Restoring User Data on New Devices

In this document

Users often invest significant time and effort creating an identity, adding data and customizing settings and preferences within your app. Preserving this data and personalization for users when they upgrade to a new device or replace a broken one is an important part of ensuring a great user experience. This section covers techniques for backing up data to the cloud and provides information on migrating identity, app data, settings data, and permissions for returning users.

Make sure you minimize the number of steps required for a user to pick up where they left off on their previous device. You can gain a number of potential benefits by ensuring a great user experience for returning users on new devices:

- Reduce user frustration.
- Increase login-rate.
- Reduce support calls.
- Minimize user attrition.
- Sustain user engagement.
- Increase user retention rate.

You can help maintain existing user engagement on a new device by providing a seamless login experience. You can integrate Smart Lock for Passwords (https://developers.google.com/identity/smartlock-passwords/android/) into your Android app to restore user sign-ins on a device. Smart Lock for Passwords supports saving both username-password credentials and federated identity provider credentials. You can also use the Account Transfer API (https://developer.android.com/guide/topics/data/account-transfer.html) to copy over custom account credentials for your app from a user's existing device to a new device. The transfer takes place during setup of the new device.

App data may include user-generated content, such as text, images, and other media. For restoring app data, see Transferring Data Using Sync Adapters (https://developer.android.com/training/sync-adapters/index.html) or Google Drive Android API (https://developers.google.com/drive/android/). You can use either approach to synchronize app data between Android-powered devices and save data which you'd like to use during the normal app lifecycle. You can also use either approach to restore a returning user's data onto a new device.

Make sure you also back up and restore settings data to preserve a returning user's personalized preferences on a new device. You can restore settings data even if a user doesn't log in to your app. You can back up settings that a user explicitly sets in your app's UI, as well as transparent data, such as a flag indicating whether a user has seen a setup wizard. For a description of the settings to back up, see Preserving settings data (https://developer.android.com/guide/topics/data/backup.html#Preserving). For a comparison of the available backup solutions for settings data, see Choosing a backup solution for app settings (https://developer.android.com/guide/topics/data/backup.html#Choosing).

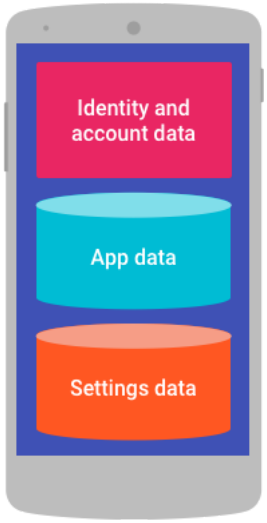

Figure 1. Make sure you restore identity, app data, and settings data for users returning to your app.

> **Note:** Any permissions a user grants to your app are automatically backed up and restored by the system on devices running Android 7.0 (API 24) or newer. However, if a user uninstalls your app, then the system clears any granted permission and the user must grant them again.

Preserving settings data

To preserve as much of an existing user's experience on a new device as possible, make sure you back up the following user settings:

- Any settings modified by the user, for example using a `PreferenceScreen` (https://developer.android.com/reference/android/preference/PreferenceScreen.html).
- Whether the user has turned notification and ringtones on or off.
- Boolean flags which indicate if the user has seen welcome screens or introductory tooltips.

