    }
}
apply plugin: 'com.microsoft.services'
```

3. In the **android** section of the build.gradle file, add a **productFlavors** section that declares Windows and Android product flavors.

```
android {
    ...
    productFlavors {
        windows { }
        android { }
    }
}
```

4. In the **dependencies** section of the build.gradle file, comment out the dependency on Google Play Services. In the next step, you will update the Android product flavor to include the Google Play Services dependency.

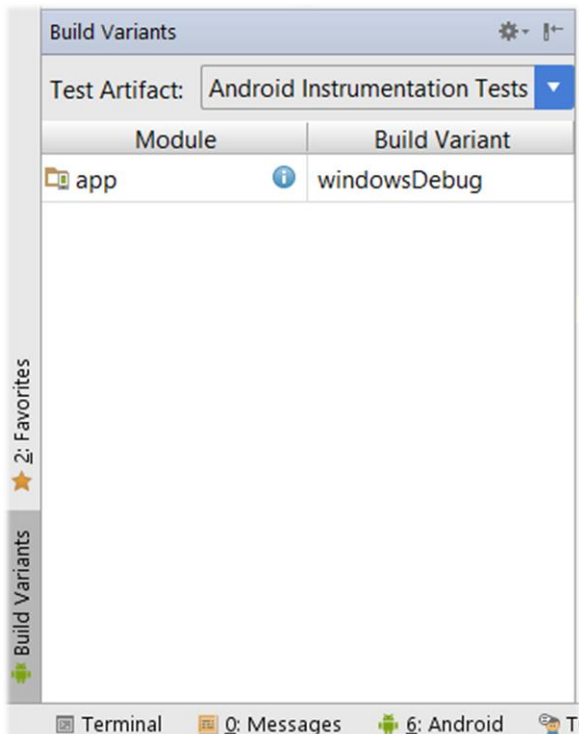
```
dependencies {
    //compile 'com.google.android.gms:play-services:6.5.87'
    ...
}
```

5. In the **dependencies** section of the build.gradle file, add the following new compile statements. These statements enable the Android product flavor to depend on Google Play Services and the Windows product flavor to depend on Microsoft's services. Update the version of Google Play Services to match the line that you commented out in the previous step. The Microsoft Services interop library version must be 6.5.87.

```
dependencies {
    ...
    androidCompile 'com.google.android.gms:play-services:6.5.87'
    windowsCompile 'com.microsoft.services:interop:6.5.87'
}
```

6. Save your changes to the build.gradle file and synchronize the file (right click the build.gradle file and choose **Synchronize 'build.gradle'**).

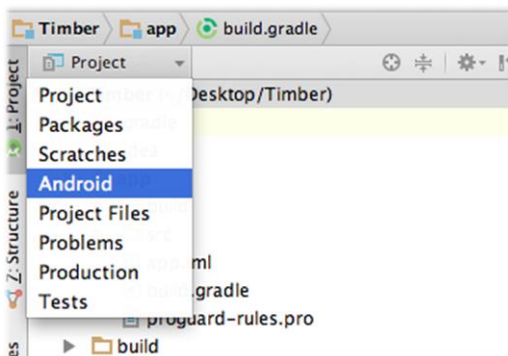
7. Open the **Build Variants** window and change the active build variant to **windowsDebug**. To open the **Build Variants** window, click **View > Tool Windows > Build Variants**.



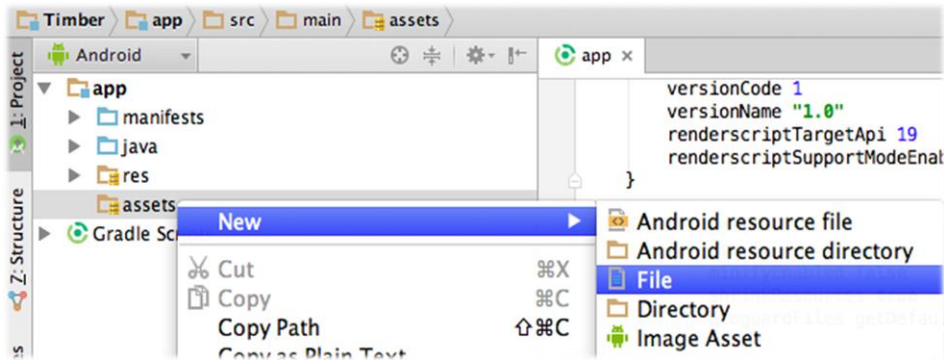
Add the configuration file for Microsoft services

After adding a Windows product flavor to your project that uses the Microsoft Services interop library, follow these steps to add a configuration file to your project that you can use to configure Microsoft services.

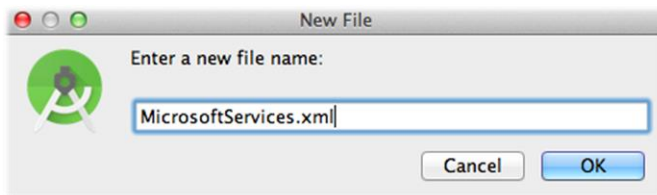
1. In your project, change the view to **Android**.



2. Under **app**, right click on **assets** and choose **New > File**.



3. Name the file **MicrosoftServices.xml**



4. Add the following text to the MicrosoftServices.xml file. When you configure specific services by following the instructions in later sections, you will add <service> elements to the <servicesConfig> element to configure specific services.

