**Configuring the Setup Wizard**

The Android Setup Wizard (SUW) is the component that connects a new device to the Internet and enables the user to sign in or create a Google account. OEMs and carriers can customize the SUW to incorporate branding and add device setup and account configuration steps.

Android安装向导（SUW）是将新设备连接到互联网的组件，并允许用户登录或创建一个Google帐户。oem厂商和运营商可以定制SUW来整合品牌，并添加设备设置和账户配置步骤。

**NOTE:** Please find the document or resource files marked as red line under SUW Ref folder.

请在SUW Ref文件夹下找到标记为红线的文档或资源文件。

Android SUW look anYou can also:

* Use the Wizard Library to easily create individual screens that harmonize with the d feel.
* Use the Wizard Manager to easily create flows of screens.
* Review design guidelines for SUW screens.

Android SUW看起来你也可以：

使用向导库轻松创建与d感觉协调的独立屏幕。

使用向导管理器来轻松创建屏幕流。

检查设计的SUW屏幕设计指南。

For help creating SUW screens in other apps, use the Setup Wizard Library User Guide.

为了帮助在其他应用程序中创建SUW屏幕，请使用安装向导库用户指南。

**NOTE:** The content on this page reflects the current setup wizard instructions as of Android 7.x.

这个页面上的内容反映了当前安装向导的指令，如Android 7.x。

* For Android 5.1 and 6.x, refer to Customizing the Setup Wizard in Android M
* For Android 5.0, refer to Customizing the Setup Wizard in Android L
* [Overview](#_Overview)
* [Customizing SUW](#_Customizing_SUW)
* [Customizing the Wizard Script](#_Customizing_the_Wizard)
* [Google Assistant](#_Google_Assistant)
* [Configuring Optional Steps](#_Configuring_Optional_Steps)
* [Enterprise](#_Enterprise)
* [Deferred Setup Wizard](#_Deferred_Setup_Wizard)

# Overview

The default Setup Wizard contains the following components:

| **Component** | **Description** |
| --- | --- |
| Language + Accessibility selection | New in Android 7.0: richer A11Y settings for users with vision impairments |
| Restore/Setup fresh choice | New in Android 7.0, designed to help guide users through correct setup path |
| SIM & cellular service activation |  |
| Wi-Fi connection |  |
| Factory Reset Protection | Prevents reuse of a device that has been reset in an untrusted manner. |
| Mandatory OTA |  |
| Early Update | Mechanism for Google Play to automatically update itself before continuing within the Setup Wizard. |
| Tap & Go | Enables users to set up their new device and trigger account transfer from their old device. |
| Google Authentication | Sign into or sign up for a Google account. |
| Date and time settings |  |
| User name settings |  |
| Restoration from backup |  |
| Play Auto Installs | Enables OEMs or carriers to customize initial home screens and install apps immediately after setup. (Recently added: a screen to allow users to opt out of non-mandatory PAI apps) |
| Google opt-ins and Terms of Service (ToS) |  |

Android 7.0 introduced an Optional Steps menu, a final screen at the end of Setup Wizard. Anything non-essential for device setup or legal reasons lives here, and partners can control the contents and ordering of this menu. Google provides the following steps; partners can add their own.

Android 8.0 added additional improvements:

Android 8.0增加了额外的改进：

* SUW is now direct boot aware, allowing for a smoother transition between bootstrap animations and the Welcome screen
* WizardScript v2 simplifies scripts and makes them easier to maintain
* A new system property controls how Wi-Fi is enabled when leaving SUW

SUW现在是直接引导意识，允许引导动画和欢迎屏幕之间的平滑过渡。

向导的v2简化了脚本，使它们更易于维护

一个新的系统属性控制了在离开SUW时如何启用Wi-Fi

Android 8.1 adds [Deferred Setup](#_Deferred_Setup_Wizard), an optional second stage which allows less-critical features to be set up later at the user’s convenience, allowing the first setup stage (First Setup) to be shortened to just the most critical features (such as phone activation). First Setup will often be performed by a sales person at a retail outlet, while Deferred Setup is always intended for use by the phone’s new owner.

Android 8.1添加了延迟设置，这是一个可选的第二阶段，它允许在用户方便的时候设置不太重要的功能，允许第一个设置阶段（第一个设置）被缩短为最关键的特性（比如电话激活）。第一个设置通常是由一个销售人员在一个零售商店执行，而延迟设置总是用于手机的新主人的使用。

## Requirements

Use of the Setup Wizard carries the following requirements:

安装向导的使用有以下要求：

* In all Android SUWs, Google account sign-in/up MUST immediately follow connection to the network.
  + Exception: you may add a system image OTA download, or terms and conditions screen, after network connection, if and only if these do not involve any account sign-in/sign up process, or similar.
* To avoid user confusion, the branding of carrier, OEM, or third-party screens MUST be distinct from Google/Android branding.
* Users MUST NOT be able to leave SUW by any means without:
  + being offered the ability to set up Factory Reset Protection; and
  + before being authenticated as part of Factory Reset Protection; and
  + before having the opportunity to see Google Terms & Conditions and Privacy consents on the Google Services screen.

在所有的Android SUWs中，Google账户的签到必须立即跟随连接到网络。

oException：您可以添加一个系统图像在线下载，或术语和条件屏幕，在网络连接之后，如果且仅当这些不涉及任何帐户签名/注册过程，或类似的情况下。

为了避免用户混淆，运营商、OEM或第三方屏幕的品牌必须与谷歌/android品牌不同。

用户不能以任何方式离开SUW：

o存在提供了设置工厂重置保护的能力;

在被认证为工厂重置保护的一部分;

在谷歌服务的屏幕上，oge有机会看到谷歌的条款、条件和隐私。

It is **required** that users not escape SUW before having seen the terms and conditions on the Google Services screen. System navigation and access to notifications or the open web should be disabled throughout SUW to prevent escapes and misconfigured devices. **This is especially important for devices which ship with Factory Reset Protection.** Such devices mustn't allow users to e.g. launch Chrome from web-based help shipping with a keyboard component; trigger a search using a keyboard or third-party headphone component; etc. **If a user can escape Setup Wizard using any method, Factory Reset Protection can be bypassed.**

在看到Google服务屏幕上的条款和条件之前，用户不需要转义SUW。系统导航和对通知或开放web的访问应该在整个SUW中禁用，以防止转义和错误配置的设备。这对于使用工厂重置保护的设备来说尤其重要。这样的设备不允许用户使用键盘组件从基于网络的帮助发货中启动Chrome;使用键盘或第三方耳机组件触发搜索;如果用户可以使用任何方法来转义安装向导，那么可以绕过工厂重置保护。

### Play Auto Installs (PAI)

If your device installs additional apps using Play Auto Installs, one of the following must be satisfied:

* In setup wizard, add a stand-alone screen (or an item in the list of optional steps) that **notifies** users which apps are automatically installed and **allows** users to choose which optional apps are installed.
* All apps are automatically installed an placed on the home screen.

## Design Principles

Consider these principles as you apply your own creativity and design thinking. Deviate with purpose.

当你运用你自己的创造力和设计思维时，考虑这些原则。偏离了目标。

1. Delight me in surprising ways. A beautiful surface, a carefully-placed animation, or a well-timed sound effect is a joy to experience. Subtle effects contribute to a feeling of effortlessness and a sense that a powerful force is at hand.

1。以惊人的方式取悦我。一个美丽的表面，一个精心放置的动画，或者一个适时的声音效果是一种体验。微妙的影响有助于一种毫不费力的感觉，以及一种强大的力量即将到来的感觉。

1. Get to know me. Learn people's' preferences over time. Rather than asking them to make the same choices over and over, place previous choices within easy reach.
2. Let me make it mine. People love to add personal touches because it helps them feel at home and in control. Provide sensible, beautiful defaults, but also consider fun, optional customizations that don't hinder primary tasks.
3. Keep it brief. Use short phrases with simple words. People are likely to skip sentences if they're long.
4. Only show what I need when I need it. People get overwhelmed when they see too much at once. Break tasks and information into small, digestible chunks. Hide options that aren't essential at the moment, and teach people as they go.
5. I should always know where I am. Give people confidence they know their way around. Make places in your app look distinct and use transitions to show relationships among screens. Provide feedback on tasks in progress.
6. It's not my fault. Be gentle in how you prompt people to make corrections. They want to feel smart when they use your wizard. If something goes wrong, give clear recovery instructions but spare them the technical details. If you can fix it behind the scenes, even better.

7所示。这不是我的错。要温柔地提醒人们改正错误。当他们使用你的向导时，他们想要感觉自己很聪明。如果出了问题，给出明确的恢复说明，但要省去技术细节。如果你能在幕后解决这个问题，那就更好了。

### Goals

The goal of the Setup Wizard is to **give** the user a delightful and visually bold out-of-box experience that sets the tone for Android and to **get** the user into a stable state as quickly and clearly as possible. Users should not only be able to move quickly between steps but also understand what each step is asking of them and why.

