Customizing the Setup Wizard in Android 5.x+

Contents

Background

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 device setup and account configuration steps.

This document, and those referred from it, are instructions for creating a SUW, and include:

- How to easily create individual screens which are harmonious with the Android SUW look and feel, using the Wizard Library.
- How to easily create flows of screens, using the Wizard Manager.
- Design guidelines for screens which make up part of SUW.

Overview

The full SUW contains the following components:

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. Email setup
13. Play Auto Installs, which allow OEMs or carriers to customize initial home screens and install apps immediately after setup
14. Google opt-ins and terms of service (ToS)
15. Google Now opt-in (if Google Now Launcher is pre-installed)

Requirements

- 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.
- Branding of carrier, OEM or third party screens MUST be distinct from Google/Android branding, to avoid user confusion.
- Users MUST NOT be able to leave SUW by any means without being offered the ability to set up Factory Reset Protection; before being authenticated as part of Factory Reset Protection; or before having the opportunity to see Google Terms & Conditions and Privacy consents on the Google Services screen.

Customization methods

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

1. Use the Wizard Manager
2. Create a SUW 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 “Use the Wizard Manager” 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.**

Using the Wizard Manager

This is the preferred method for creating Setup Wizards as of Android M.

Wizard Manager is the part of the Android Setup Wizard that 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.

Two basic actions control Wizard Manager: LOAD, and NEXT. The LOAD action is sent by SUW to Wizard Manager with a link to the script that will be used. The NEXT action delivers the result of the current action back to Wizard Manager, which then launches the next scripted activity. See “Customizing the Setup Wizard in Android 5.x” for notes on how to load a custom script or to start the NEXT action.

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.

Overlaying the Android SUW

Starting from Lollipop MR1, setup wizard introduced a new way for customization. Using this mechanism, partners can add an additional APK in the system image that provides assets in specific extension points built-in to setup wizard in order to customize things like the Wizard Script, welcome screen background and theme.

Declaring the Overlay APK

In order to declare the overlay APK, the APK needs to declare a broadcast receiver for the intent action `com.android.setupwizard.action.PARTNER_CUSTOMIZATION`. The broadcast receiver does not have to perform any action, it is just used by setup wizard to find the APK by asking `PackageManager` to resolve the intent. For this reason, the broadcast receiver 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>
```

Note that the overlay APK must be a system package, with `ApplicationInfo.FLAG_SYSTEM` in order for setup wizard to recognize it.

Providing resource assets

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

Overlayable resources

`@plurals/sim_missing_text`
Text shown when no SIM is detected in the device. This is of type plurals to allow support for multi-SIM devices, which will select the string based on the number of SIM slots. This string should help the user figure out how to insert their SIM into the device.

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.

`@string/theme_type`
The theme to be used in Setup Wizard. Possible values are "material" and "material_light".

`@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`.

`@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`
See the wizard script section for more info on how to customize the script.

`@string/wizard_script_user_uri`
An android resource URI pointing to the modified wizard script for secondary users (i.e. users added via Settings > Users > Add User).
e.g. `android.resource://com.my.package/raw/custom_wizard_script_user`
See the wizard script section for more info on how to customize the script.

`@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.

`@raw/smartdevice_d2d_target_nfc_alternative_video`
(GMS Core Tala 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 Supported Media Format video.

Customizing wizard script

By providing the values for `@string/wizard_script_uri` or `@string/wizard_script_user_uri`, additional screens can be added or existing screens can be replaced. Please refer to the design specs for the appropriate places for adding and customizing screens, as there are steps that should not be moved or replaced.
First note that the wizard scripts are provided in URI format. An example of that will be `android.resource://com.my.package/raw/custom_wizard_script`. See here for documentation on the format of the URI.

Second, in the overlay package, the custom wizard script should use the raw resource type instead of the XML resource type. This is because the custom wizard script should not be compiled by AAPT in order to retain the setup wizard namespace.

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, change the result tag of the corresponding result to point to your newly added action, and then set the result tag of your new action to the next action it should go to.

For example, here is an excerpt of the 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`.

```
<!-- 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 an existing 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