```
<?xml version="1.0" encoding="utf-8"?>
<servicesConfig>
</servicesConfig>
```

Configure specific services

The Microsoft Services interop library provides a solution for developers using Google Play Services to run on Windows 10 Mobile using Microsoft Services. Code modifications are minimal, but you will require keys for certain Microsoft services. If you have not already, create a MicrosoftServices.xml file as described in [Add the configuration file for Microsoft services](#).

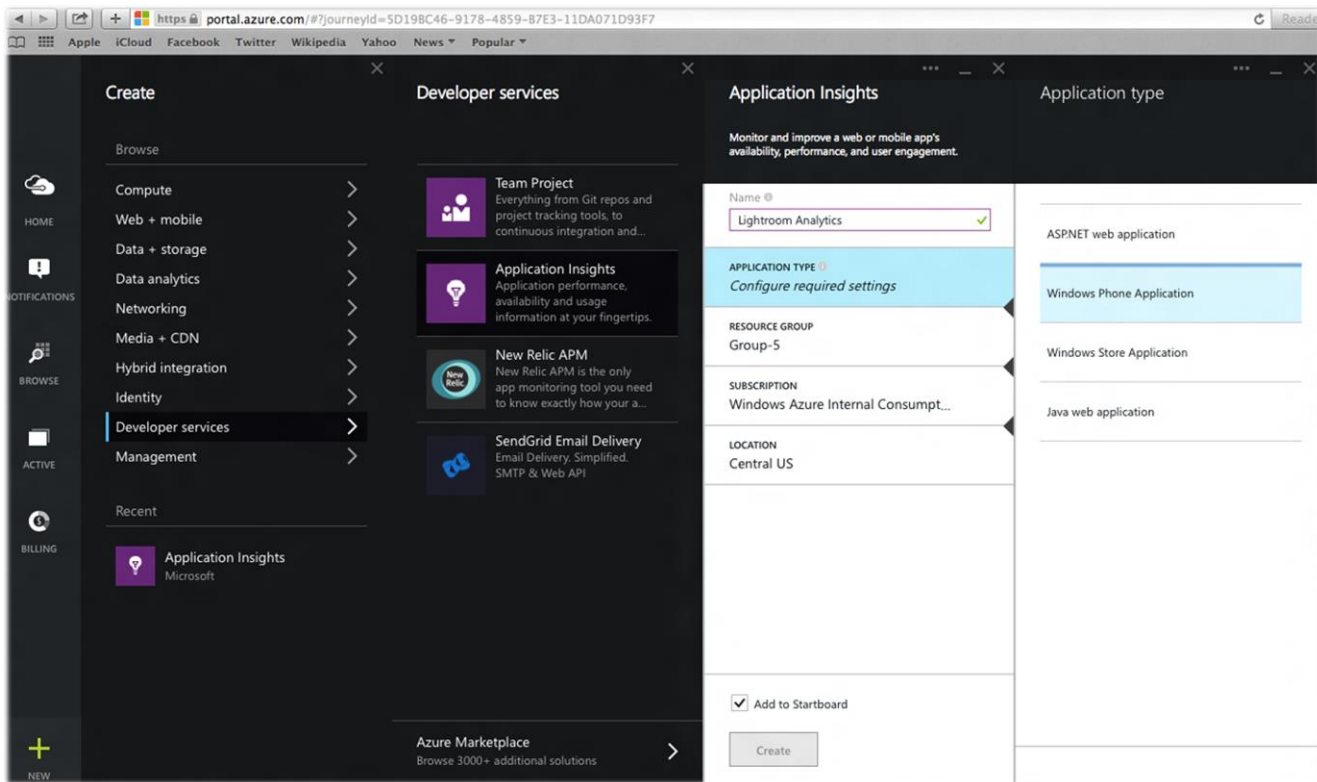
The following steps use the sample provided inside the samples folder of the “Project Astoria” SDK as an example.

Configuring Analytics

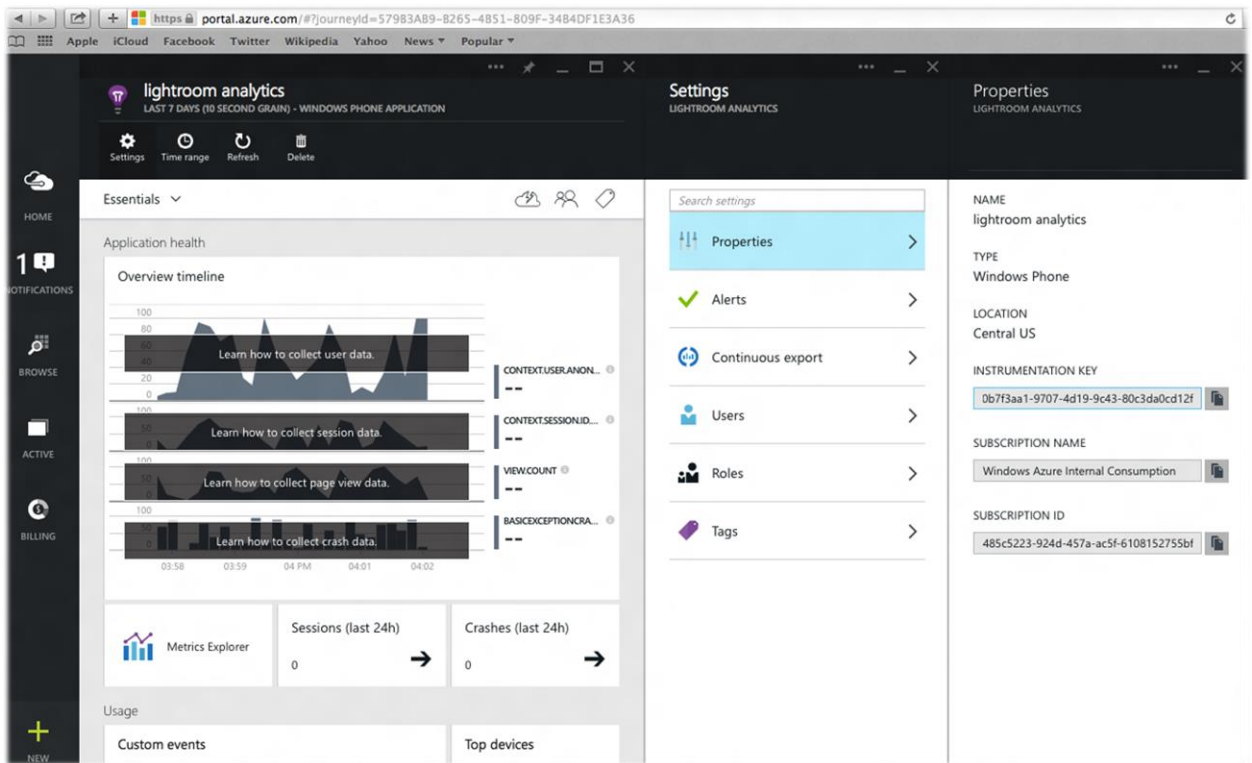
If your app uses the Google Play Analytics service, follow these steps to configure your app to instead use analytics provided by Azure Application Insights.

1. Begin by signing up for a free Windows Azure account at <http://azure.microsoft.com/en-us/pricing/free-trial/> if you don’t already have one.
2. Go to <https://portal.azure.com> to create a new Application Insights service.
 - a. Choose **NEW > Developer Services > Application Insights**.
 - b. Give the service a **Name**.
 - c. Select **Windows Phone Application** application type.

d. Tap **Create**.



- Choose **Settings > Properties** and copy the **Instrumentation key** from the right-hand pane.



- In the `MicrosoftServices.xml` file you previously added to your project, add a new `<service>` element with the content shown below. Paste the instrumentation key into the `<key>` value. Optionally, if you are using metrics and dimensions without a name, you can supply a friendly name in this configuration file. Otherwise, they will appear with a generic name in the Application Insights dashboard.

```
<?xml version="1.0" encoding="utf-8"?>
<servicesConfig>
  <service name="analytics">
    <key>your instrumentation key goes here</key>
    <customDimensions>
      <customDimension index="1" name="Custom dimension 1" />
      <customDimension index="2" name="Custom dimension 2" />
    </customDimensions>
    <customMetrics>
      <customMetric index="1" name="Custom metric 1" />
      <customMetric index="2" name="Custom metric 2" />
    </customMetrics>
  </service>
</servicesConfig>
```

- Rebuild your app.

Configuring Maps

If your app uses the Google Play Maps service, follow these steps to configure your app to use Bing Maps instead.


1. Go to the Bing Maps Dev Center at <https://www.bingmapsportal.com>
 - a. If you have a Bing Maps account sign in with the Microsoft account that you used to create the account, otherwise create a new account. For new accounts, follow the instructions in [Creating a Bing Maps Account](#) on MSDN.
2. Now that your account is created, login to the Bing Maps Dev Center and select **Keys** under **My Account**.
3. Provide the following information to create a key:
 - a. **Application name**: Required. The name of the application.
 - b. **Application URL**: The URL of the application.
 - c. **Key type**: Required. Select the key type that you want to create. You can find descriptions of key and application types [here](#).
 - d. **Application type**: Required. Select the application type that best represents the application that will use this key. You can find descriptions of key and application types [here](#).
4. Type the characters of the security code, and then click **Create**. The new key displays in the list of available keys.
5. In the MicrosoftServices.xml file you previously added to your project, add a new <service> element as a child of the <servicesConfig> element. Add a name attribute with the value "maps", and a child <key> element that contains the Bing Maps key.