安装向导的目标是给用户一个令人愉快的、直观的开箱即用体验，为Android设置基调，并尽可能快、清晰地让用户进入一个稳定的状态。用户不仅应该能够在不同的步骤之间快速移动，还应该了解每个步骤对他们的要求和原因。

A stable state can be defined as the following:

* User has selected language preference
* SIM card is active (for phones)
* User is on a Wi-Fi network
* User is signed in
* User has been offered an opportunity to secure their device
* If desired, the user has restored their new device from existing backup set
* User has accepted Google Terms of Service, Privacy Policy, and is opted-in to Google services such as Location History, and Device Backup.

一个稳定的状态可以定义如下：

用户选择了语言偏好

SIM卡是活跃的（用于手机）

用户在Wi-Fi网络上

用户登录

用户得到了一个保护他们设备的机会

如果需要，用户已经从现有的备份集中恢复了他们的新设备

用户已经接受了Google的服务条款，隐私政策，并被嵌入到Google服务中，比如位置历史和设备备份。

## Recommendations

### Design guidelines

For more details on screen layouts and recommended styles, refer to the Setup Wizard design guidelines:

有关屏幕布局和推荐样式的更多细节，请参考安装向导设计指南：

* Setup Wizard Design Guidelines (Android 8.0)
* Setup Wizard Design Guidelines (Android 7.1)
* Setup Wizard Design Guidelines (Android 7.0)
* Setup Wizard Design Guidelines (Android 6.0)

### Implementing animations

Screens using the theme @style/SuwThemeMaterial or @style/SuwThemeMaterial.Light from the Setup Wizard Library automatically inherit the slide transitions appropriate for use during setup. To manually specify a different activity transition, use the following:

* Activity#overridePendingTransition(R.anim.suw\_slide\_back\_in, R.anim.suw\_slide\_back\_out);
* Activity#overridePendingTransition(R.anim.suw\_slide\_next\_in, R.anim.suw\_slide\_next\_out);

### Implementing layouts

To give users a harmonious setup experience, customized SUWs should use the same layouts as the Android SUW screens. We strongly recommend that SUW screens be implemented using the Setup Wizard Library, which helps implement layouts while the Wizard Manager lets you gather screens together into a setup flow. OEMs may use either the light or dark SUW color schemes as needed to be consistent with their design language. This will ensure that screens are sized and laid out correctly on various device sizes.

为了给用户提供一个和谐的设置体验，定制的SUWs应该使用与Android SUW屏幕相同的布局。我们强烈建议使用安装向导库来实现SUW屏幕，这有助于实现布局，而向导管理器允许您将屏幕聚集到一个设置流程中。oem厂商可以根据需要使用浅色或深色的SUW配色方案，以符合他们的设计语言。这将确保在不同设备大小上正确地显示屏幕大小和布局。

For details, refer to the Setup Wizard Library User Guide

有关详细信息，请参见安装向导库用户指南

# Customizing SUW

Android 7.0 deprecates older methods of creating a setup experience in favor of the Wizard Manager, which launches activities from a script. Activities must be customized for Setup Wizard, including layout standards and navigation controls, with result codes returned to Wizard Manager.

Android 7.0反对使用更老的方法来创建安装体验，以支持向导管理器，向导将从脚本启动活动。必须为安装向导定制活动，包括布局标准和导航控制，结果代码返回给向导管理器。

Two basic actions control the Wizard Manager:

两个基本动作控制向导管理器：

* **LOAD**. Sent by SUW to Wizard Manager with a link to the script that will be used.
* **NEXT**. Delivers the result of the current action back to Wizard Manager, which then launches the next scripted activity.

**LOAD**。由SUW发送给向导管理器，并链接到将要使用的脚本。

**NEXT**。把当前动作的结果返回给向导管理器，然后启动下一个脚本化的活动。

For details on loading a custom script or starting the NEXT action, refer to Customizing the Setup Wizard in Android 5.x. Keep in mind that while using a custom script enables an extremely flexible level of customization, the resulting SUW implementation **must** adhere to the guidelines stated here and other contracts.

有关加载自定义脚本或启动下一个动作的详细信息，请参阅在Android 5.x中定制安装向导。请记住，虽然使用自定义脚本可以实现非常灵活的定制级别，但由此产生的SUW实现必须遵循本文所述的指导原则和其他合同。

## Declaring the Overlay APK

To customize the Wizard Script, welcome screen background, theme, and more, partners can add an additional APK in the system image that provides assets in specific extension points in the SUW.

为了自定义向导脚本，欢迎屏幕背景、主题等等，合作伙伴可以在系统映像中添加额外的APK，从而在SUW中的特定扩展点提供资产。

To declare the overlay APK, the APK declares a broadcast receiver for the intent action com.android.setupwizard.action.PARTNER\_CUSTOMIZATION. The broadcast receiver does not need to perform any action as the SUW uses it only to find the APK by asking PackageManager to resolve the intent. For this reason, the broadcast receiver must be declared in the manifest XML.

为了声明覆盖APK，APK声明了一个广播接收器，用于意图动作com.android.setupwizard.action.PARTNER\_CUSTOMIZATION. 广播接收器不需要执行任何操作，因为SUW只使用它来查找APK，请求包装器来解决这个意图。由于这个原因，广播接收器必须在manifest XML中声明。

<receiver android:name=".PartnerReceiver">

  <intent-filter>

    <action android:name="com.android.setupwizard.action.PARTNER\_CUSTOMIZATION" />

  </intent-filter>

</receiver>

For SUW to recognize the overlay APK, it must be a system package with [ApplicationInfo.FLAG\_SYSTEM](http://developer.android.com/reference/android/content/pm/ApplicationInfo.html" \l "FLAG_SYSTEM" \t "_blank).

### Providing resource assets

Setup wizard retrieves resources from the overlay APK as necessary. Resources are resolved by name, so the resource type and entry name must match exactly what setup wizard expects for the resource to be resolved. For all the overlay resources, setup wizard has a fallback that is used if the resource is not provided in the overlay package. The fallback is typically what is used in the Nexus setup wizard.

安装向导根据需要从覆盖APK中检索资源。资源是通过名称来解析的，因此资源类型和入口名称必须与安装向导所期望的资源完全匹配。对于所有的覆盖资源，安装向导有一个备用方案，如果在覆盖包中没有提供资源，则使用该备用方案。后退通常是在Nexus安装向导中使用的。

Overlayable resources include the following:

