Customizing the Setup Wizard in Android 5.x

Contents

Background

Android Setup Wizard (SUW) is the Android component that, on first run, connects a new device to the Internet and allows the user to sign in or create a Google account. OEMs and carriers may customize SUW to incorporate their own branding and add their own setup and account configuration steps.

Overview

The full SUW contains the following components:s

1. Language + Accessibility selection
2. SIM & cellular service activation
3. Wi-Fi connection
4. Factory Reset Protection, to prevent reuse of a device which has been reset in an untrusted manner.
5. Mandatory OTA
6. Early Update, a mechanism for Google Play to automatically update itself before continuing within SUW
7. Tap & Go, lets users set up their new device by tapping it with their old device
8. Google Authentication (sign into or sign up for a Google account, which includes wallet setup)

9. Date and time settings
10. User name settings
11. Restoration from backup
12. Play Auto Installs, which allow OEMs or carriers to customize initial home screens and install apps immediately after setup
13. Google opt-ins and terms of service (ToS)
14. Google Now opt-in (if Google Now Launcher is pre-installed)

Customization methods

OEMs can customize the setup experience for an Android device in one of two ways:

- Wrap the complete Android SUW by prepending and appending screens
- Create a setup application from scratch

Note: Android devices compliant with Google Mobile Services (GMS) MUST use one of these methods.

Each method is detailed below. The first method “Wrapping the Android SUW” is strongly preferred.

Note: In either method, 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.

N.B. 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.**

Wrapping the Android SUW

To prepend a screen, create an Activity that handles the `com.android.setupwizard.OEM_PRE_SETUP` intent. After completion, this Activity should call the NEXT action, with the scriptUri and actionId extras sent when the prepended Activity is started, thus:

```
Intent intent = new Intent("com.android.wizard.NEXT");
intent.putExtra("scriptUri",
getIntent().getStringExtra("scriptUri"));
intent.putExtra("actionId", getIntent().getStringExtra("actionId"));
intent.putExtra("theme", getIntent().getStringExtra("theme"));
intent.putExtra("com.android.setupwizard.ResultCode", <result code>);
startActivityForResult(intent, 10000);
```

<result code> is usually RESULT_OK, but can be any valid result code that is not RESULT_CANCELED.

To append a screen, create an Activity that handles the `com.android.setupwizard.OEM_POST_SETUP` intent. After completion, this Activity should call the NEXT action, with the scriptUri and actionId extras sent when the prepended Activity is started, as above.

OEMs may choose between a light and dark color scheme for SUW by setting the system property `ro.setupwizard.theme` to `material` to use the dark theme, or `material_light` for the light theme. The default theme is `material_light`.

Overlay mechanism (available starting in L MR1)
A new overlay mechanism is available starting in MR1 to allow customizing the setup wizard flow with much greater flexibility. To take advantage of that, you will need to have a system APK that receives the broadcast `com.android.setupwizard.action.PARTNER_CUSTOMIZATION`. The broadcast will not be sent — it is only used for resolving the package.

The overlay APK can then change the flow by providing a custom wizard script. First in `config.xml` add a string resource `wizard_script_uri`. The value of the resource should be a URI pointing to a raw resource available in the APK itself, for example,

```
     <string name="wizard_script_uri" translatable="false">
android.resource://com.google.android.setupwizard.overlay.sample/raw/custom_wiz
ard_script
      </string>
```

A corresponding resource `wizard_script_user_uri` is also available for customizing the flow when creating a new user profile.

The scripts should be based on the samples provided below:
For `wizard_script_uri`: wizard_script.xml
For `wizard_script_user_uri`: wizard_script_user.xml

Note: whilst this method enables an extremely flexible level of customization, the resulting Setup Wizard implementation **must** adhere to the rest of these guidelines and other contracts.

Debug Logging
For additional logging of Setup Wizard internals during development, issue the following adb

command:

```
adb shell setprop log.tag.SetupWizard VERBOSE
```

Creating a Setup application from scratch

1. If you need to block users from advancing in Setup Wizard without network connectivity, define “ro.setupwizard.require_network” in the makefile.

   Please note that "ro.setupwizard.network_required" (deprecated) was a boolean, while “ro.setupwizard.require_network” takes “wifi” to specifically require Wifi or “any” to also allow mobile data. "ro.setupwizard.network_required" is deprecated in Lollipop, and will be removed in M-release.