```
<?xml version="1.0" encoding="utf-8"?>
<servicesConfig>
  <service name="maps">
    <key>your Bing Maps key goes here</key>
  </service>
</servicesConfig>
```

6. Rebuild your app.

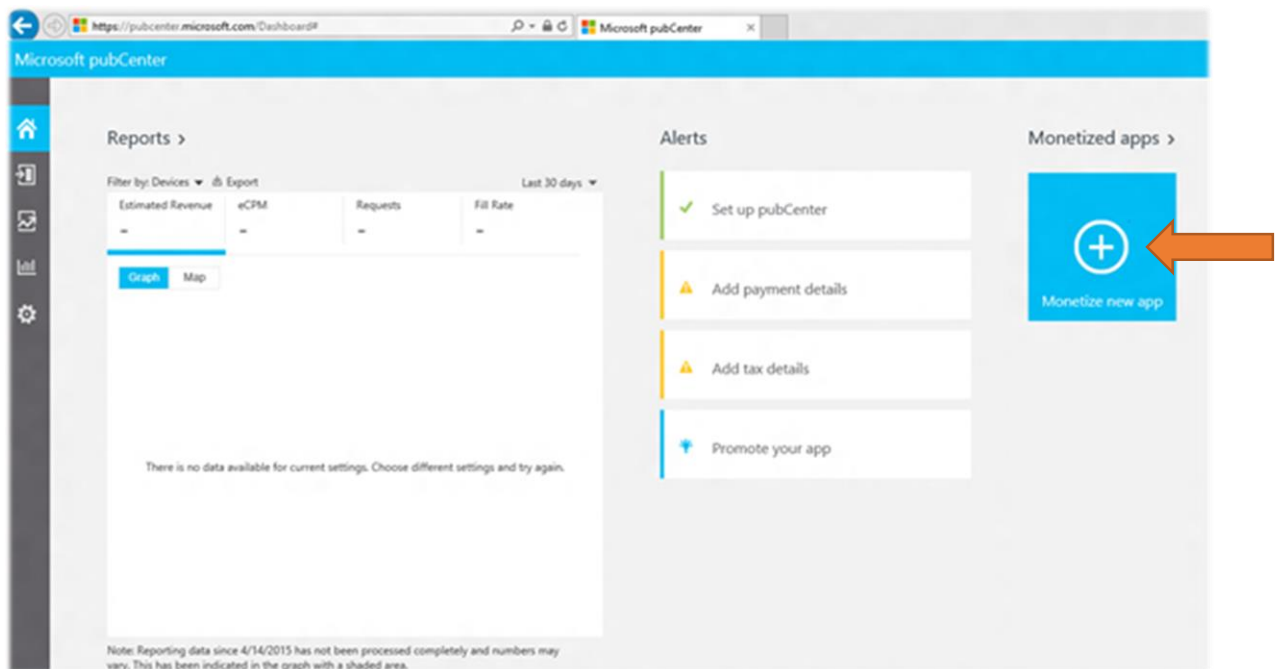
Configuring Ads (without Ad Network Mediation)

If your app uses the Google Mobile Ads service, follow these steps to configure your app to use the Microsoft Advertising (Ads in Apps) service instead.

 **Important Note:** *Interstitial/video ad formats are supported starting with the July TAP release of the “Project Astoria” SDK.*

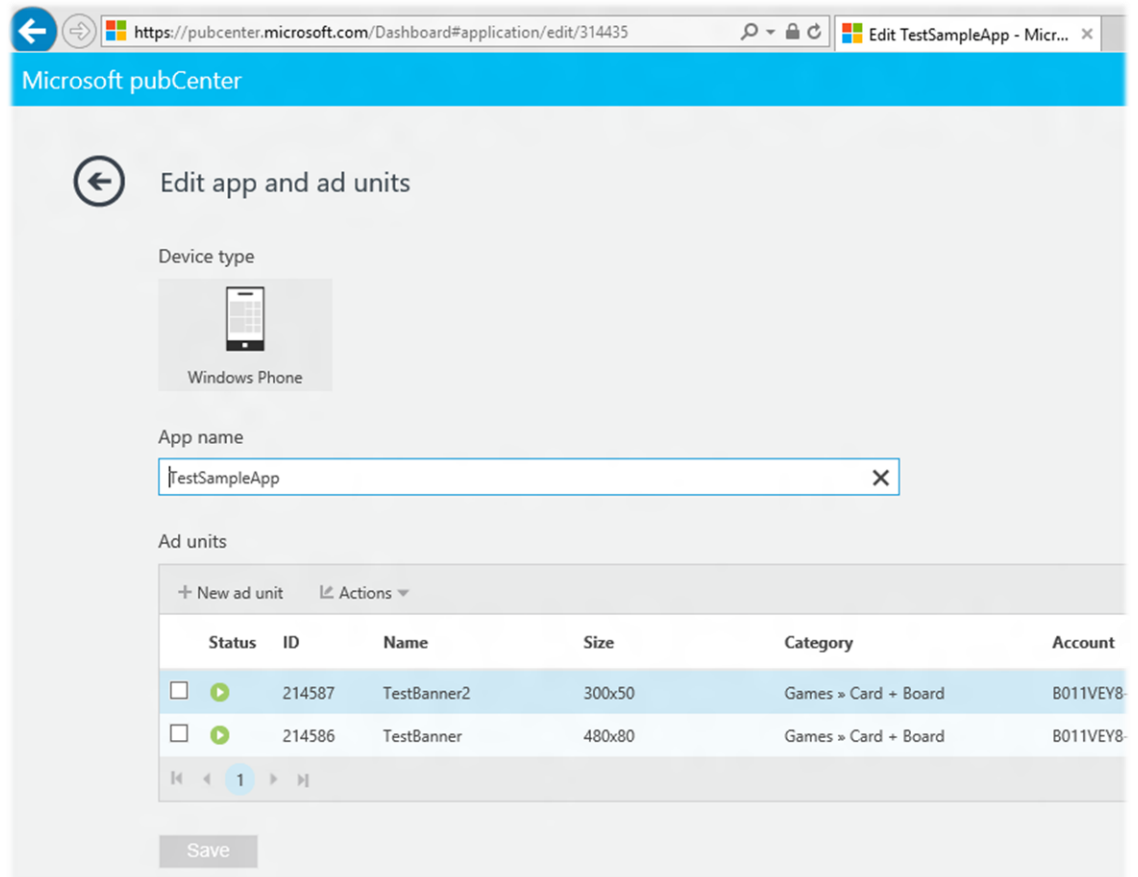
 **Important Note:** *Interoperability with Search Ads for Mobile is not supported.*

1. Sign up for a Microsoft pubCenter account:
 - a. Go to the pubCenter home page at <https://pubcenter.microsoft.com/Login>.
 - b. If you already have a Microsoft Account, you can use it to create an account. Otherwise, follow the steps to create a Microsoft Account and then a pubCenter account.
 - c. For more information about using pubCenter, see <http://adsinapps.microsoft.com/en-us/pubcenter>
2. After your account is created, log-in to pubCenter and click the **Monetize new app** button.



3. Select **Windows Phone** as the device type.
4. Type a friendly name for your app in the **App name** edit box.
5. For each AdMob or DoubleClick for Publishers ad unit ID that is used by your app for requesting ads, you must create a corresponding Microsoft Advertising ad unit ID.
 - a. Create the Microsoft Advertising ad unit IDs in the **Ad units** section (you must specify a name, size and category for each requested ad unit).
 - b. Click **Save**.

Example:



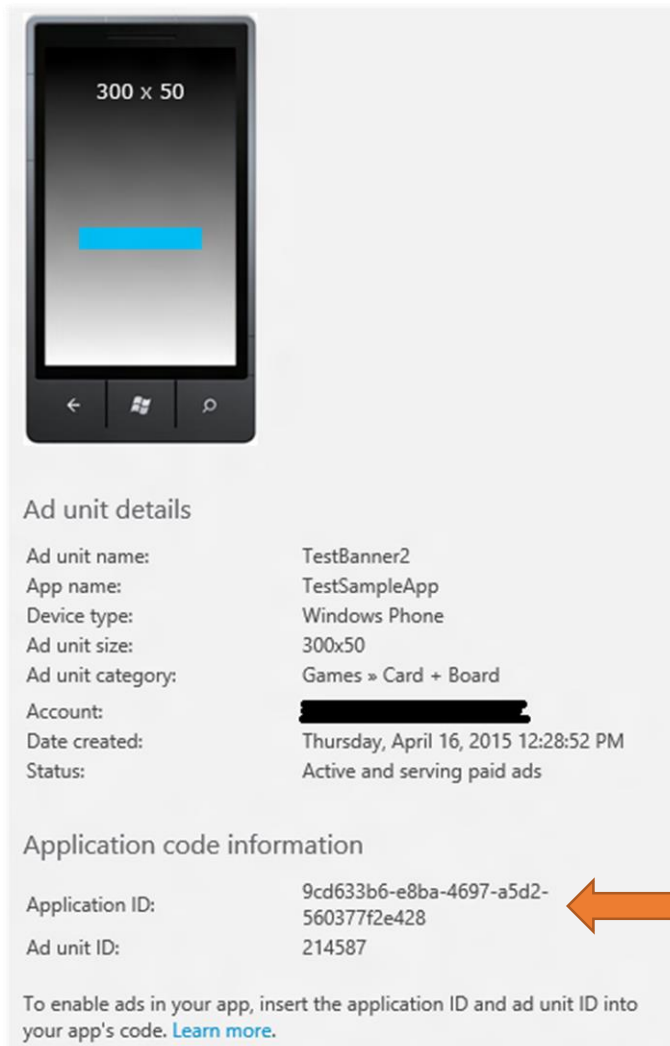
6. The Microsoft Advertising ad unit IDs for each requested ad unit are listed in the **ID** column. Make note of these IDs – you will need to enter them into your project's `MicrosoftServices.xml` file in a later step.

Ad units

+ New ad unit Actions

Status	ID	Name	Size	Category
<input type="checkbox"/>	214587	TestBanner2	300x50	Games » Card + Board
<input type="checkbox"/>	214586	TestBanner	480x80	Games » Card + Board

7. Also note the Application ID for your app (the Application ID is not the same as the Publisher ID). You will also need to enter this in the `MicrosoftServices.xml` file in a later step.



8. Click **Done**.
9. Be sure to also add **Payment Details** and **Tax Details** to your pubCenter account.
10. In your AndroidManifest.xml, add a meta tag with your ApplicationId:

```
<meta-data android:name="com.microsoft.windows.applicationid"
  android:value="af48cd02-40e7-407f-b5ad-394ab4bb8052" />
```

11. In your project, open `MicrosoftServices.xml` and add a service section for Ads:

```
<service name="ads">
  <googleMobileAds>
    <!-- OPTIONAL: Specify the expected eCPM value of AdMob / DFP
      (Default value is '1' for each ad network) -->
    <targetEcpm>target-ecpm-value</targetEcpm>

    <!-- Specify ad unit mapping for each AdMob/DFP ad unit ID in your app -->
    <adUnitMapping>
      <googleAdUnitId>google-ad-unit-id#1</googleAdUnitId>
      <msAdUnitId>microsoft-ad-unit-id#1</msAdUnitId>
    </adUnitMapping>

    <adUnitMapping>
      <googleAdUnitId>google-ad-unit-id#2</googleAdUnitId>
      <msAdUnitId>microsoft-ad-unit-id#2</msAdUnitId>
    </adUnitMapping>
  </googleMobileAds>
</service>
```

12. For each Google Mobile Ads ad unit ID in your app, do the following:

- a. In the `<googleMobileAds>` section, add another `<adUnitMapping>` block with the sub-tags `<googleAdUnitId>` and `<msAdUnitId>`:


```
<adUnitMapping>
  <googleAdUnitId>google-ad-unit-id</googleAdUnitId>
  <msAdUnitId>microsoft-ad-unit-id</msAdUnitId>
</adUnitMapping>
```

- b. Replace `google-ad-unit-id` with the appropriate Google Mobile Ads ad unit ID
- c. Replace `microsoft-ad-unit-id` with the appropriate Microsoft Advertising ad unit ID

13. Rebuild your app.

Configuring Ads (with Ad Network Mediation)

If your app uses an AdMob mediation adapter from Millennial Media, InMobi, Flurry, AdColony or MobFox, follow these steps to enable interoperability between these adapters and the Microsoft Advertising service in your app.

 **Important Note:** Microsoft intends to build interoperability with additional AdMob mediation adapters in a future release.

1. Follow all the steps in the previous section, [Configuring Ads \(Without Ad Network Mediation\)](#).
2. Open `MicrosoftServices.xml` and make the following edits in your project:
 - a. In the **Ads** key section, add a `<mediationAdapters>` element after the `<googleMobileAds>` element:

```
<mediationAdapters>
</mediationAdapters>
```

- b. If your project depends on a **MobFox**, **Millennial Media**, or **InMobi** mediation adapter, do the following:

- i. Add the following lines inside the <mediators> element for each adapter you depend on. Then edit the highlighted strings per the instructions below.