|  |
| --- |
| @array/time\_zones |
| The array of time zones, in Olson ID, to show in Date & Time screen. The time zone spinner will display the list of time zones in the order they are defined (which should typically be ordered by the GMT offset). Additionally, localLabelType and foreignLabelType annotations can be added to the strings to use a different label format. For example,  <item><annotation localLabelType="location">America/Phoenix</annotation></item> will display the time zone as "Phoenix" instead of "Mountain Standard Time". Other label types include timeZoneName (which displays "Mountain Standard Time"), and full (which displays "Mountain Standard Time (Phoenix)"). |
| @drawable/welcome\_bg |
| The background used on the welcome screen. This image will be displayed as the background of the welcome screen, using CENTER\_CROP.  在欢迎屏幕上使用的背景。这张图片将显示为欢迎屏幕的背景 |
| @raw/smartdevice\_d2d\_target\_nfc\_alternative\_video |
| [Removed from Android 7.0] (GMS Core 8.2 and later) A video for use on the Tap & Go NFC instruction page to better illustrate the NFC bump. No illustration is needed for devices with a NFC chip located on the center-back. However, for devices with other NFC chip placements, an illustration has been found to greatly improve the user experience and increase the NFC success rate.  An appropriate size for the video is 600x600dp, and the video should be an Android  一段用于Tap和Go NFC指令页面的视频，以更好地说明NFC的凸起。对于带有NFC芯片的设备，在中后卫位置上没有任何说明。然而，对于带有其他NFC芯片的设备来说，已经发现了一个可以大大改善用户体验和提高NFC成功率的例子。  视频的合适大小是600x600dp，视频应该是Android  [Supported Media Format](http://developer.android.com/guide/appendix/media-formats.html" \t "_blank) video. |
| @string/sim\_missing\_text |
| [Changed in Android 7.0] Text shown when no SIM is detected in the device. This string supports the ICU PluralFormat, with the key "sim\_slots" referring to the number of SIM slots available on the device. This string should help the user figure out how to insert their SIM into the device.  在设备中检测到没有SIM卡的情况下，Android 7.0的文本发生了变化。该字符串支持ICU PluralFormat，关键的“sim插槽”指的是设备上可用的SIM槽的数量。这个字符串应该帮助用户找到如何将他们的SIM卡插入到设备中。  Device manufacturers are encouraged to customize this string in order to provide more detailed explanation on how to insert the SIM card, for example where the SIM slot is located and if a tool is required.  鼓励设备制造商定制这个字符串，以便提供关于如何插入SIM卡的更详细的解释，例如，SIM卡槽所在的位置，以及是否需要一个工具。 |
| @string/smartdevice\_d2d\_target\_nfc\_description |
| A string to override the default description for initiating an NFC bump between an existing device and a new device on the Tap & Go NFC instruction page. The Tap & Go NFC experience is shown on most phones, and the default description has been found to perform well in user studies for phones with NFC chips located on the center-back of the device. For devices with other NFC chip placements, an alternative string can provide a better user experience and increase the NFC success rate.  一种用来覆盖默认描述的字符串，用于在现有设备和Tap&Go NFC指令页面上的新设备之间启动NFC连接。在大多数手机上都显示了Tap和Go的NFC体验，而默认的描述已经被发现在用户研究中表现良好，在设备的中端有NFC芯片的手机。对于带有其他NFC芯片的设备，另一种可选的字符串可以提供更好的用户体验，并提高NFC的成功率。 |
| @string/suggested\_actions\_uri |
| [New from Android 7.0] An Android resource URI pointing to the modified list of options steps categories, e.g. android.resource://com.my.package/raw/custom\_suggested\_actions. For details on how to customize suggested actions, see [Optional Steps](https://support.google.com/androidpartners_gms/answer/7385349). |
| @string/suggested\_actions\_deferred\_uri |
| [New from Android 8.1] An Android resource URI pointing to the modified list of options steps categories available during [Deferred Setup](#_Deferred_Setup_Wizard), e.g. android.resource://com.my.package/raw/custom\_suggested\_actions\_deferred. For details on how to customize suggested actions, see [Optional Steps](https://support.google.com/androidpartners_gms/answer/7385349). |
| @string/theme\_type |
| The theme to be used in Setup Wizard. Possible values are "material" and "material\_light". |
| @string/wizard\_script\_uri |
| An android resource URI pointing to the modified wizard script for device owner, e.g. android.resource://com.my.package/raw/custom\_wizard\_script. For details, see [Customizing the Wizard Script](https://support.google.com/androidpartners_gms/answer/7385410). |
| @string/wizard\_script\_user\_uri |
| An android resource URI pointing to the modified wizard script for secondary users such as users added via Settings > Users > Add User (e.g. android.resource://com.my.package/raw/custom\_wizard\_script\_user). For details, see [Customizing the Wizard Script](https://support.google.com/androidpartners_gms/answer/7385410). |

## System properties

In addition to the overlay mechanism, setup wizard also uses [system properties](https://android.googlesource.com/platform/frameworks/base.git/+/master/core/java/android/os/SystemProperties.java" \t "_blank) for functional customizations. These properties can be customized in the device makefile using the PRODUCT\_PROPERTY\_OVERRIDES variable.

|  |
| --- |
| ro.setupwizard.hfa\_always |
| [boolean] When set, setup wizard will always invoke HFA activation, regardless of the current reported activation state. Default value is false.  布尔值设置时，设置向导将始终调用HFA激活，不管当前报告的激活状态如何。默认值是错误的。 |
| setupwizard.logging |
| [boolean] Enables/disables anonymous logging of setup wizard usage. Default value is true.  布尔启用/禁用安装向导使用的匿名日志记录。默认值是正确的。 |
| ro.setupwizard.suppress\_d2d |
| [boolean] Suppresses device-to-device setup (e.g. Tap & Go) if true. Default value is false.  布尔抑制设备到设备的设置（例如，点击和Go）如果是真的。默认值是错误的。 |
| ro.setupwizard.suppress\_d2d\_nfc |
| [boolean] Available as of GMS core 8.2. Suppresses NFC for device-to-device setup. On phones, the instructions for device-to-device setup via Google Settings app are shown instead.  布尔可用于GMS的核心8.2。抑制设备到设备设置的NFC。在手机上，通过谷歌设置应用程序显示设备到设备设置的指令。 |
| ro.setupwizard.require\_network, ro.setupwizard.user\_req\_network |
| [string] Can be "any", "wifi" or empty string. For "wifi", setup wizard will not allow the user to skip selecting a Wi-Fi network. For "any", some form of Internet connection has to be established first (e.g. cellular data), otherwise Setup Wizard will force the user to choose a Wi-Fi network.  字符串可以是“any”、“wifi”或空字符串。对于“wifi”，安装向导将不允许用户跳过选择Wi-Fi网络。对于“any”，必须首先建立某种形式的互联网连接（例如蜂窝数据），否则安装向导将迫使用户选择Wi-Fi网络。 |
| ro.setupwizard.wifi\_on\_exit |
| [boolean] (Available since Android 8.0) Re-enable Wi-Fi at the end of setup, regardless of what the user chose as the connection method to use during setup. Default value is true.  布尔（自Android 8.0以来可用）在设置结束时重新启用Wi-Fi，不管用户选择什么作为连接方法在安装过程中使用。默认值是正确的。 |
| setupwizard.feature.deferred\_setup\_suggestion |
| [boolean] (Available since Android 8.1) Enable [Deferred Setup](#_Deferred_Setup_Wizard), giving the user a second chance to finish setting up their phone at time that’s more convenient than when the phone is first being activated (often by a sales representative). Default value is true.  布尔（自Android 8.1以来可用）支持延迟设置，让用户有第二次机会完成手机的设置，这比手机第一次被激活时更方便（通常是由销售代表）。默认值是正确的。 |

### Debug logging

For additional logging of Setup Wizard internals during development, issue the following adb command:

adb shell setprop log.tag.SetupWizard VERBOSE

### Supporting direct boot

Starting in Android 8.0, Setup Wizard is direct boot aware, which means it will start up earlier than other components in the system. If you have screens injected early into the setup flow, you may need to modify your code or the wizard script to support direct boot.

For general information about supporting direct boot, see *[Supporting Direct Boot](https://developer.android.com/training/articles/direct-boot.html)*.

In the wizard scripts for the setup flow, the action com.android.setupwizard.CHECK\_USER\_UNLOCK has been added near the start of the script. This action will block the user from proceeding in setup wizard until the user is unlocked. In general, any step that starts before CHECK\_USER\_UNLOCK will need to be direct boot aware, while steps after that have the option to not be direct boot aware. In particular, note that OEM\_PRE\_SETUP is called before the user is unlocked in the default scripts we provide.

If you have a component that runs early in the setup flow, you'll need to either:

* Make that activity and other components direct boot aware. See *[Supporting Direct Boot](https://developer.android.com/training/articles/direct-boot.html)* for instuctions.
* Move CHECK\_USER\_UNLOCK before your non-direct boot activity in the wizard script. This has the downside that the user may see a loading screen as the first thing in Setup Wizard, so we strongly encourage you to go for the option above.

# Customizing the Wizard Script

You can add new screens and/or replace existing screen by providing values for

您可以添加新的屏幕和/或替换现有的屏幕，通过提供值

@string/wizard\_script\_uri or @string/wizard\_script\_user\_uri. (refer to design specs for the appropriate places for adding and customizing screens, as there are steps that should not be moved or replaced).

（参考设计规范，用于添加和定制屏幕的适当位置，因为有些步骤不应该移动或替换）。

Android 8.1 also introduces @string/wizard\_script\_deferred\_uri for Deferred Setup.

## Script format

Wizard scripts are provided in URI format. Example:

android.resource://com.my.package/raw/custom\_wizard\_script

For details on the URI format, refer to [OpenAssetFileDescriptor](http://developer.android.com/reference/android/content/ContentResolver.html" \l "openAssetFileDescriptor%28android.net.Uri,%20java.lang.String%29" \t "_blank).

In the overlay package, the custom wizard script should use the raw resource type instead of the XML resource type (to retain the setup wizard namespace, the custom wizard script should not be compiled into binary format by AAPT).

## Adding a step

Each step in the wizard script is represented by a WizardAction tag, and each action links to the next one using the result tag. To add an additional step:

向导脚本中的每一步都由一个向导动作标记来表示，并且每个动作都使用结果标记链接到下一个动作。添加一个额外步骤：

1. Change the result tag of the corresponding result to point to your newly added action.

改变相应结果的结果标签，以指向您新添加的动作。

1. Set the result tag of your new action to the next action it should go to.

将您的新动作的结果标记设置为它应该去的下一个动作。

Example of wizard script for the welcome step and the carrier setup step:

<!-- Welcome screen with language selection -->  
<WizardAction id="welcome"  
    wizard:uri="intent:#Intent;action=com.android.setupwizard.WELCOME;end">  
    <result wizard:action="carrier\_setup" />  
</WizardAction>

<!-- Privileged carrier setup app -->  
<WizardAction id="carrier\_setup"  
    wizard:uri="intent:#Intent;action=com.android.setupwizard.CARRIER\_SETUP;end">  
    <result wizard:name="ok"  
        wizard:resultCode="-1"  
        wizard:action="wifi\_settings" />  
    <result wizard:name="not\_ready"  
        wizard:resultCode="1"  
        wizard:action="wifi\_settings" />  
    <result wizard:action="activation\_check" />  
</WizardAction>

To add a step after the welcome screen, change the result tag of welcome to point to your next action, and then set the result of the new action to carrier\_setup.