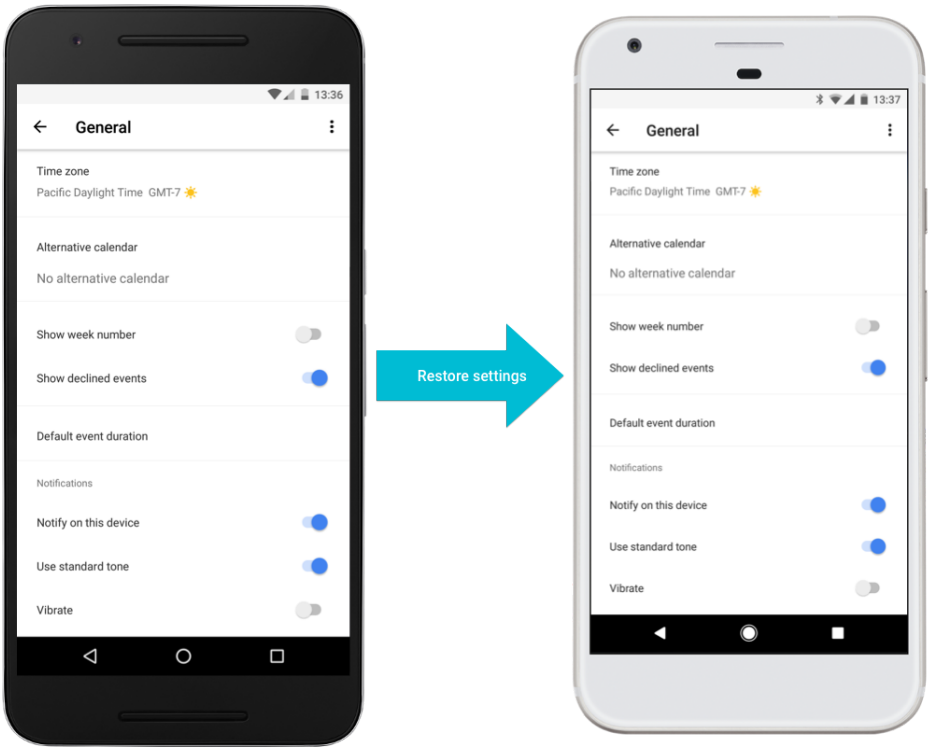

Figure 2. Restoring settings on new devices ensures a great user experience.

You must not back up URIs for ringtones because URIs are unstable. In some cases a restoration to a new mobile device may result in a URI which points to no ringtone, or a different ringtone from the one intended. Instead, you can back up ringtones using either their title or a hash of the ringtone.

Choosing a backup approach for app settings

Android provides two ways for apps to back up their data to the cloud: Auto Backup for Apps (https://developer.android.com/guide/topics/data/autobackup.html) and Key/Value Backup (https://developer.android.com/guide/topics/data/keyvaluebackup.html). Auto Backup, which is available starting Android 6.0 (API level 23), preserves data by uploading it to the user's Google Drive account. Auto Backup includes files (https://developer.android.com/guide/topics/data/autobackup.html#Files) in most of the directories that are assigned to your app by the system. Auto Backup can store up to 25 MB of file-based data per app. The Key/Value Backup feature (formerly known as the Backup API and the Android Backup Service) preserves settings data in the form of key/value pairs by uploading it to the Android Backup Service (https://developer.android.com/google/backup/index.html).

Generally, we recommend Auto Backup because it requires no work to implement. Apps that target Android 6.0 (API level 23) or higher are automatically enabled for Auto Backup. The Auto Backup feature is a file-based approach to backing up app data. While Auto Backup is simple to implement, you may consider using the Key/Value Backup feature if you have more specific needs for backing up data.

> **Note:** If your app doesn't have a backup mechanism for app contents and the size of your app contents is unlikely to exceed the 25 MB limit, then Auto Backup may be sufficient for your needs.

The following table describes some of the key differences between Key/Value Backup and Auto Backup:

Category	Key/Value Backup	Auto Backup
Android API level	Available in API 8, Android 2.2 and higher.	Available in API 23, Androic
Implementation	Apps must implement a `BackupAgent` (https://developer.android.com/reference/android/app/backup/BackupAgent.html). The backup agent defines what data to back up and how to restore data.	By default, Auto Backup inc (https://developer.android.com/gu bundled into the SDK.
Frequency	Apps must issue a request when there is data that is ready to be backed up. Requests from multiple apps are batched and executed every few hours.	Backups happen automatic
Transmission	Backup data can be transmitted via Wi-Fi or cellular data.	Backup data is tranmitted c
App shut down	Apps are not shut down during backup.	The system shuts down the
Backup storage	Backup data is stored in Android Backup Service (https://developer.android.com/google/backup/index.html) and limited to 5MB per app. Google treats this data as personal information in accordance with Google's Privacy Policy	Backup data is stored in the accordance with Google's F

	(https://www.google.com/privacypolicy.html).	
User login	Doesn't require a user to be logged into your app. The user must be logged into the device with a Google account.	Doesn't require a user to be
API	Related API methods are entity-based: • `onBackup()` (https://developer.android.com/reference/android/app/backup/BackupAgent.html#onBackup(android.os.ParcelFileDescriptor, android.app.backup.BackupDataOutput, android.os.ParcelFileDescriptor)) • `onRestore()` (https://developer.android.com/reference/android/app/backup/BackupAgent.html#onRestore(android.app.backup.BackupDataInput, int, android.os.ParcelFileDescriptor))	Related API methods are fil • `onFullBackup()` (https://developer.andro • `onRestoreFile()` (https://developer.andro java.io.File, int, long,

Note: If Wi-Fi isn't available, Key/Value Backup may use mobile data. Key/Value Backup is therefore typically not suitable for app data contents, such as media, downloaded files, and caches, unless the amount of data is very small.