```
<mediator mediatorName="name">
  <targetEcpm>target-ecpm-value</targetEcpm>
  <mediatorKey mediationId="pubIdNumber">id-key</mediatorKey>
</mediator>
```

- ii. In the **mediatorname** attribute, replace *name* with **MobFox**, **Millenial**, or **InMobi**.
 - iii. Replace *target-ecpm-value* with a target eCPM value for this ad network. This value will be used by the mediation framework to prioritize ad fills from ad networks with higher eCPM. If you are unsure what the target eCPM value should be, set this to '1'.
 - iv. Replace *id-key* with your MobFox publisher ID, Millenial Media publisher ID, or InMobi App ID.
- c. If your project depends on an **AdColony** mediation adapter, do the following:
 - i. Add the following lines inside the <mediators> element:

```
<mediator mediatorName="AdColony">
  <targetEcpm>target-ecpm-value</targetEcpm>
  <mediatorKey mediationId="appId">adcolony-app-id-key</mediatorKey>
  <mediatorKey mediationId="zone_id">adcolony-zone-id</mediatorKey>
</mediator>
```

- ii. Replace *target-ecpm-value* with a target eCPM value for this ad network. This value will be used by the mediation framework to prioritize ad fills from ad networks with higher eCPM. If you are unsure what the target eCPM value should be, set this to '1'.
 - iii. Replace *adcolony-app-id-key* with your AdColony App ID.
 - iv. Replace *adcolony-zone-id-key* with your AdColony Zone ID.
- d. If your project depends on a **Flurry** mediation adapter, do the following:
 - i. Add the following lines inside the <mediators> element:

```
<mediator mediatorName="Flurry">
  <targetEcpm>target-ecpm-value</targetEcpm>
  <mediatorKey mediationId="projectApiKey">flurry-api-
key</mediatorKey>
  <mediatorkey
mediationId="adSpaceName">IAB_BANNER_ANDROID</mediatorKey>
</mediator>
```

- ii. Replace *target-ecpm-value* with a target eCPM value for this ad network. This value will be used by the mediation framework to prioritize ad fills from ad networks with higher eCPM. If you are unsure what the target eCPM value should be, set this to '1'.
- iii. Replace *flurry-api-key* with your Flurry API key.

Example:

```
<service name="ads">

  <googleMobileAds>
    <!-- OPTIONAL: Specify the expected eCPM value of AdMob / DFP
      (Default value is '1' for each ad network) -->
    <targetEcpm>target-ecpm-value</targetEcpm>

    <!-- Specify ad unit mapping for each AdMob/DFP ad unit ID in your app -->
    <adUnitMapping>
      <googleAdUnitId>google-ad-unit-id#1</googleAdUnitId>
      <msAdUnitId>microsoft-ad-unit-id#1</msAdUnitId>
    </adUnitMapping>

    <adUnitMapping>
      <googleAdUnitId>google-ad-unit-id#2</googleAdUnitId>
      <msAdUnitId>microsoft-ad-unit-id#2</msAdUnitId>
    </adUnitMapping>
  </googleMobileAds>

  <mediationAdapters>
    <mediator mediatorName="MobFox">
      <targetECPM>target-ecpm-value</targetEcpm>
      <mediatorKey mediationId="pubIdNumber">mobfox-mediation-id-key</mediatorKey>
    </mediator>

    <mediator mediatorName="Millennial">
      <targetEcpm>target-ecpm-value</targetEcpm>
      <mediatorKey mediationId="apid">millenial-mediation-id-key</mediatorKey>
    </mediator>

    <mediator mediatorName="InMobi">
      <targetEcpm>target-ecpm-value</targetEcpm>
      <mediatorKey mediationId="appId">inmobi-app-id-key</mediatorKey>
    </mediator>

    <mediator mediatorName="AdColony">
      <targetEcpm>target-ecpm-value</targetEcpm>
      <mediatorKey mediationId="appId">adcolony-app-id-key</mediatorKey>
      <mediatorKey mediationId="zone_id">adcolony-zone-id-key</mediatorKey>
    </mediator>

    <mediator mediatorName="Flurry">
      <targetEcpm>target-ecpm-value</targetEcpm>
      <mediatorKey mediationId="projectApiKey">flurry-api-key</mediatorKey>
      <mediatorKey mediationId="adSpaceName">IAB_BANNER_ANDROID</mediatorKey>
    </mediator>
  </mediationAdapters>
</service>
```

3. Rebuild your app.

Configuring Cloud Messaging

If your app uses the Google Cloud Messaging service, follow these steps to configure your app server to use the Windows Notification Service instead.

1. Go to <http://dev.windows.com>
2. Select **Dashboard**
3. Select **Windows Phone Store**
4. Select **Submit App**
5. Select **App Info**
6. On the App Info page, fill out **App name** and other App information.
7. Click **Reserve Name** and wait for the Application ID to populate.
8. Then scroll down to **More Options** and open the section by clicking the arrows
9. Now go to the Windows Push Notifications section and click the "[here](#)" link.

Windows Push Notifications (WNS)

Everything you need to enable push notifications for your 8.1 app is [here](#). [Learn more](#) about Windows Push Notifications service.

10. The Windows Push Notification Service app settings will appear on <https://account.live.com/developers/applications/appsettings/<ClientID>>.

The screenshot shows the Microsoft account Developer Center interface. At the top, there's a navigation bar with "Home", "My apps", "Docs", "Downloads", and "Support". Below this, the breadcrumb trail reads "My applications > MyTestWNSApp > App Settings". The main heading is "MyTestWNSApp". On the left, a sidebar lists "Settings" with sub-items: "Basic Information", "API Settings", "App Settings" (highlighted with a blue box), and "Localization". The main content area is titled "To protect your app's security, Windows Push Notification Services (WNS) and services using Microsoft account use client secrets to authenticate the communications from your server." It contains several sections: "Package SID" with a long alphanumeric string and a note "This is the unique identifier for your Windows Store app."; "Application identity" with XML-like tags for Name, Publisher, and Application ID, and a note "To set your application's identity values manually, open the AppManifest.xml file in a text editor and set these attributes of the <identity> element using the values shown here."; "Client ID" with a string and a note "This is a unique identifier for your application."; "Client secret" with a string and a note "For security purposes, don't share your client secret with anyone." Below these is a note about creating a new client secret if compromised, followed by a "Create a new client secret" link. At the bottom, there's a "Note" about waiting 24 hours before activating a new client secret. The footer includes the Microsoft logo, copyright notice "© 2015 Microsoft. All rights reserved.", and links for "Terms of use", "Trademarks", "Privacy & Cookies", "Site Feedback", and "United States (English)".

11. In your app server, make the following changes:
 - a. The registration ID generated by apps that are recompiled for “Project Astoria” have a “Microsoft” prefix. Because of this, you must update your app server to send the POST request for the registration ID to <https://global.notify.windows.com/v2/send> rather than <https://android.googleapis.com/gcm/send>. The API request and response parameters, format, and behavior are otherwise the same as GCM.
 - b. Replace your GCM authorization key with a new string made up of the Package SID and Client secret obtained from the Windows Push Notification Service app settings page. The new string should use the following format “<Package SID>;<Client secret>”. Note that the two keys are separated by a semicolon.
12. In the AndroidManifest.xml file for your app project, add the following <meta-data> elements as children of the <application> element. Replace the placeholder strings with the app identity name and publisher obtained from the Windows Push Notification Service app settings page and the publisher display name.