要在欢迎屏幕之后添加一个步骤，请将欢迎的结果标签更改为指向您的下一个动作，然后将新动作的结果设置为carriersetup。

<!-- Welcome screen with language selection -->  
<WizardAction id="welcome"  
    wizard:uri="intent:#Intent;action=com.android.setupwizard.WELCOME;end">  
    <result wizard:action="additional\_action" />  
</WizardAction>

<!-- New custom action -->  
<WizardAction id="additional\_action"  
    wizard:uri="intent:#Intent;action=com.my.package.ADDITIONAL;end">  
    <result wizard:action="carrier\_setup" />  
</WizardAction>

<!-- Privileged carrier setup app -->  
<WizardAction id="carrier\_setup"  
    wizard:uri="intent:#Intent;action=com.android.setupwizard.CARRIER\_SETUP;end">  
    <result wizard:name="ok"  
        wizard:resultCode="-1"  
        wizard:action="wifi\_settings" />  
    <result wizard:name="not\_ready"  
        wizard:resultCode="1"  
        wizard:action="wifi\_settings" />  
    <result wizard:action="activation\_check" />  
</WizardAction>

## Replacing a step

Similarly, an existing step can be replaced by changing the uri attribute of the action. For example, to provide a custom welcome screen, the script can be modified as follows:

<!-- Welcome screen with language selection -->  
<WizardAction id="welcome"  
    wizard:uri="intent:#Intent;action=com.my.package.CUSTOM\_WELCOME;end">  
    <result wizard:action="carrier\_setup" />  
</WizardAction>

<!-- Privileged carrier setup app -->  
<WizardAction id="carrier\_setup"  
    wizard:uri="intent:#Intent;action=com.android.setupwizard.CARRIER\_SETUP;end">  
    <result wizard:name="ok"  
        wizard:resultCode="-1"  
        wizard:action="wifi\_settings" />  
    <result wizard:name="not\_ready"  
        wizard:resultCode="1"  
        wizard:action="wifi\_settings" />  
    <result wizard:action="activation\_check" />  
</WizardAction>

## Proceeding to next step

继续下一个步骤

When the additional or replaced screen is done with its job, to proceed to the next screen, the screen should send an intent with action com.android.wizard.NEXT. The intent should also contain extras to indicate the setup script, the current action and the result code.

当额外的或替换的屏幕完成它的工作时，进入下一个屏幕，屏幕应该发送一个带有动作 com.android.wizard.NEXT意图。意图还应该包含额外的内容来指示安装脚本、当前动作和结果代码。

For simplicity, all of this is encapsulated in WizardManagerHelper.getNextIntent(), which will return an intent to be started with startActivityForResult:

public class MyActivity extends Activity {

    private static final int NEXT\_REQUEST\_CODE = 1; // Can be any positive int

    public void done(boolean success) {  
        int resultCode = success ? Activity.RESULT\_OK : ResultCodes.RESULT\_SKIP;  
        Intent intent = WizardManagerHelper.getNextIntent(getIntent(), resultCode);  
        startActivityForResult(intent, NEXT\_REQUEST\_CODE);  
        // finish();  // Optional. Calling finish will remove this activity from the  
                      // back stack, such that this activity will be skipped if the  
                      // user clicks back from the next screen.  
    }  
}

During a Wizard Manager session, the back button is usually available on the navigation bar, bringing the user back to the previous screen. This uses the framework's back stack, so, after notifying Wizard Manager to advance to the next screen, an activity that is to remain on the back stack must not call finish().

**TIP:** For details on WizardManagerHelper, refer to the Setup Wizard Library User Guide.

## Scripted activities

Activities invoked as a <WizardAction> should follow the guidelines of the Setup Wizard Library User Guide. Pressing the default NEXT button on the navigation bar will send a RESULT\_OK back to the script. Other common results include:

* RESULT\_SKIP (1).
* RESULT\_RETRY (2).
* RESULT\_CANCELED. Sent when navigating back to the previous screen (not processed by Wizard Manager).

To avoid conflict with future SUW standards, increment custom result codes upward starting from RESULT\_FIRST\_SETUP\_USER (100).

The first matching resultCode determines which action will be executed next. If a <result> element does not contain a resultCode attribute, it will match any resultCode. If a <WizardAction> element does not contain a <result> element, the script will end.

### Standard scripts

GMS includes sample scripts you should use as your starting point.

Starting in GMS 7.0 GMS R9, 7.1 GMS R5, and 8.0 GMS R1, these reference scripts can be found in the GMS bundle ZIP file at these paths:

* - 7.0 GMS: partner\_gms/apps/OpaSuwIntegrationSample/res/raw or ref Standard scripts 7.0.zip
* - 7.1 GMS: partner\_gms/apps/OpaSuwIntegrationSample/res/raw or ref Standard scripts 7.1.zip
* - 8.0 GMS: partner\_gms/apps/GmsSampleIntegration/res/raw or ref Standard scripts 8.0.zip

Previous versions may consult the samples provided in the WizardScript-N-developer-preview.zip.

## WizardScript v2

Android 8.0 introduces some improvements to the WizardScript language that simplify the content and upkeep of its scripts. These v2 enhancements are available when declared in the <WizardScript> element:

<WizardScript

xmlns:wizard="http://schemas.android.com/apk/res/com.google.android.setupwizard"

wizard:version="2">

### Sequential actions

Probably the most common pattern in setup scripts is for one action to link to the next one in sequence. In WizardScript v1, every action in a script must include a link for every anticipated result code; if no matching result is found, the script will exit. For v2, actions are assumed to be in sequential order; result codes only need to be included for branching.

For example (v1):

<!-- Resolve captive portal access, and wait for check-in [REQUIRED] -->

<WizardAction id="captive\_portal"

wizard:uri="intent:#Intent;action=com.android.setupwizard.CAPTIVE\_PORTAL;end">

<result wizard:action="gms\_checkin" />

</WizardAction>

<WizardAction id="gms\_checkin"

wizard:uri="intent:#Intent;action=com.google.android.setupwizard.GMS\_CHECKIN;end">

<result wizard:action="ota\_update" />

</WizardAction>

<!-- Update system packages [REQUIRED] -->

<WizardAction id="ota\_update"

wizard:uri="intent:#Intent;action=com.google.android.setupwizard.OTA\_UPDATE;end">

<result wizard:name="skip"

wizard:resultCode="1"

wizard:action="early\_update" />

<result wizard:action="system\_update" />

</WizardAction>

For v2, actions are executed in order of appearance, unless branching with a matching result code:

<!-- Resolve captive portal access, and wait for check-in [REQUIRED] -->

<WizardAction id="captive\_portal"

wizard:uri="intent:#Intent;action=com.android.setupwizard.CAPTIVE\_PORTAL;end" />

<WizardAction id="gms\_checkin"

wizard:uri="intent:#Intent;action=com.google.android.setupwizard.GMS\_CHECKIN;end" />

<!-- Update system packages [REQUIRED] -->

<WizardAction id="ota\_update"

wizard:uri="intent:#Intent;action=com.google.android.setupwizard.OTA\_UPDATE;end">

<result wizard:name="skip"

wizard:resultCode="1"

wizard:action="early\_update" />

</WizardAction>

Additionally, v2 scripts do not need to specify a wizard:firstAction tag in the <WizardScript> element; they will automatically start with the topmost <WizardAction>.

### Branch labels

Another frequent pattern is for various execution paths to reconverge at a common point to continue with another phase of setup. For v1, this common point is labeled with an ID associated with the next specific action to be performed (e.g. date\_time\_check). As such, when actions are shuffled around to address UX or other functional issues and a common branching target changes from one action to another, all of its links need to be rewired.