2. You must call `AccountManager.addAccount()` to allow users to sign into or create a Google account, using the appropriate color scheme.

   This sign-in/creation step must occur before any option to sign into or create other accounts is provided and immediately after connection to the Internet (if an Internet connection is achieved during SUW), before any other activities.

   Here is sample code:

```
AccountManager acctMgr = AccountManager.get(this);
Bundle options = new Bundle();
options.putBoolean("firstRun", true);
options.putBoolean("setupWizard", true);

Bundle accountManagerOptions = new Bundle();
accountManagerOptions.putBoolean("allowSkip", true);
options.putBundle("accountManagerOptions", accountManagerOptions);
AccountManagerFuture futBundle = acctMgr.addAccount(
        "com.google",  // Google’s account type.
        null,   // The authTokenType is not relevant to Google.
        null,   // No requiredFeatures.
        options,
        null,  // The Activity context.
        null,  // Simplest just to handle an error intent directly.
        null);  // No callback implies no handler.
try {
    Intent googleSignInWorkFlow =
        futBundle.getResult().getParcelable(AccountManager.KEY_INTENT);
    startActivityForResult(googleSignInWorkFlow, REQUEST_CODE);
} catch (Exception e) {
    // AccountManager.addAccount throws various exceptions.
    // Left to the implementer.
}
```

OEMs may to switch from the light to a dark color scheme for SUW (except for the Google sign-in screens) by passing an extra `theme=material` in the "options" bundle.

Factory Reset Protection support (Android 5.1 or later): The Google Account Authenticator bundled in Google Play Service[1]s will handle **factory reset protection** without any additional work. But there is also a FrpClient utility class bundled with Google Play services client jars that will allow SUW implementers to determine if Google's factory reset protection is supported by the hardware and whether or not it is enabled (e.g. whether a challenge would be required to proceed through Google account sign-in). This class can be used by SUW implementers to decide whether or not a user should be allowed to sign into a walled garden Wi-Fi network or prevent the user from finishing the SUW.

To implement factory reset protection in a custom SUW, `FrpClient.isChallengeSupported` and `FrpClient.isChallengeRequired` should be loaded from a background thread, and if the result of `isChallengeRequired` is true, then the SUW should make sure the user goes through the `AccountManager.addAccount` flow above (i.e. force the user to connect to Internet). If `isChallengeSupported` is true, the SUW should prompt the user to set up a screen lock in order to activate factory reset protection, and warn them if they choose not to.

Below is a snippet from the Android SUW showing how it loads the status of factory reset protection.

```
private Context mContext;
private final FrpClient mFrpClient;
private FrpTask mFrpRequiredTask;

public FrpHelper(Context context) {
    mContext = context.getApplicationContext();
    mFrpClient = new FrpClient(mContext);
    loadFrpStatus();
}
```

[1] Additional aar library files are required for FRP support. Please refer to GMS release notes for the details.

```
    private FrpTask loadFrpStatus() {
        mFrpRequiredTask = new FrpTask();
        mFrpRequiredTask.execute();
        return mFrpRequiredTask;
    }

    private FrpTask getFrpRequiredTask() {
        return mFrpRequiredTask;
    }

    private class FrpTask extends AsyncTask<Void, Void, FrpStatus> {

        @Override
        protected FrpStatus doInBackground(Void... voids) {
            FrpStatus status = new FrpStatus();

            status.challengeRequired = mFrpClient.isChallengeRequired();
            status.supported = mFrpClient.isChallengeSupported();
            Log.i(TAG, "FRP status: " + status);

            return status;
        }
    }

    public static class FrpStatus {
        public boolean challengeRequired = false;
        public boolean supported = false;

        @Override
        public String toString() {
            return "supported: " + supported + ", challengeRequired: " +
challengeRequired;
        }
    }
```

3. Invoke `SetupWizard.VENDOR_SETUP` activity. This shows the Google Services screen, including Location consent checkbox.

   Here is sample code:

```
Intent intent = new
Intent("com.google.android.setupwizard.VENDOR_SETUP");
intent.setPackage("com.google.android.setupwizard");
startActivityForResult(intent, REQUEST_CODE);
```

4. If OEMs pre-install the Google Now Launcher, they must include the Google Now opt-in during their setup application using the intent `com.google.android.apps.now.OPT_IN_WIZARD` and declaring the permission also named: `com.google.android.apps.now.OPT_IN_WIZARD`

5. Call Android APIs to restore user data. The method here triggers a pre-K restoration flow which automatically selects an appropriate device, associated with the users Google account, to restore from. The first and preferred method for customizing Setup Wizard allows users to choose from a list of available backup sets for their devices.