```
<meta-data
    android:name="Windows.package.identity.name"
    android:value="<identity name>" />
<meta-data
    android:name="Windows.package.identity.publisher"
    android:value="<identity publisher>" />
<meta-data
    android:name="Windows.publisher.display.name"
    android:value="<publisher display name>" />
```

Here is an example:

```
<meta-data
    android:name="Windows.package.identity.name"
    android:value="9686MyCreative.MyTestWNSApp" />
<meta-data
    android:name="Windows.package.identity.publisher"
    android:value="CN=6056FA26-F6F7-4C3C-9A12-A02A263F92C8" />
<meta-data
    android:name="Windows.publisher.display.name"
    android:value="Contoso" />
```

Configuring Location Services

No configuration is required for apps using Google Play Location APIs. Calls to Location and Geofencing service APIs should work after you follow the instructions in [Add a Windows product flavor that uses the Microsoft Services interop library](#) and then rebuild your project.

Platform differences between Android and “Project Astoria”

This section describes ways in which the “Project Astoria” app platform differs from Android.

Activity stack management

In general, “Project Astoria” manages activities and their associated intents the same way they are handled by Android. Activities that users interact with when performing a certain job are organized into a *task*, and the task contains a *back stack* of activities that operates as a “last in, first out” structure.

However, the current release of “Project Astoria” has one major difference: all activities belong to a back stack within a single task. This is different from Android, where activities can belong to different tasks, each with their own back stack. This section highlights how this difference affects different activity launch mode scenarios.

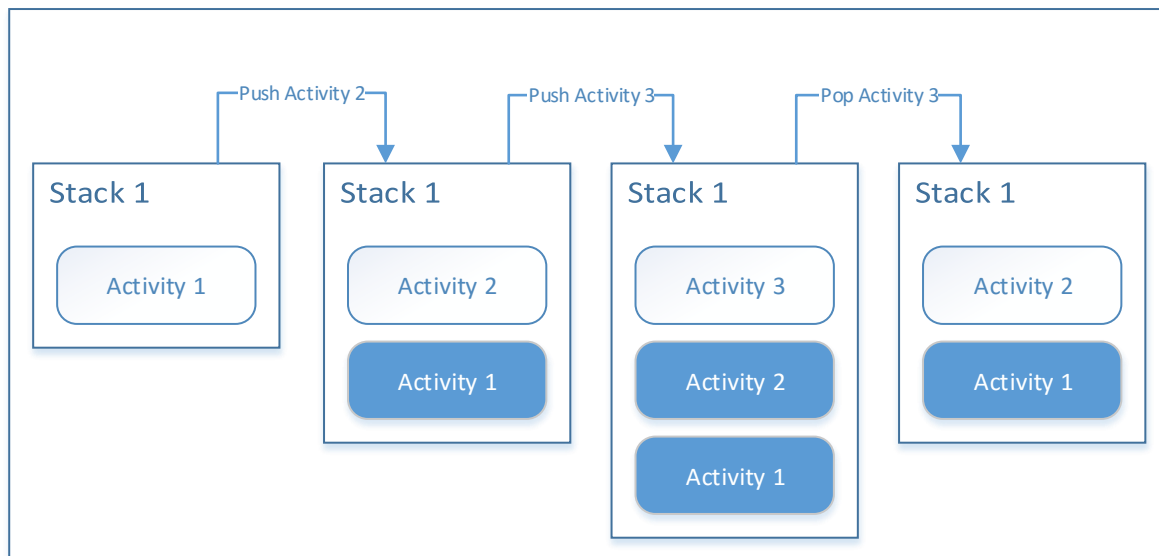
For more information about tasks in Android, see [Tasks and Back Stack](#).

Standard behavior

When an activity is started with the *standard* launch mode, there is no difference in behavior between “Project Astoria” and Android. In this scenario, the system simply creates a new instance of the activity in the task that started it and puts this activity at the top of the task’s back stack. The standard launch mode is used in any of the following cases:

- The `<activity>` element’s `launchMode` attribute is not specified or it is set to “standard”.
- No intent flags are passed to `startActivity()`.

The following diagram illustrates this scenario.



Single top behavior

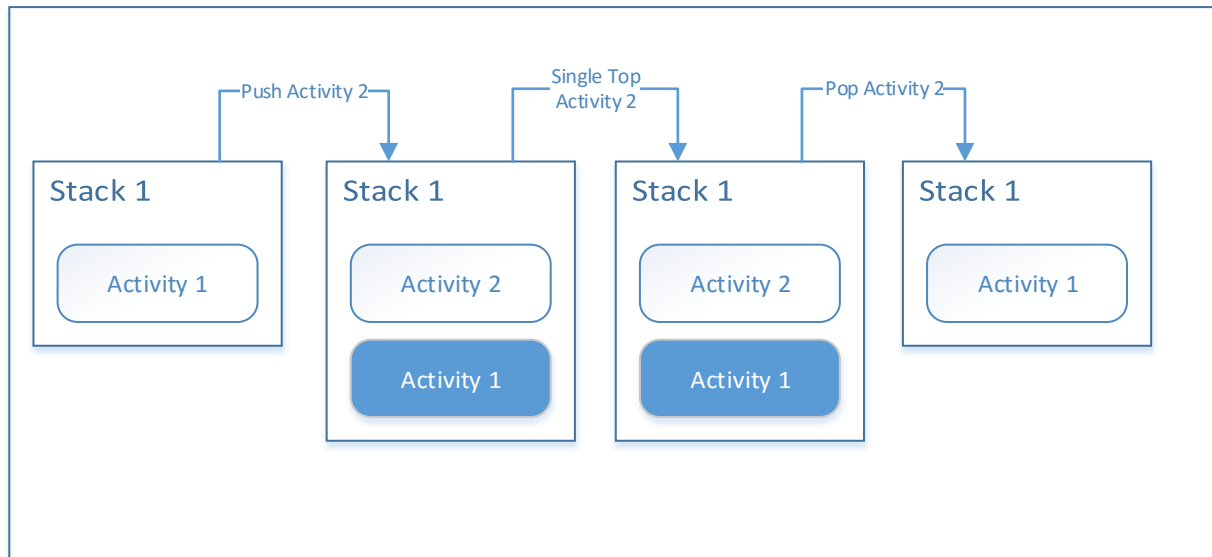