For example (v1):

<WizardAction id="restore\_picker"

wizard:uri="intent:#Intent;action=com.google.android.setupwizard.RESTORE;end">

<result wizard:name="skip"

wizard:resultCode="1"

wizard:action="date\_time\_check" />

<result wizard:name="no\_content"

wizard:resultCode="2"

wizard:action="date\_time\_check" />

<result wizard:action="restore\_start" />

</WizardAction>

<WizardAction id="restore\_start"

wizard:uri="intent:#Intent;action=com.google.android.setupwizard.START\_RESTORE;end">

<result wizard:action="date\_time\_check" />

</WizardAction>

<!-- Inserting a new instruction here requires relinking all of the date\_time\_check links -->

<WizardAction id="date\_time\_check"

wizard:uri="intent:#Intent;action=com.google.android.setupwizard.DATE\_TIME\_CHECK;end">

<result wizard:name="skip"

wizard:resultCode="1"

wizard:action="user\_name\_check" />

<result wizard:action="date\_time" />

</WizardAction>

With v2, you can add a branch point containing only an ID (e.g. local\_details), denoting the next stage of setup without presuming any specific action:

<WizardAction id="restore\_picker"

wizard:uri="intent:#Intent;action=com.google.android.setupwizard.RESTORE;end">

<result wizard:name="skip"

wizard:resultCode="1"

wizard:action="local\_details" />

<result wizard:name="no\_content"

wizard:resultCode="2"

wizard:action="local\_details" />

</WizardAction>

<WizardAction id="restore\_start"

wizard:uri="intent:#Intent;action=com.google.android.setupwizard.START\_RESTORE;end">

</WizardAction>

<!-- Branch point for configuring user details that are local to this device -->

<WizardAction id="local\_details" />

<WizardAction id="date\_time\_check"

wizard:uri="intent:#Intent;action=com.google.android.setupwizard.DATE\_TIME\_CHECK;end">

<result wizard:name="skip"

wizard:resultCode="1"

wizard:action="user\_name\_check" />

</WizardAction>

# Google Assistant

Google Assistant makes it easier for users to use their Android devices because many interactions such as asking for the weather or directions, sending a text, and calling can be done by talking to the device in natural language. Artificial Intelligence (AI) technology allows the Assistant to understand the intention of users from the context of interactive conversation as well as information such as user's location, activity on the device, and personal data.

To do all this, Assistant must acquire permission from the user before accessing personal information, during an opt-in flow built into the Google app. To ensure the Assistant is ready to use by the time an Android device completes out-of-the-box setup procedure, we recommend that partners integrate the Assistant opt-in flow into the setup wizard implementation, as described below.

**NOTE:** The following guidelines apply to new devices launching with Android 7.0 or higher.

## Prerequisites

Before starting, ensure you are familiar with the following processes:

* Building Android device system image from AOSP (for details, refer to [Requirements](https://source.android.com/source/requirements" \t "_blank) on source.android.com).
* Integrating Android devices with GMS
* Customizing the GMS SUW (applies only when Google app v6.13 or higher is preloaded on the system image).

## Default opt-in flow

Integrating the Assistant requires a minimum of one (1) screen, with more screens shown depending on the user's actions and configurations.

**NOTE:** The SUW for Android 7.1 provides an opt-in flow reference implementation, so integrating Google Assistant on Android 7.1 devices requires only slight modifications to source code. However, integrating with Android 7.0 devices requires additional source code provided in the associated zip file.

This default flow is composed of the following screens:

|  |  |
| --- | --- |
| https://lh3.googleusercontent.com/nb3VfIqehvQAFt9W01WrpMKaBY5WuFVw8KDSArUnk0UzeyOVCBCEioXhjWOU31j1gJU=w300 | Step 1: Value Proposition Screen If a user chooses a language supported by Google Assistant in the beginning of setup wizard, the default opt-in flow takes the user to this value proposition screen.  The user can proceed to the next step by tapping CONTINUE. |
| https://lh3.googleusercontent.com/xdeciKb0Mf5lT-w2vpfKh43aCNA4eOLclbNENlyB0fQmA2n8SB0Gpriqx0CGHdLZDf4y=w300 | Step 2: Activity Controls Opt-In Screen This step asks the user to give Assistant permission to access the personal information below:   * Web & App Activity from this device * Location History * Device Information from this device * Enable Voice and Audio Activity from this device   The user can opt in and grant the permissions by tapping **YES I'M IN** or tap **NO THANKS** to opt out and exit from the opt-in flow. |
| https://lh3.googleusercontent.com/fePwyqt2X0nC27ooEtkd6GzbGc4zVDXGFy_bVOI0eSkzhoWp2MGXzcdiQxSVLBja6Bus=w300 | Step 3A: Hotword Training Screen Landing Screen (Optional) The Assistant provides the best experience when it is trained to recognize the user's voice. After the Assistant is trained, the user can bring up the Assistant by saying the "OK Google" hotword from any screen.  If the device supports Device Side Processing (DSP) based hotword detection capability, saying "OK Google" also works even if the screen is turned off.  The user can choose to skip this step by tapping DO IT LATER, or start training the Assistant by tapping GET STARTED. If this step is skipped, "OK Google" hotword works only if the Google app runs in the foreground. |
| https://lh3.googleusercontent.com/PPuIZEn0rNOvBYswyJH9BmgWFnhe4GMt7qyGjFkLE7ktR0XsQ-0POvGta3oq9p6vImI=w300 | Step 3B: Hotword Training Screen (Optional) If the user selected GET STARTED from the previous step, the device asks the user to say "OK Google" three times.  Based on the recorded voice, Assistant creates the user's voice model data, which allows to recognize hotword in the background. The user can re-train the Assistant later, if necessary, from the settings of the Google app. |
| https://lh3.googleusercontent.com/7DFlRVWoNraDkQJ-Rde9Maoyj8E0TiHn8bH68ftN_xmDwwqAqwgUscKHGOKmvsVUbho=w300 | Step 4: Trusted Voice Setup Screen (Optional) Trusted Voice is a Smart Lock option provided by Google Play services. It allows the user to unlock the device by saying "Ok Google'" on the lock screen.  This optional screen appears only if the user enabled the secure lock screen (PIN, Pattern, or Password) during the preceding step of the setup process and trained the hotword in step 3B. |
| https://lh3.googleusercontent.com/sYTdPMOhdVYPnBIx9Tou4YB-k1pxaSC5A8B3sVj3xbEjbGsQXlvpqqRY5dNpDkWxGJQ=w300 | Step 5: Screen Context Opt-In Screen (Optional) [Screen Search](https://www.google.com/search/about/learn-more/screen-search/) is a Google app feature that enables the Assistant to show the user instant information about what's currently on the screen. This feature is only enabled by opt-in.  This optional screen appears only if the user tapped DO IT LATER in step 3A - Hotword Training Landing screen or if the Hotword Training screen was not shown. |

### Hotword Training

For devices without Device Side Processing (DSP), training the Hotword enables the user to use "OK Google" whenever the screen is on and the lockscreen (as opposed to only on the Home Screen and the minus-one screen).

If your device has Device Side Processing (DSP) for hotwording or you are interested in adding DSP to your device, contact Google's Hotword team:

* Product Manager: Nino Tasca [nino@google.com](mailto:nino@google.com)
* Engineering Lead: Diego Melendo Casado [mompe@google.com](mailto:mompe@google.com)
* Partnerships: hotword-partnerships@google.com

### Enabling Assistant for Android 7.0 and 7.1

The SUW for Android 7.0 does not have a built-in implementation of the default opt-in flow. However, the GMS distribution zip file contains sample source code for the SetupWizard Overlay APK (which enables the default opt-in flow) in vendor/partner\_gms/apps/OpaSuwIntegrationSample. You can merge this sample source code with the code of your own Overlay APK to add the Assistant opt-in flow to your device setup scenario.

Normally, an Overlay APK does not need to be a privileged app on the system image. However, any Overlay APK that implements the Assistant opt-in flow MUST be privileged as it needs to acquire the permission com.google.android.apps.now.OPT\_IN\_WIZARD from the Google app. For details, refer to the AndroidManifest.xml file in the sample source code.

## Fallback opt-in flow

If the user chooses a language that is not supported by the Assistant, the default opt-in flow in SUW falls back to the legacy opt-in flow, which enables the Google Feed (previously known as Google Now) feature. Examples:

|  |  |
| --- | --- |
| https://lh3.googleusercontent.com/1BQVbtvXhjWsF4yxlKZdH1Ilvor1pfLP946p9r3yJewztzRf5Q2bRSfEj_9fhh8A0qQ=w300 | https://lh3.googleusercontent.com/og_3n5Hv4ujYtCMHoMcF8PypJn9XIoboIMgvyHOV_GZSmVw4njxExBZVoRQrPhNlGA=w300 |

As the Assistant becomes available in other languages, the SUW and the Google app on the system image will automatically take the user to the Assistant opt-in flow when the language is supported. Partners do not need to update the GMS software on the system image for newly-supported languages.