   Before calling restore API,
   Begin restore session:
   `BackupManager.beginRestoreSession()`

   Find available snapshots to restore:
   `RestoreSession.getAvailableRestoreSets()`

   Restore data:
   `RestoreSession.restoreAll()`

   OEMs should always use the first entry in the array returned by `getAvailableRestoreSets()`, which is the recommended dataset by the Google backend for setup time restore for the device.

6. Android for work device owner provisioning: to support this, when the setup wizard opens, it needs to check if another process (such as device owner provisioning) has already finished the user setup. If this is the case, it needs to fire a home intent and finish. This intent will be caught by the provisioning application, which will then hand over control to the newly set device owner. This can be achieved adding the Add the following to your setup wizards main activity:

```
@Override
 protected void onStart() {
   super.onStart();

   // When returning to setup wizard, check if another setup process
   // has intervened and, if so, complete an orderly exit
   boolean completed = Settings.Secure.getInt(getContentResolver(),
           Settings.Secure.USER_SETUP_COMPLETE, 0) != 0;
   if (completed) {
     startActivity(new Intent(Intent.ACTION_MAIN, null)
          .addCategory(Intent.CATEGORY_HOME)
          .addFlags(Intent.FLAG_ACTIVITY_NEW_TASK
                      | Intent.FLAG_ACTIVITY_CLEAR_TASK
                      | Intent.FLAG_ACTIVITY_RESET_TASK_IF_NEEDED));
      finish();
    }
     …
  }
```

On devices with NFC, 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) withno SUW UI shown. While it is encouraged to show only minimal UI in this mode, if there is a need for legal agreements or screens which cannot be skipped, they can be displayed by checking for the following:

If SUW comes into focus and the USER_SETUP_COMPLETE flag has been set by another process[2], then only the essential UI should be shown before exiting as described above..

Design Principles

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

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.

2. *Get to know me*
   Learn peoples' preferences over time. Rather than asking them to make the same choices over and over, place previous choices within easy reach.

3. *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.

4. *Keep it brief*
   Use short phrases with simple words. People are likely to skip sentences if they're long.

[2] This presumes that ManagedProvisioning is the process other than SUW itself which sets the flag

5. *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.

6. *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.

7. *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.

Goals

The goals for SUW are:

- To give the user a delightful and visually bold out-of-box experience that sets the tone for Android Lollipop.

- To get the user into a stable state as quickly and clearly as possible; the user should not only be able to move quickly between steps but also understand what each step is asking of them and why.

A “good 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 restores 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.

OEMs may use either the light or dark SUW color schemes, as they wish, to be consistent with their design language.

Recommendations

OEMs should:

1. Use equivalent headers for screens provided by themselves to ensure visual consistency when moving forward and backward throughout SUW.

2. Use a color palette consistent with the theme chosen for SUW to ensure visual consistency throughout SUW.

3. Ensure all screens conform to these navigation and interaction guidelines and fit with these typography, keylines and spacing guides.

4. Have interstitials conform to these style guidelines and ensure animations between screens fit with these guidelines.

Implementing animations

The slide transition between screens can be expressed as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<translate xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="300"
    android:fromXDelta="100%"
    android:toXDelta="0%"
    android:interpolator="@android:anim/decelerate_interpolator"
    />
```

Implementing layouts

SUW layouts are implemented using the following XML.

Full-width card with header

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

xmlns:wizard="http://schemas.android.com/apk/res/com.google.android.setupwizard
"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <com.google.android.setupwizard.util.StickyHeaderScrollView
        android:id="@+id/bottom_scroll_view"
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
            <com.google.android.setupwizard.util.SetupWizardIllustration
                android:id="@+id/illustration_phone"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:elevation="@dimen/title_area_elevation"
                android:background="@drawable/illustration_bg"
                android:foreground="@drawable/illustration_generic"
                android:tag="stickyContainer"
                wizard:aspectRatio="2.22">

                <TextView
                    android:id="@+id/title"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:tag="sticky"
                    android:text="@string/setup_wizard_title"
                    style="@style/setup_wizard_header_title" />

            </com.google.android.setupwizard.util.SetupWizardIllustration>

            <FrameLayout
                android:id="@+id/setup_content"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_below="@id/illustration_phone" />

        </RelativeLayout>

    </com.google.android.setupwizard.util.StickyHeaderScrollView>

    <fragment
android:name="com.android.setupwizard.navigationbar.SetupWizardNavBar"
        android:id="@+id/navigation_bar"
        style="@style/setup_wizard_navbar_style" />

</LinearLayout>
```