There is also no difference in behavior between “Project Astoria” and Android when an activity is started with the *single top* launch mode. In this scenario, the system does the following:

- If an instance of the specified activity already exists at the top of the task’s back stack, the system routes the intent to the existing activity.
- Otherwise, the system creates a new instance of the activity and moves this new instance to the top of the task’s stack.

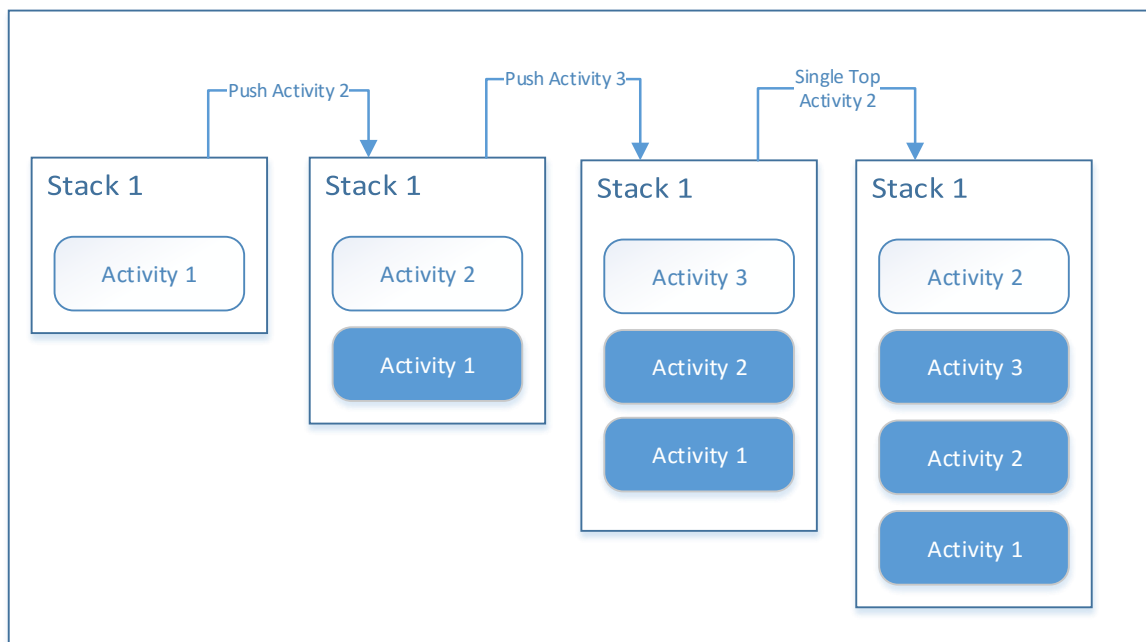
The single top launch mode is used in any of the following cases:

- The <activity> element’s `launchMode` attribute is set to “singleTop”.
- `FLAG_ACTIVITY_SINGLE_TOP` is passed to `startActivity()`.

The following diagram illustrates single top launch mode when the activity (in this case, “Activity 2”) already exists at the top of the task’s back stack.



The following diagram illustrates single top launch mode when the activity (in this case, “Activity 3”) does not already exist at the top of the task’s back stack.

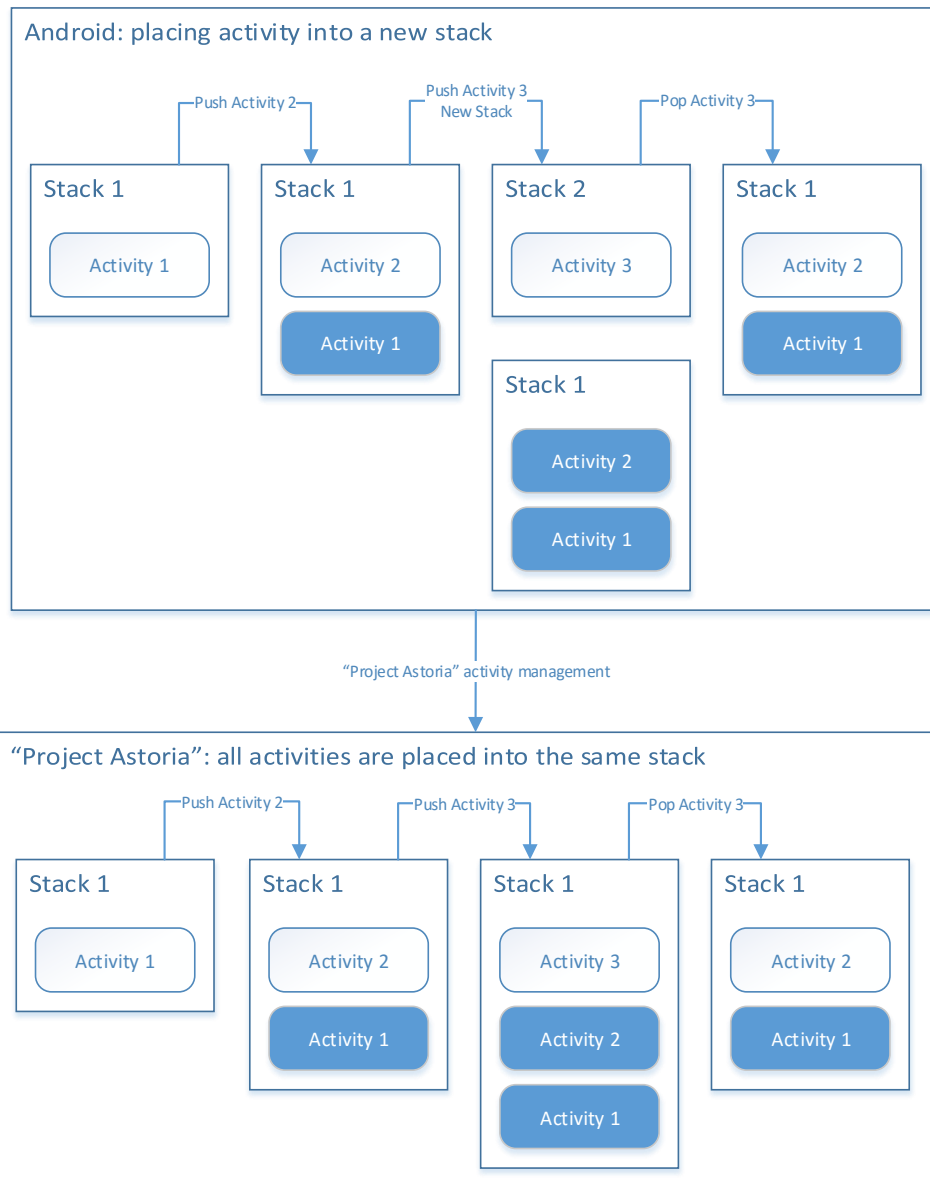


New task behavior

“Project Astoria” places all activities in the same back stack within a single task. Therefore, if your code does any of the following operations, be aware that on “Project Astoria” the new activity will be added to the current task’s back stack rather than a different task’s back stack.

- The <activity> element’s launchMode attribute is set to “singleTask” or “singleInstance”.
- FLAG_ACTIVITY_CLEAR_TOP or FLAG_ACTIVITY_NEW_TASK are passed to startActivity().

The following diagram illustrates how the behavior of this scenario differs between Android and “Project Astoria”. Note that if the end user is simply stepping through the activity back stack without switching between tasks, there is no difference in the end user experience between Android and “Project Astoria”.



Clear top behavior

When an activity is started with the *clear top* launch mode, the behavior in “Project Astoria” may diverge from Android.

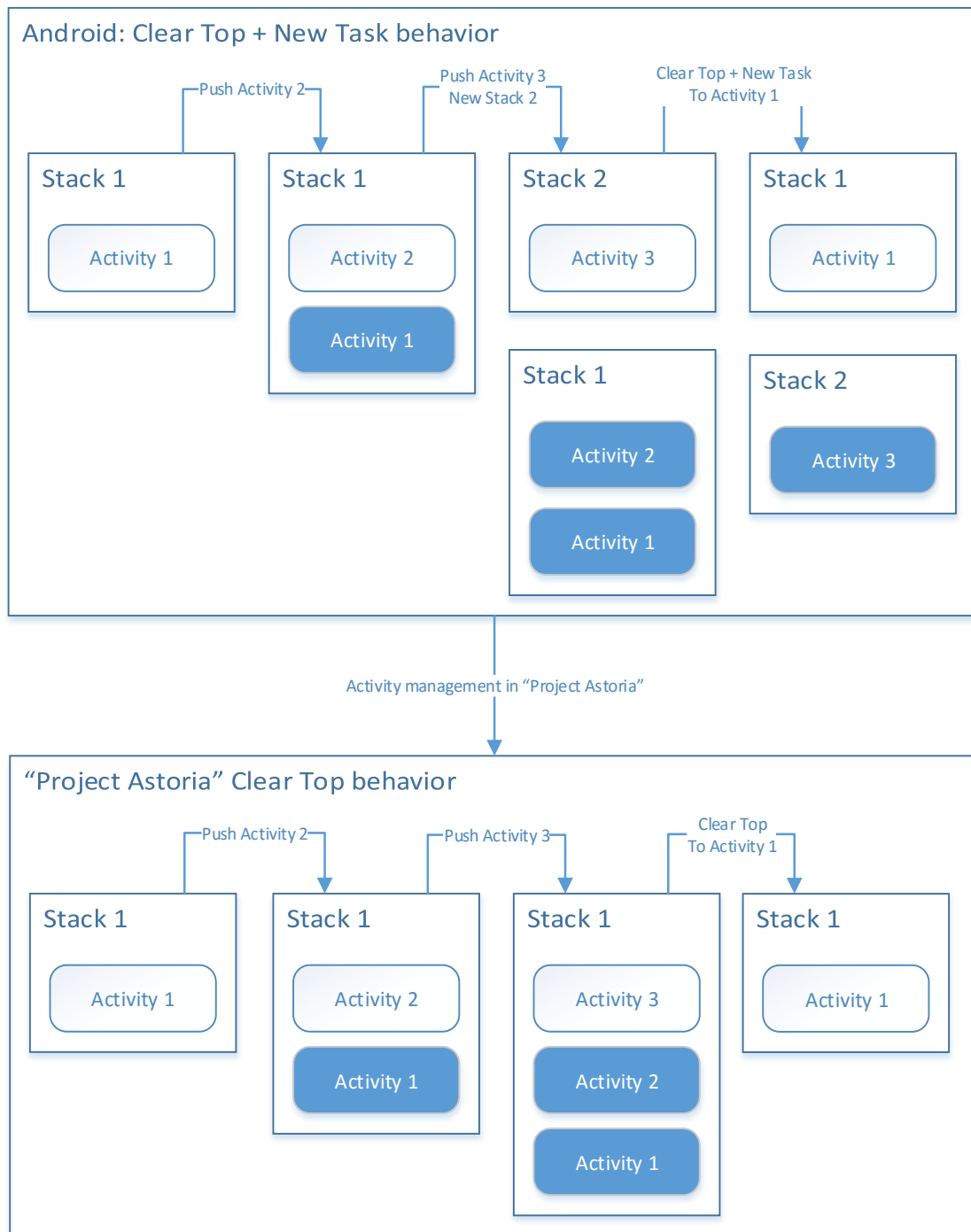
In this launch mode, the system determines whether the specified activity is already running in the targeted task. If so, the system destroys any other activities on top of the specified activity, resumes the activity, and routes the intent to it.

If your code starts the activity within the current task, the behavior in “Project Astoria” is the same as Android. However, if your code starts the activity in a different task, be aware that on “Project Astoria” the new activity will be started within the current task and any activities above it in the same back stack will be destroyed. This is because “Project Astoria” places all activities in the same back stack within a single task.