## Enabling Google Assistant

Normally, Google Assistant checks in with Google servers to verify a device is compatible. For a seamless user experience, you can skip this step by setting the following system property at OS compile time:

ro.opa.eligible\_device=true

There are two options to build the combined Overlay APK:

|  |  |
| --- | --- |
| **Android Platform** | Involves compiling the source code and building the APK with the Android build system included in the AOSP. The sample source code in the GMS distribution relies on the Setup Wizard library included in frameworks/opt/setupwizard/library in the AOSP repository.  If you merge the sample code into your Overlay APK, it must link with the Setup Wizard library, which you can do by adding the following line to your Android.mk file:  include frameworks/opt/setupwizard/library/common.mk |
| **Android Studio** | Involves compiling the source code and building the APK with the Android Studio and Gradle. If you're using the Android Studio to build your Overlay APK, the Setup Wizard library must be manually added to the project. For details on creating an Android Library (AAR) file from the source code, refer to the [Android Studio user guide](https://developer.android.com/studio/projects/android-library.html" \t "_blank).  For your convenience, we provide a precompiled SUW Library AAR. For details on adding an AAR library file to your project, refer to [Add your library as a dependency](https://developer.android.com/studio/projects/android-library.html" \l "AddDependency" \t "_blank). |

## Customizing the opt-in flow

When the default opt-in flow is used during the setup process, the user might traverse all five (5) steps. To simplify the flow, you can customize the default opt-in flow using one or more of the following:

* **Making Value Proposition Screen skippable**. Add a SKIP button on the screen so users can skip the entire opt-in flow.
* **Disabling Hotword Training Screen**. Remove the hotword training screens from the opt-in flow (also removes the Screen Context Opt-In Screen from the app). End users can perform hotword training at any time from the Google app settings menu.

### Enabling customizations

To enable customization options, create an Overlay APK with the reference SUW scripts.

**IMPORTANT:** Supported only by Android 7.1 GMS SetupWizard version 225.XXXXXXX or higher.

Some scripts include the WizardAction tag that triggers the default opt-in flow:

<!-- Set up Google Assistant (OPA). This step should come right after Google Services. -->  
<WizardAction id="opa" wizard:uri="intent:#Intent;action=com.google.android.setupwizard.OPA\_OPT\_IN;end">  
    <result wizard:action="suggested\_actions" />  
</WizardAction>

To enable customization options, add the boolean extra values to the WizardAction:

<WizardAction id="opa"  
wizard:uri="intent:#Intent;action=com.google.android.setupwizard.OPA\_OPT\_IN;B.valuePropSkippable=true;B.hotword=false;end">  
    <result wizard:action="suggested\_actions" />  
</WizardAction>

### Customizing look and feel

As OEMs can select which SUW steps to integrate, the look and feel of each step might not match (e.g., Status Bar color). This is a broader challenge within Android and is not addressed in the Google Assistant integration.

# Configuring Optional Steps

Users are diverse in what they look for in a setup experience. While some users want to complete setup as quickly as possible and start using the device, others want to spend time learning about a new purchase. Carrier and OEM needs are similarly diverse, and often involve customizing or adding features to the SUW. Many such features are of interest to a subset of users: for instance, we saw that 30% of Nexus users chose to add a non-Google email account during Android 6.0 setup.

In Android 6.0 and earlier, the standard pattern for adding a step to the SUW was to place it into the main flow so every user sees it; users who don't want to configure that step must decline it. While simple to configure, this approach does not scale: As partners add steps, the setup experience gets longer for every user, with a growing percentage of steps garnering little interest among most users. In Android 7.0, setup customization was changed to use optional steps.

## Core concepts

Optional steps use the following core concepts to ensure the setup flow is short, remains configurable by partners, and skipped steps remain configurable by users.

* **The core setup flow is as short as possible**, consisting only of essential steps such as:
  + Choosing a language
  + Connecting to the network (including SIM activation)
  + Adding a Google account (to receive app updates, restore from a backup, or activate device protection)
  + Setting up a lock screen (to activate device protection)
  + Essential terms and conditions and consents.
* **Non-essential items reside in an Optional Steps menu**, in which:
  + The contents and ordering of this menu are partner-configurable.
  + Users can pick and choose which steps to configure at setup time.
  + Steps living in this menu conform to the UI guidelines for Setup Wizard.
* **Items a user does not configure during setup can be configured to appear later** in a Suggestions section at the top of the Settings apps.

Use these guidelines, partners can add steps to the setup experience for their customers, users can configure these steps at a time and place of their choosing; and the setup wizard can scale to meet increasingly diverse needs.

## Google-provided optional steps

As of Android 7.1, Google provides the following optional steps:

* Add a non-Google email account
* Set up "OK Google" for voice search
* Add a form of payment (Android Pay, credit card)
* Set Emergency Contact Info

## Declaring an optional step

You can declare an optional setup step without code changes in SUW. A system app can declare an optional step by adding an activity with intent action android.intent.action.MAIN and a corresponding intent category:

<activity  
    android:icon="@drawable/my\_icon"  
    [...]>  
<intent-filter android:priority="10">  
    <action android:name="android.intent.action.MAIN" />  
    <category android:name="com.android.settings.suggested.category.DEFAULT" />  
</intent-filter>  
<meta-data android:name="com.android.settings.title"  
    android:resource="@string/my\_title" />  
<meta-data android:name="com.android.settings.summary"   
android:resource="@string/my\_summary" />  
</activity>

Specify the attributes of the optional step as <meta-data> of the activity in the manifest, and extract using PackageManager APIs. Non-system bundled applications are ignored.

## Controlling optional steps

The SUW controls the list of options shown in the optional step screen using XML. Each step defines a category, and different categories appear on the screen in the order they are declared in XML. OEMs can customize the XML using resource overlays:

<array xmlns:android="http://schemas.android.com/apk/res/android">  
    <action android:category="com.android.settings.suggested.category.LOCK\_SCREEN" />  
    <action android:category="com.android.settings.suggested.category.EMAIL" />  
<action android:category="com.android.settings.suggested.category.PAYMENT"  
android:package="com.google.android.gms" />  
    <action  android:category="com.android.settings.suggested.category.PARTNER\_ACCOUNT"  
android:isRepeatable="true" />  
<action android:category="com.android.settings.suggested.category.DEFAULT"  
android:isRepeatable="true" />  
</array>

To provide a custom list of optional step categories using resource overlays, the APK must declare a broadcast receiver for the intent action com.android.setupwizard.action.PARTNER\_CUSTOMIZATION. The broadcast receiver is used by setup wizard to find the APK and therefore must be declared in the manifest XML.

<receiver android:name=".PartnerReceiver">  
    <intent-filter>  
        <action android:name="com.android.setupwizard.action.PARTNER\_CUSTOMIZATION" />  
    </intent-filter>  
</receiver>

For the SUW to recognize the overlay APK, the APK must be a system package with [ApplicationInfo.FLAG\_SYSTEM](http://developer.android.com/reference/android/content/pm/ApplicationInfo.html" \l "FLAG_SYSTEM). By providing a value for suggested\_actions\_uri, the APK can point to an alternative list of categories for optional steps. Values must be in URI format, e.g., android.resource://com.my.package/raw/custom\_suggested\_actions.

### Restricting by package

To restrict activity to a known package, you can add an optional package attribute to a step. This restriction is can be used in late stages of building a system image to restrict steps shown in the list to known steps or to implement a whitelist. For example, to restrict PARTNER\_ACCOUNT setup to setting up HTC and T-mobile accounts, specify:

<action android:category="com.android.settings.suggested.category.PARTNER\_ACCOUNT"  
    package="com.htc.setup" />  
<action android:category="com.android.settings.suggested.category.PARTNER\_ACCOUNT"  
    android:package="com.tmobile.mytmobile" />

### Using multiple-item categories

A step can specify android:isRepeatable="true" to indicate that multiple implementers of the category will be shown in the list and ordered by their priority. If multiple is false (default) but the category has multiple implementers, only the implementer with the highest priority is shown.

**NOTE:** When package is defined, the android:isRepeatable attribute is implied to be false.

### Setting the list order

Suggested actions are shown to users in the order they are declared in the XML file. However, as the SUW and settings use separate XML files, items can have a different order in each file. For such discrepancies, within a multiple-item category items are ordered according to their priority (higher priority means closer to top of the list).

### Removing duplicates

If an activity (or alias) implements multiple categories on the list, attempts are made to show each step only once. For example, if EMAIL and PARTNER\_ACCOUNT resolve to the same component, option steps show only EMAIL because it is higher in the steps schema.

## Completing a step

After a step is completed, a step should return an activity result code using Activity.setResult(). If the result is not RESULT\_CANCELED, the step is considered to be completed successfully. All successfully completed components are disabled using PackageManager. Similarly, each step completed in Settings is disabled as soon as it is completed.

## Configuring conditional items

Some steps might be available only when specific conditions are met, such as not showing voice hotwording setup on a low-RAM device. Conditional steps can use the android:enabled flag for the activity component (or activity-alias) to denote whether they should be shown or not. Items can also be conditioned on support for a required feature, the presence of an account, or based on localized values.