Full-width card with collapsed header

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

    <RelativeLayout
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <FrameLayout
            android:id="@+id/title_area"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_alignParentTop="true"
            android:background="@drawable/illustration_bg"
            android:elevation="@dimen/title_area_elevation">

            <TextView
                android:id="@+id/title"
                style="@style/setup_wizard_header_title"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/setup_wizard_title" />
        </FrameLayout>

        <com.google.android.setupwizard.util.BottomScrollView
            android:id="@+id/bottom_scroll_view"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@id/title_area"
            android:layout_alignParentBottom="true">

            <FrameLayout android:id="@+id/setup_content"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                />
        </com.google.android.setupwizard.util.BottomScrollView>

    </RelativeLayout>

    <fragment
android:name="com.android.setupwizard.navigationbar.SetupWizardNavBar"
        android:id="@+id/navigation_bar"
        style="@style/setup_wizard_navbar_style" />
</LinearLayout>
```

Template card with header

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

    <com.google.android.setupwizard.util.SetupWizardIllustration
        android:id="@+id/illustration_tablet"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@drawable/illustration_bg"
        android:foreground="@drawable/illustration_tablet">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"

android:layout_marginTop="@dimen/setup_wizard_tablet_illustration_height"
            android:orientation="vertical"
            android:paddingLeft="@dimen/setup_wizard_card_port_margin_sides"
            android:paddingRight="@dimen/setup_wizard_card_port_margin_sides">

            <TextView
                android:id="@+id/title"
                style="@style/setup_wizard_card_title"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="@string/setup_wizard_title" />

            <com.google.android.setupwizard.util.BottomScrollView
                android:id="@+id/bottom_scroll_view"
                android:layout_width="match_parent"
                android:layout_height="0dp"
                android:layout_weight="1"
                android:background="@drawable/setup_wizard_card_bg"
                android:elevation="@dimen/setup_wizard_card_elevation">

                <FrameLayout
```

```
                    android:id="@+id/setup_content"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content" />

            </com.google.android.setupwizard.util.BottomScrollView>

        </LinearLayout>

    </com.google.android.setupwizard.util.SetupWizardIllustration>

    <fragment
        android:id="@+id/navigation_bar"
        android:name="com.android.setupwizard.navigationbar.SetupWizardNavBar"
        style="@style/setup_wizard_navbar_style" />

</LinearLayout>
```

Template card with side header

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

    <com.google.android.setupwizard.util.SetupWizardIllustration
        android:id="@+id/illustration_tablet"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:background="@drawable/illustration_bg"
        android:foreground="@drawable/illustration_tablet">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:gravity="start|top"
            android:weightSum="16">

            <TextView
                android:id="@+id/title"
                style="@style/setup_wizard_card_title"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
```

```
android:layout_marginTop="@dimen/setup_wizard_tablet_illustration_height"
                android:layout_weight="6"
                android:text="@string/setup_wizard_title"/>

            <com.google.android.setupwizard.util.BottomScrollView
                android:id="@+id/bottom_scroll_view"
                android:layout_width="0dp"
                android:layout_height="match_parent"

android:layout_marginTop="@dimen/setup_wizard_card_land_margin_top"
                android:layout_weight="8"
                android:background="@drawable/setup_wizard_card_bg"
                android:elevation="@dimen/setup_wizard_card_elevation">

                <FrameLayout android:id="@+id/setup_content"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content" />

            </com.google.android.setupwizard.util.BottomScrollView>

        </LinearLayout>

    </com.google.android.setupwizard.util.SetupWizardIllustration>

    <fragment
android:name="com.android.setupwizard.navigationbar.SetupWizardNavBar"
        android:id="@+id/navigation_bar"
        style="@style/setup_wizard_navbar_style" />

</LinearLayout>
```

Revision history

Date	Changes
1.20.2015	Added Factory Reset Protection integration details in "Creating a Setup application from scratch", step 1
1.29.2015	Added the description of the overlay mechanism in "Wrapping the Android SUW"
3.17.2015	Added notes in "Customization methods"

3.30.2015	Added guidelines for Android for work device owner provisioning