You will encounter this scenario in either of the following cases:

- `FLAG_ACTIVITY_CLEAR_TOP` and `FLAG_ACTIVITY_NEW_TASK` are passed to `startActivity()`.
- `FLAG_ACTIVITY_CLEAR_TOP` is passed to `startActivity()` and the `<activity>` element’s `launchMode` attribute is set to “singleTask”.

The following diagram illustrates how the behavior of this scenario differs between Android and “Project Astoria”.



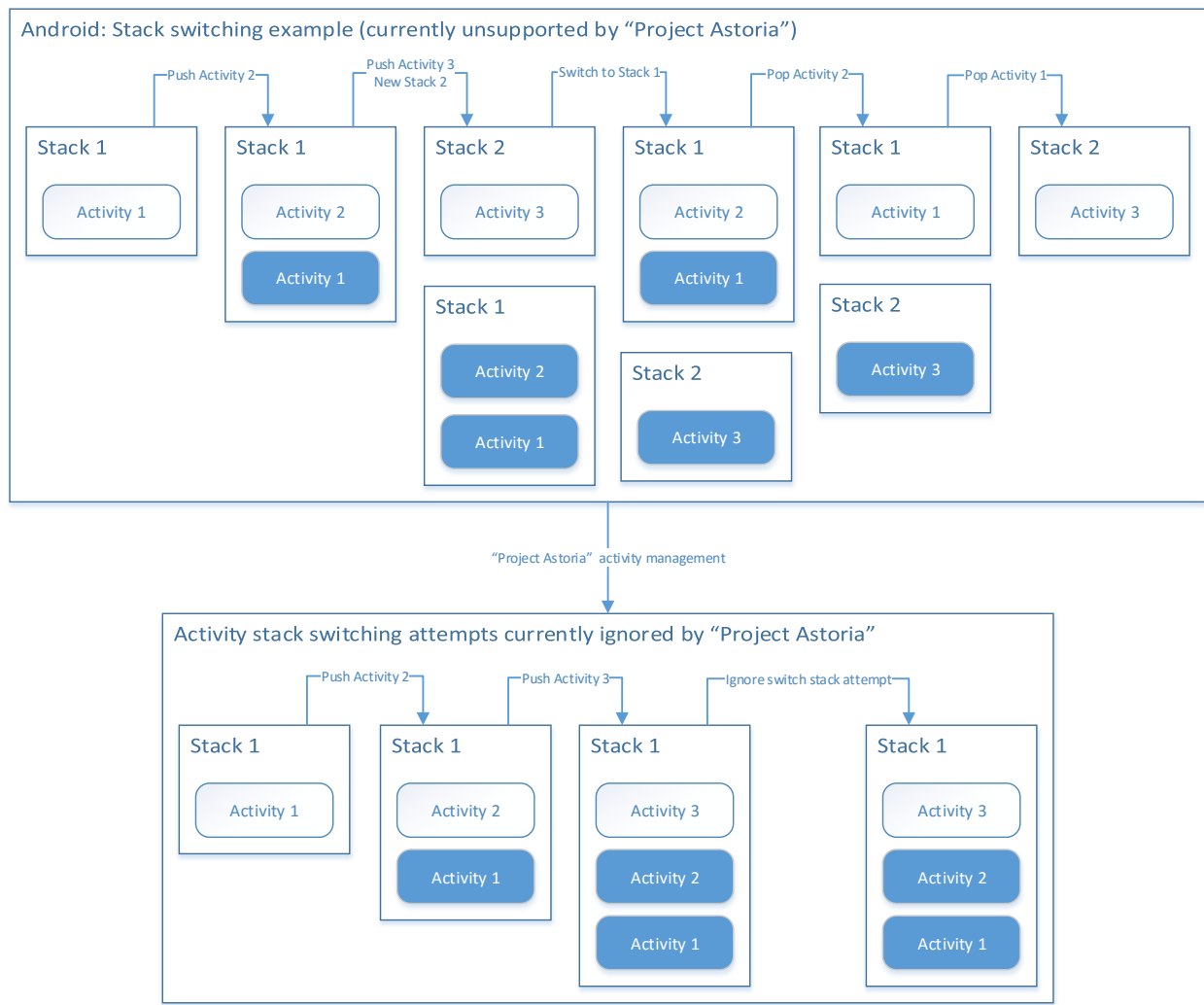
Switching between activity stacks

In Android, when there are multiple tasks that contain activities, it is possible to switch between those tasks by defining the expected behavior in the `<activity>` element's `launchMode` attribute or by using specific intent flags that you pass to `startActivity()`. In the current release of "Project Astoria", all activities are placed in the same task, even when it would normally result in the creation of a new task in Android. Because of this, switching to another task's back stack is not supported the current release of "Project Astoria".

Switching to another task's back stack can occur in either of the following cases when the activity being launched already exists in a different task:

- The `<activity>` element's `launchMode` attribute is set to "singleTask" or "singleInstance".
- `FLAG_ACTIVITY_NEW_TASK` is passed to `startActivity()`.

The following diagram illustrates how the behavior of this scenario differs between Android and “Project Astoria”.



Appendix A: Troubleshooting

Trouble connecting to the device?

- Confirm the following, and then try reconnecting to the device again.
 - Make sure the screen is on and the device is unlocked. We recommend that you increase the time after which the screen times out and a password is required to the maximum values. To change these options on the phone, open **Settings**, tap **Personalization**, tap **Lock screen**, and tap **Advanced settings**.
 - Make sure ports 22 and 50505 are allowed by your computer's firewall policy. The **wconnect** program uses these ports to connect to the device.
 - Make sure your computer's firewall settings allow wconnectsrv to access the network.
- If none of the previous options fixed the problem, try the following procedure:
 1. Using the **wconnect** program available in the **tools** directory of the "Project Astoria" SDK, do the following to kill any existing connection.

From a PC:

- a. Open a command prompt, and change the directory to the extras\Microsoft\AstoriaSDK\tools folder in your Android SDK path. The default Android SDK path is %LOCALAPPDATA%\Android\sdk.