Examples:

* To require a Google account, include the following meta-data for the activity:

<meta-data android:name="com.android.settings.require\_account" android:value="com.google" />

* To disable an item based on the availability of a specific feature, include the following meta-data:

<meta-data android:name="com.android.settings.require\_feature" android:value="android.hardware.fingerprint" />

* To disable an item based on locale, include the the following meta-data:

<meta-data android:name="com.android.settings.is\_supported" android:resource="@bool/is\_optional\_step\_supported" />

## OEM customization

As partners can add optional steps without SUW code changes, OEMs can simply add implementations of PARTNER\_ACCOUNT or DEFAULT to show more steps. For example, an OEM who wants to provide a different default email client can provide an implementation for EMAIL with a higher priority than Gmail. For other unexpected use cases, OEMs can use resource overlay to override the <optional-steps> XML (described above) and provide a custom ordering or custom categories.

For example, OEMs and carriers often ask users to sign in to an account such as a T-mobile or Samsung account. When implementers of the T-mobile account setup and Samsung account setup are both on the device, both should be shown to users to enable setting up both accounts if they choose to. Even though both T-mobile account and Samsung account setup flows implement PARTNER\_ACCOUNT category, both flows will be shown using the the android:isRepeatable="true" attribute in the steps declaration XML.

# Enterprise

To ensure Enterprise devices can be provisioned successfully, custom SUW implementations must comply the following requirements.

## Google account-based setup

As of Android 6.0, users adding a managed Google account to the device are prompted to download and install the EMM management application, which guides users through additional setup steps. When installed, the management application can set up devices in any Enterprise modes, including device owner, profile owner, and device admin. After a management mode is selected, the provisioning process continues normally.

**NOTE:** When device owner provisioning completes, the SUW is expected to close.

## User provisioning state

Managed Android devices can be provisioned using different methods in the SUW, so each activity in the SUW must determine whether the device has been provisioned for management and exit the consumer flow if necessary. (This is already implemented in the Google SUW.)

### Checking state in custom actions

Provisioning state is indicated via the DevicePolicyManager.getUserProvisioningState system API added in Android 7.0 (API level 24).

* Starting from Android 8.1 (API level 27), when NFC provisioning completes, SUW will jump to oem\_post\_setup action directly. Custom activities should still check for provisioning state as it may be set by QR code provisioning.
* If the user provisioning state is STATE\_USER\_SETUP\_COMPLETE, the remainder of SUW should be suppressed.
* If the state is STATE\_USER\_SETUP\_INCOMPLETE or STATE\_USER\_PROFILE\_COMPLETE, the remainder of SUW should be shown normally:

@Override  
protected void onResume() {  
    super.onResume();

    // For managed provisioning with QR code or NFC, and for managed  
    // provisioning after encryption-induced reboot, the user may be  
    // returned here. Check the DPM provisioning state and exit early  
    // if the provisioning is already complete. If resume was triggered  
    // while finishing (as can occur with auto-rotation redraws),  
    // do nothing.  
    final DevicePolicyManager dpm =   
            getSystemService(DevicePolicyManager.class);  
    if (dpm.getUserProvisioningState() ==   
            DevicePolicyManager.STATE\_USER\_SETUP\_COMPLETE) {  
        nextAction(ManagedProvisioningResultCodes.RESULT\_DPM\_USER\_COMPLETE);  
    }  
[...]

<!-- Welcome screen with language selection [RECOMMENDED, CUSTOMIZABLE] -->  
<WizardAction id="welcome"  
        wizard:uri="intent:#Intent;action=com.android.setupwizard.WELCOME;end">  
    <result wizard:name="dpm\_user\_complete"  
            wizard:resultCode="111"    <!-- RESULT\_DPM\_USER\_COMPLETE -->  
            wizard:action="oem\_post\_setup" />  
    [...]  
</WizardAction>

<!-- OEM completion [CUSTOMIZABLE] -->  
<WizardAction id="oem\_post\_setup"  
        wizard:uri=  
                "intent:#Intent;action=com.android.setupwizard.OEM\_POST\_SETUP;end">  
    <result wizard:action="exit" />  
</WizardAction>

### Finalizing managed provisioning setup

If a provisionings occurred during SUW, i.e. if the user provisioning state is different from STATE\_USER\_UNMANAGED, SUW should start the activity intent ACTION\_PROVISION\_FINALIZATION instead of the HOME intent upon completion. This launches the provisioning application that transfers control to the newly set device or profile owner.

Google's SetupWizardExitActivity implements the correct behavior, so if SUW is finished via com.android.setupwizard.EXIT intent, no work is needed. If the OEM implements their own exit activity, add the following code snippet to its onCreate() function:

final DevicePolicyManager dpm = getSystemService(DevicePolicyManager.class);  
if (dpm.getUserProvisioningState() !=   
        DevicePolicyManager.STATE\_USER\_UNMANAGED) {  
    // Start the finalization step to allow managed provisioning to finalize   
    // the setup. The finalization step will launch MAIN / HOME when it   
    // completes.  
    startActivity(  
            new Intent(DevicePolicyManager.ACTION\_PROVISION\_FINALIZATION)  
                    .addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK  
                            | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK  
                            | Intent.FLAG\_ACTIVITY\_RESET\_TASK\_IF\_NEEDED));  
    finishAllAppTasks();  
} else {  
    // Launch the home action, which should lead to the real home screen now  
    launchHome();

    // SetupWizardE​xitActivity is NoDisplay and must finish before resume  
    finishAllAppTasks();  
    TransitionHelper.applyForwardTransition(this,  
            TransitionHelper.TRANSITION\_FADE);  
 }

## Device Owner Mode

Enterprise device owner mode must be provisioned from the device setup wizard. As a security precaution, this can occur via NFC bump or by signing in with a managed Google account. In the case of an NFC bump, the provisioning process is designed to replace all user interactions required in the SUW.

**NOTE:** The user provisioning guidance applies in all cases (for example, on activities that allow an NFC tap to trigger NFC provisioning).

### NFC Provisioning

The NFC provisioning flow can be used for large scale enterprise deployment of corporate-owned devices. In this mode, all settings are transferred through NFC and subsequent DevicePolicyManager API calls from the Device Policy Client (DPC) with no SUW UI shown (to enable NFC provisioning, devices with NFC must enable NFC throughout the SUW).

While it is encouraged to show minimal UI in this mode, you can display legal agreements or other important screens that cannot be skipped by checking for the following:

The extensions of the SUW that show the agreements can listen to the HOME intent with a priority higher than 2 and show the legal agreements when the intent is received. This is an intent that would usually be used to allow finishing the device setup in a clean way. It is therefore essential that after the legal statements have been accepted before another HOME intent is fired to return to the device setup flow.

### QR Code Provisioning

**Devices with a camera must support QR Code Device Owner provisioning**, an alternative to NFC Provisioning. When using this method, instead of the configuration payload being delivered via an NFC bump, it is delivered by a QR code that is scanned during SUW.

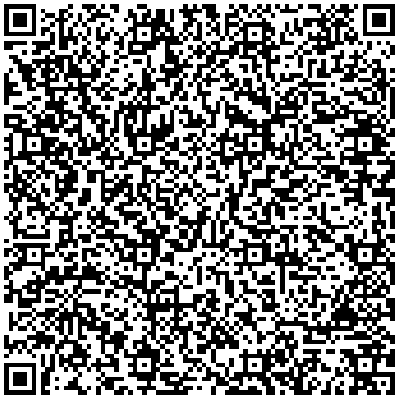
* To activate QR Code Provisioning, a user makes six (6) consecutive taps on the same area of the opening screen of SUW; third-party SUWs can use the ConsecutiveTapsGestureDetector class in the Setup Wizard Library UI for assistance (get [sample code to incorporate into a custom Welcome Activity](#_Sample_Code:_QR)).
* The first screen of the QR Code Provisioning process describes the flow to the end user. The device is then taken online to download a component necessary for the QR scan. As a precaution to ensure users with accessibility needs do not enter the QR provisioning flow accidently, the first Activity times out after 60 seconds of user inactivity and returns to the previous/calling Activity.
* After the QR scan, the remainder of the flow is identical to NFC Provisioning, i.e. minimal UI that includes any necessary legal agreements and the Device Policy Client (DPC) that takes over with no SUW UI shown.