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. For simplicity, all of this is encapsulated in `WizardManagerHelper.getNextIntent()`, which will return an intent to be started with `startActivityForResult`. See *Setup Wizard Library User Guide* for more information on `WizardManagerHelper`.

```
public class MyActivity extends Activity {

    private static final int NEXT_REQUEST_CODE = 1; // Can be anything

    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()`.

Scripted activities

Activites 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 are `RESULT_SKIP` (1) and `RESULT_RETRY` (2). `RESULT_CANCELED` is sent when navigating back to the previous screen and is not processed by Wizard Manager. To avoid conflict with future SUW standards, custom result codes should increment 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.

Debug logging

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

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

System properties

Parallel to the overlay mechanism, setup wizard also uses system properties to allow for some functional customizations. These properties can be customized in the device makefile through the `PRODUCT_PROPERTY_OVERRIDES` variable.

`ro.setupwizard.hfa_always`
Boolean flag, when set, setup wizard will always invoke HFA activation, regardless of the current reported activation state. Default value is false.

`setupwizard.logging`
Boolean flag, which will enable or disable anonymous logging of setup wizard usage. Default value is true.

`ro.setupwizard.suppress_d2d`
Boolean flag, which will suppress device-to-device setup (e.g. Tap & Go) if true. Default value is false.

`ro.setupwizard.suppress_d2d_nfc`
(Available since GMS core Tala) Boolean flag, which will suppress NFC for device-to-device setup. This means that for phones the instructions for device-to-device setup via Google Settings app will be shown instead.

`ro.setupwizard.require_network`
`ro.setupwizard.user_req_network`
A string value that 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.

Standard scripts

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

Creating a SUW from scratch

We strongly advise OEMs and carriers to use the Wizard Manager method above, instead of relying on creating their own SUW. Over the last few releases, the number of integrations required between SUW and Android to ensure a secure SUW with a good experience has increased. Android accordingly focuses its development efforts on the overlay method, and away from supporting this legacy mechanism. **This method will be deprecated in the near future.**

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.

3. **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);
```

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

```
        loadFrpStatus();
    }

    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;
        }
    }
```

4. 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);
```

5. 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`

6. 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()`

   Although it is recommended to the user a way to pick the backup set to restore from, OEMs which do not, should use the first entry in the array returned by `getAvailableRestoreSets()`. This is the recommended dataset by the Google backend for setup time restore for the device.

   To implement a screen allowing users to select a backup set, OEMs can use the RestoreWizardSample code.

8. If you wish to disable Tap & Go, please call AccountManager#addAccount() directly from your custom SUW, and add a boolean "suppress_device_to_device_setup" set to true in the options bundle.

Android for Work

Custom setup wizard implementations must comply with these requirements in order to ensure that Android for Work devices can be provisioned successfully.

Google Account Based Setup

With the 'M' release of Android, users adding a managed Google account to the device will be prompted to download and install the management application belonging to their EMM, which will guide users through additional setup steps.

Once installed, the management application will be able to set up devices in all modes of Android for Work management, including device owner, profile owner and device admin. Once the management mode has been selected, the rest of the provisioning process takes place as normal. It is important to note that device owner provisioning will expect to close SUW once it completes. Please follow the NFC provisioning guidance in order to insert any essential UI in this flow.

Non-device owner modes will not close the SUW and will return to it once completed.

Device Owner Mode

Android for Work device owner mode must be provisioned from the device setup wizard as a security precaution, this can either happen 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 any user interaction required in setup wizard.

When provisioning corporate owned devices in device owner mode, the setup wizard needs to check when it opens whether 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 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();
             }
         …
       }
```

NFC Provisioning

Devices with NFC must enable this throughout SUW in order to allow for NFC provisioning. 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) with no 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:

For custom setup wizards:

If the setup wizard 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. The buttons on these UI should have meaningful text or content description, such as “Skip”, “Next”, “Accept”, “Finish”, “Agree” or “Continue” to navigate the next page.

For extensions of the standard Google Setup Wizard, see here.

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 which 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 another HOME intent is fired to return to the device setup flow.

Locked Task Mode

SUW implementations using locked task mode for any compulsory sections must finish this before entering the Google account section. This is in order to allow for possible download and install of the Device Policy Client (DPC), which will be launched from inside SUW.

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

Play Auto Install

Please note that since Device Owner mode is designed for fully managed devices, Play Auto Install for applications will be suppressed when the Android Managed Provisioning process sets the USER_SETUP_COMPLETE flag. Play Auto Install is likewise unavailable when setting up Android for Work Managed Profiles. In both cases applications are available for install and management through Play for Work, managed by the user's organization.

Finishing SUW

It is important that SUW implementations finish by invoking the HOME intent no more than once. This is to ensure that the DPC can finish any required setup after it is set as the Device Owner.

Testing Android for Work Features

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

Setup Wizard 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.

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.

- 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

Customized SUWs should use the same layouts as the Android SUW screens, in order to give users a harmonious setup experience. The Wizard Library helps implement these layouts. The Wizard Manager lets you gather screens together into a setup flow.

See the Setup Wizard design guidelines for more details of color palettes, screen layouts and recommended styles.

Implementing animations

Screens implemented with the Wizard library will inherit the correct slide transitions automatically.

The slide transition between screens can be expressed by hand 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

We strongly recommend that SUW screens be implemented using the Wizard Library. This will ensure that they are sized and laid out correctly on various device sizes.

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
7.10.2015	Added references to new design, Wizard Manager, Wizard Library docs
7.14.2015	Pre-release tidy-up, removed deprecated append/prepend method
7.17.2015	Minor copy fixes
9.4.2015	Added overlay section

References

- Setup Wizard design guidelines
- Wizard Library documentation