```
cd %LOCALAPPDATA%\Android\sdk\extras\Microsoft\AstoriaSDK\tools
```

- b. Start **wconnect** with the **kill-server** parameter:

```
wconnect.exe kill-server
```

From a Mac:

- a. Open a terminal window. Change directory to the extras/microsoft/AstoriaSDK/tools folder in your Android SDK.

```
cd ~/Library/Android/sdk/extras/Microsoft/AstoriaSDK/tools
```

- a. Start **wconnect** with the **kill-server** parameter:

```
./wconnect.exe kill-server
```

2. Reconnect to the device by using the `wconnect <ip address>` command. For more information, see [Connect using wconnect](#).

3. Test the connection.

From a PC:

- a. Open a command prompt, and change the directory to the platform-tools directory in your Android SDK path.

```
cd %LOCALAPPDATA%\Android\sdk\platform-tools
```

- b. Run the following command:

```
adb shell ls
```

From a Mac:

- a. Open a command prompt, and change the directory to the platform-tools directory in your Android SDK path.

```
cd ~/Library/Android/sdk/platform-tools
```

- b. Run the following command:


```
./adb shell ls
```

4. If the connection still is not working, reboot the phone and try steps 1-3 again.

Appendix B: wconnect tool reference

Use the **wconnect** tool to establish a connection between your development computer and the Windows 10 Mobile device where you want to deploy or debug your app. This tool is provided in the **tools** directory of the “Project Astoria” SDK. After you create a device connection using **wconnect**, you can use adb commands or your IDE to perform tasks such as installing and debugging your app.

For instructions about how to use **wconnect** to establish a connection between your computer and a device, see [Connect to your phone by using the wconnect tool](#).

 **Important Note:** To establish a connection to a Windows 10 Mobile device, you must use the **wconnect** tool. The `adb connect` command cannot be used to establish a connection.

wconnect command line options

The **wconnect** tool currently supports the following command line arguments.

Argument	Description
<IP address>	Establishes a connection to the device with the specified IP address.
start-server	Checks whether the wconnect server process (wconnectsrv) is running, and starts the process if it is not running.
kill-server	Terminates the wconnect server process (wconnectsrv).
--noadb <IP address>	Do not use these arguments. These arguments are currently intended for internal use only.
--install <.appx path>	
--uri <URIToLaunch>	

wconnect errors

The **wconnect** tool returns errors in the format “<error message>. Error code = <code>”. The following table describes some of the most common errors.

Error	Description/resolution
Error connecting to ADBD. Error code = -7	Try rebooting the device.
Error bootstrapping the device. Error code = -8	Make sure you are using the latest version of the “Project Astoria” SDK with the device.
Error connecting to the device. Error code = -9	If connecting over USB from a PC, make sure to install the PC pre-requisites using PC – Install pre-requisites for the wconnect tool . From “Add Remove Programs” in the Control Panel, you should see an entry called “Windows Phone IP over USB”. Make sure the device is unlocked and in developer mode, and Device discovery is set to On . For more information, see Appendix D: Enabling developer mode on the phone .

Appendix C: Using adb commands with “Project Astoria”

The “Project Astoria” SDK includes the Android Debug Bridge (adb) tool, which Android developers use to communicate with a device to perform tasks such as installing apps and copying files to the device. The adb tool in the “Project Astoria” SDK supports most of the commands that Android developers are familiar with. However, be aware of the following limitations:

- The `adb connect` command cannot be used to establish a connection to a Windows 10 Mobile device. You must use the **wconnect** tool instead.
- Using the remote shell is not supported. Instead, use the specific shell command for your task. For example, if you want to start an Activity, use the `adb shell am start <INTENT>` command directly instead of running `adb shell` and starting the Activity from the remote shell.

adb errors

The table below describes common errors that you may encounter by running one of the following commands:

- `adb install`
- `adb shell pm install`
- `adb shell pm uninstall`

Error	Description/resolution
Failure [MANIFEST_DECODER_ERROR] Failure [CONVERTER_ERROR]	Make sure you have compiled your APK using the “Project Astoria” SDK.
Failure [OUT_OF_DISK_SPACE]	Try freeing up space on the device by uninstalling apps or deleting media such as pictures and music.
Failure [SCREEN_LOCKED]	Unlock the device.
Failure [PACKAGE_INSTALL_ERROR] Failure [PACKAGE_INSTALL_REGISTRATION_ERROR]	This error indicates one of the following: <ul style="list-style-type: none">• Your APK has not been injected/recompiled using the “Project Astoria” SDK.• The device is in a bad state and must be hard reset from Settings, System, About, Reset your phone.
Failure [PACKAGE_UNINSTALL_ERROR]	The device is in a bad state. Try rebooting the device.
Failure [PACKAGE_NOT_FOUND]	The specified package does not exist.
Failure [PACKAGE_INVALID_NAME]	The specified package has an invalid name.
Failure [DEVICE_NOT_DEVELOPER_UNLOCKED]	Make sure the device is in developer mode. For instructions, see Appendix D: Enabling developer mode on the phone .

Failure [EXCEEDED_APP_DEPLOY_LIMIT]	Too many apps have been side loaded. Try uninstalling a side loaded application.
--	--

The following table describes common errors that you may encounter by running the `adb shell am start` command.

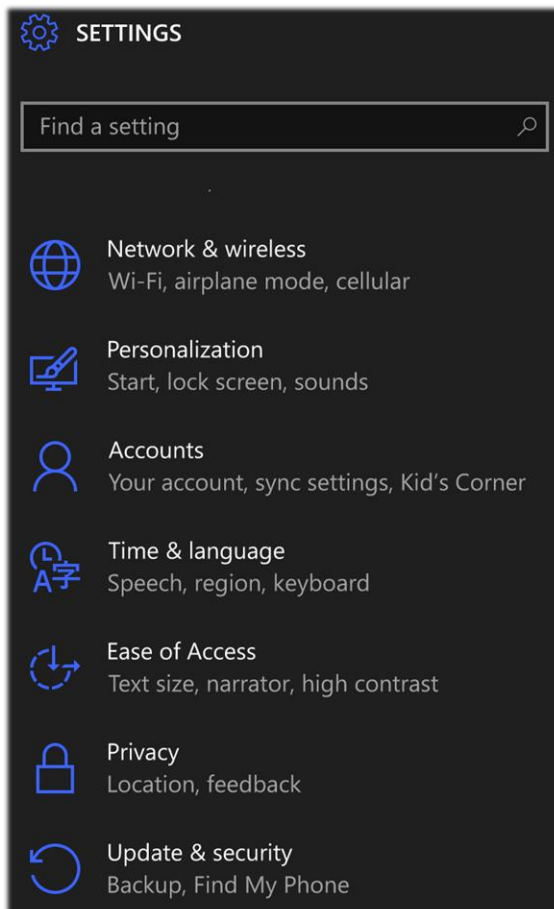
Error	Description/resolution
Failure [INTENT_NOT_SPECIFIED]	No intent was specified.
Failure [INTENT_NOT_SUPPORTED]	The specified type of intent is not currently supported. Only implicit intents are supported.
Failure [INTENT_START_ERROR]	There was an error while starting the intent.

Appendix D: Enabling developer mode on the phone

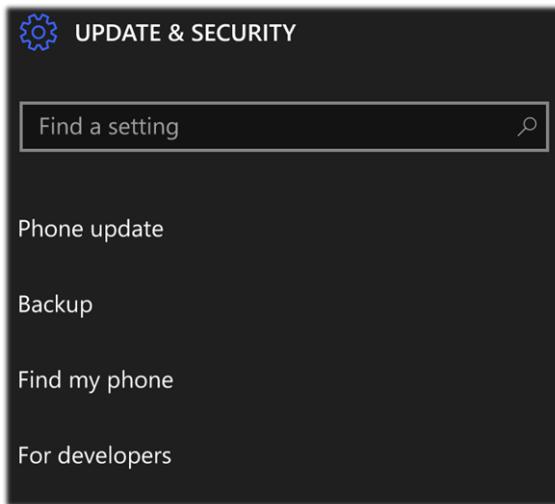
Before you can establish a connection between your development computer and phone, you must first enable developer mode on the phone. For more information about establishing a device connection, see [Connect to your phone by using the wconnect tool](#).

To enable developer mode:

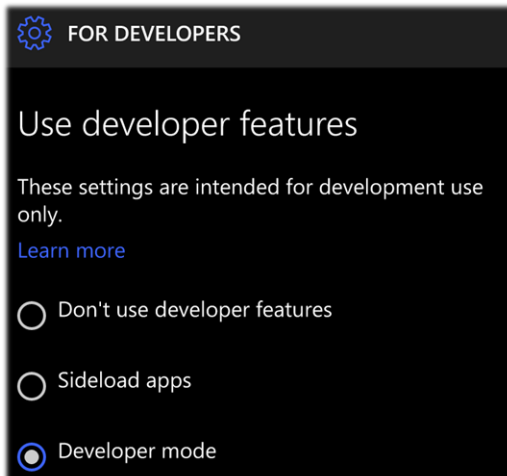
1. Under the **Settings** menu, choose **Update & security**:



2. Choose **For developers**:



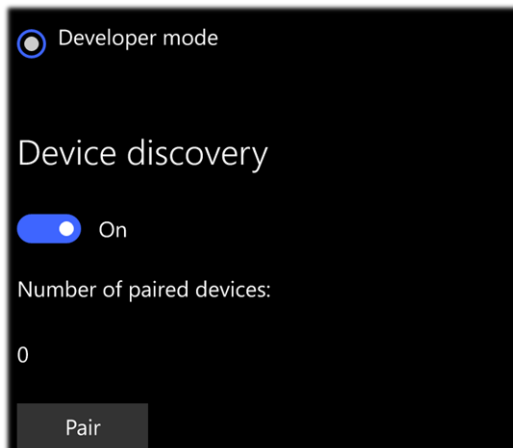
3. Choose **Developer mode**:



4. In the pop-up message, tap **Yes**.



5. Set the toggle button under **Device discovery** to **On** and then tap **Pair**.



6. You will be given a pairing code. Keep this screen open until you have completed the steps to connect to your phone in [Connect to your phone by using the wconnect tool](#).

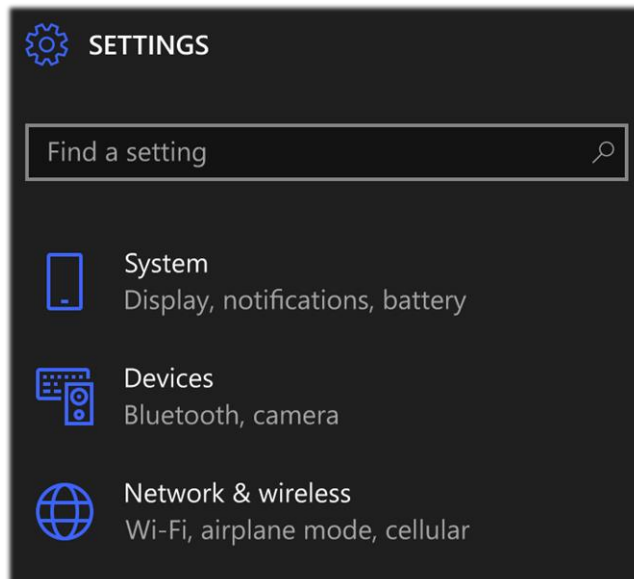


Appendix E: Getting the IP address of the phone

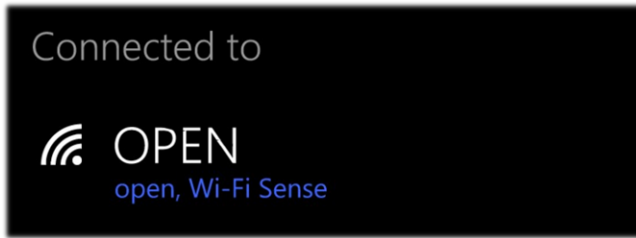
Before you can establish a connection between your development computer and phone via Wi-Fi, you must first retrieve the IP address of the phone. For more information about establishing a device connection via Wi-Fi, see [To connect to your phone over Wi-Fi \(Mac or PC\)](#).

To get the IP address of the phone:

1. Open **Settings** and tap **Network & wireless**.



2. Tap **Wi-Fi**.
3. Tap the network that you are connected to.



4. The IP address is available on the **Edit network** page.