#### Testing QR code provisioning

QR codes used for provisioning should be UTF-8 encoded JSON strings that contain the properties outlined in DevicePolicyManager.ACTION\_PROVISION\_MANAGED\_DEVICE\_FROM\_TRUSTED\_SOURCE. Example valid code:

{  
    "android.app.extra.PROVISIONING\_DEVICE\_ADMIN\_COMPONENT\_NAME":   
            "foo.bar/foo.bar.DeviceAdminReceiver",  
    "android.app.extra.PROVISIONING\_DEVICE\_ADMIN\_SIGNATURE\_CHECKSUM":   
            "gJD2YwtOiWJHkSMkkIfLRlj-quNqG1fb6v100QmzM9w=",  
    "android.app.extra.PROVISIONING\_DEVICE\_ADMIN\_PACKAGE\_DOWNLOAD\_LOCATION":   
            "http://foobar/foobar.apk",  
    "android.app.extra.PROVISIONING\_SKIP\_ENCRYPTION": false,  
    "android.app.extra.PROVISIONING\_WIFI\_SSID": "FooBar",  
    "android.app.extra.PROVISIONING\_ADMIN\_EXTRAS\_BUNDLE": {  
        "dpc\_device\_account": "device1@acme.com",  
        "dpc\_device\_auth": "device\_token",  
        "another\_custom\_dpc\_key": "dpc\_custom\_value"  
    }  
}

To test, use the following sample QR code to trigger QR provisioning:

****

This QR code triggers the download of a device policy app called [TestDPC](https://play.google.com/store/apps/details?id=com.afwsamples.testdpc" \t "_blank) (sample app written by Google) and sets it as a device owner.

#### Sample Code: QR Code Provisioning

This code can be used in the first (Welcome) Activity of a custom SUW to setup the necessary gesture to invoke the QR Code Provisioning DeviceOwner setup:

private final OnConsecutiveTapsListener mOnConsecutiveTapsListener =  
        new OnConsecutiveTapsListener() {  
            @Override  
            public void onConsecutiveTaps(int welcomeTapCounter) {  
                /\* 6 taps to triggering QR provisioning. \*/  
                if (welcomeTapCounter >= 6) {  
                    nextAction(101 /\* start qr provisioning \*/);  
                }  
           }  
       };  
onCreate:

mConsecutiveTapsGestureDetector = new ConsecutiveTapsGestureDetector(  
                    mOnConsecutiveTapsListener,  
                    /\* target view \*/);  
onResume:

mConsecutiveTapsGestureDetector.resetCounter();

#### Sample Code: Wizard Script

Here is an excerpt of the wizard script for the welcome and QR code provisioning step:

<!-- Welcome screen with language selection -->  
<WizardAction id="welcome"  
    wizard:uri="intent:#Intent;action=com.android.setupwizard.WELCOME;end">  
    <result wizard:name="start\_qr\_provision"  
            wizard:resultCode="101"  
            wizard:action="qr\_provision\_flow" />  
    <result wizard:name="dpm\_user\_complete"  
            wizard:resultCode="111"  
            wizard:action="oem\_post\_setup" />  
    <result wizard:action="sim\_missing" />  
</WizardAction>  
    <!-- QR provision flow -->  
    <WizardAction id="qr\_provision\_flow"  
        wizard:script="android.resource://com.google.android.setupwizard/xml/wizard\_script\_qr\_provision\_flow">  
        <result wizard:action="oem\_post\_setup" />  
    </WizardAction>

#### Sample Code: Other Provisioning Methods

A manufacturer can enable provisioning methods that use other techniques (such as barcode reader) to deliver the same information in an NFC tag or QR code. For example, a device may have a dedicated barcode scanner or other peripheral that can be used to establish the provisioning parameters. The manufacturer should extract the information required from the input and then send an intent similar to:

Intent qrProvisionIntent = new Intent(ACTION\_PROVISION\_MANAGED\_DEVICE\_FROM\_TRUSTED\_SOURCE);  
qrProvisionIntent.putExtra(EXTRA\_PROVISIONING\_DEVICE\_ADMIN\_COMPONENT\_NAME, componentName)  
        // DPC download and checksum  
        .putExtra(EXTRA\_PROVISIONING\_DEVICE\_ADMIN\_PACKAGE\_CHECKSUM, dpcCheckSum)  
        .putExtra(EXTRA\_PROVISIONING\_DEVICE\_ADMIN\_PACKAGE\_DOWNLOAD\_COOKIE\_HEADER,  
                cookieHeader)  
        .putExtra(EXTRA\_PROVISIONING\_DEVICE\_ADMIN\_PACKAGE\_DOWNLOAD\_LOCATION,  
                dpcDownloadLocation)  
        // Locale and time  
        .putExtra(EXTRA\_PROVISIONING\_LOCAL\_TIME, localTime)  
        .putExtra(EXTRA\_PROVISIONING\_LOCALE, locale)  
        .putExtra(EXTRA\_PROVISIONING\_TIME\_ZONE, timeZone)  
        // Encryption  
        .putExtra(EXTRA\_PROVISIONING\_SKIP\_ENCRYPTION, skipEncryption)  
        // Wi-Fi configuration  
        .putExtra(EXTRA\_PROVISIONING\_WIFI\_HIDDEN, wifiHidden)  
        .putExtra(EXTRA\_PROVISIONING\_WIFI\_PASSWORD, wifiPassword)  
        .putExtra(EXTRA\_PROVISIONING\_WIFI\_PAC\_URL, wifiPacUrl)  
        .putExtra(EXTRA\_PROVISIONING\_WIFI\_PROXY\_BYPASS, wifiProxyByPass)  
        .putExtra(EXTRA\_PROVISIONING\_WIFI\_PROXY\_HOST, wifiProxyHost)  
        .putExtra(EXTRA\_PROVISIONING\_WIFI\_PROXY\_PORT, wifiProxyPort)  
        .putExtra(EXTRA\_PROVISIONING\_WIFI\_SECURITY\_TYPE, wifiSecurityType)  
        .putExtra(EXTRA\_PROVISIONING\_WIFI\_SSID, wifiSsid)  
        // Admin extras bundle  
        .putExtra(EXTRA\_PROVISIONING\_ADMIN\_EXTRAS\_BUNDLE, adminExtrasBundle);  
startActivity(qrProvisionIntent);

## Play Auto Install

Device Owner mode is designed for fully-managed devices, Play Auto Install (PAI) for applications is suppressed when the Android Managed Provisioning process sets the USER\_SETUP\_COMPLETE flag. PAI is also unavailable when setting up Enterprise Managed Profiles. In both cases, applications are available for install and management through Play for Work, managed by the user's organization.

## Testing

An automated test harness and test Google account can be provided for this purpose. For details, contact your Technical Account Manager.

# Deferred Setup Wizard

Deferred Setup Wizard allows users to finish setting up and customizing a device after initial device setup. Deferred SUW consists of a series of steps that are similar in flow and operation to SUW, but has minor differences to keep in mind when implementing:

* Deferred Setup is an **optional** feature for users and therefore should not contain setup steps or information the user must not skip.
* Users will have some time between completing Setup Wizard and starting Deferred Setup where they have full access to their device. As such, all steps that must be completed before giving users this access must happen in the initial Setup Wizard.
* Deferred Setup does not prevent users from pausing or quitting its sequence of steps before completion. This is in contrast to Setup Wizard that disables the ability to change apps or navigate to the home screen. Therefore, steps in Deferred Setup cannot assume they will be completed when started or will not be interrupted.

Despite these differences, Deferred Setup is very similar to SUW in both user experience and underlying technology. It can be customized in a similar fashion with its own wizard script and suggested actions category list, and uses the same intents to launch common actions (for details, see [Customizing SUW](https://support.google.com/androidpartners_gms/answer/7385409)).

## Deferred Setup entry points

Deferred Setup in Android 8.1 encourages users to complete setup at the following entry points:

* In **Settings**, which temporarily takes the place of Suggested Actions at the top. After Deferred Setup is either completed or dismissed, Suggested Actions returns to the top.
* A **notification** displayed approximately 15 minutes after Setup Wizard finishes.

Both entry points are managed by a scheduler that is launched at the end of initial Setup Wizard.

## Snoozing Deferred Setup notification

Users have the option to snooze or dismiss the notification entry point to Deferred Setup. Doing so tells the scheduler to hide the notification and when to display it again. Snoozing behavior and options are not customizable at this time.

## Scheduler behavior

The Deferred Setup Scheduler wakes every 15 minutes to monitor the state of the phone after initial Setup Wizard finishes. Based on many conditions, the Scheduler determines whether or not to show the Deferred Setup entry points and the message variant to display to the user. Changes to the notification (as a result of the user snoozing or dismissing them or as a result of the user starting or completing Deferred Setup) are also handled by the Scheduler. The Scheduler is not customizable at this time.

## Deferred Setup flag

Deferred Setup has its own lifecycle manager and based on that, any intents in the wizard script that are started during Deferred Setup will have the extra "**deferredSetup**" set to true. Based on the deferredSetup value, Actions that can be performed in either Setup Wizard or Deferred Setup can have different user experiences for each type of setup.