OTA Updates

Android devices in the field can receive and install over-the-air (OTA) updates to the system and application software. Devices have a special recovery partition with the software needed to unpack a downloaded update package and apply it to the rest of the system.

This section describes the structure of these packages and the tools provided to build them. It is intended for developers who want to make the OTA update system work on new Android devices and those who are building update packages for use with released devices. OTA updates are designed to upgrade the underlying operating system and the read-only apps installed on the system partition; these updates do *not* affect applications installed by the user from Google Play.

This section describes the OTA system as of the Android 5.x release. For help porting OTA-related code from older releases, see Migrating from previous releases (#migrating).

Android device layout

The flash space on an Android device typically contains the following partitions.

boot

Contains the Linux kernel and a minimal root filesystem (loaded into a RAM disk). It mounts system and other partitions and starts the runtime located on the system partition.

system

Contains system applications and libraries that have source code available on Android Open Source Project (AOSP). During normal operation, this partition is mounted read-only; its contents change only during an OTA update.

vendor

Contains system applications and libraries that do *not* have source code available on Android Open Source Project (AOSP). During normal operation, this partition is mounted read-only; its contents change only during an OTA update.

userdata

Stores the data saved by applications installed by the user, etc. This partition is not normally touched by the OTA update process.

cache

Temporary holding area used by a few applications (accessing this partition requires special app permissions) and for storage of downloaded OTA update packages. Other programs use this space with the expectation that files can disappear at any time. Some OTA package installations may result in this partition being wiped completely.

recovery

Contains a second complete Linux system, including a kernel and the special recovery binary that reads a package and uses its contents to update the other partitions.

misc

Tiny partition used by recovery to stash some information away about what it's doing in case the device is restarted while the OTA package is being applied.

Life of an OTA update

A typical OTA update contains the following steps:

1. Device performs regular check in with OTA servers and is notified of the availability of an update, including the URL of the update package and a description string to show the user.
2. Update downloads to a cache or data partition, and its cryptographic signature is verified against the certificates in `/system /etc/security/otacerts.zip`. User is prompted to install the update.

3. Device reboots into recovery mode, in which the kernel and system in the recovery partition are booted instead of the kernel in the boot partition.
4. Recovery binary is started by init. It finds command-line arguments in `/cache/recovery/command` that point it to the downloaded package.
5. Recovery verifies the cryptographic signature of the package against the public keys in `/res/keys` (part of the RAM disk contained in the recovery partition).
6. Data is pulled from the package and used to update the boot, system, and/or vendor partitions as necessary. One of the new files left on the system partition contains the contents of the new recovery partition.
7. Device reboots normally.
    a. The newly updated boot partition is loaded, and it mounts and starts executing binaries in the newly updated system partition.
    b. As part of normal startup, the system checks the contents of the recovery partition against the desired contents (which were previously stored as a file in `/system`). They are different, so the recovery partition is reflashed with the desired contents. (On subsequent boots, the recovery partition already contains the new contents, so no reflash is necessary.)

The system update is complete!

Migrating from previous releases

When migrating from Android 2.3/3.0/4.0 release, the major change is the conversion of all the device-specific functionality from a set of C functions with predefined names to C++ objects. The following table lists the old functions and the new methods that serve a roughly equivalent purpose:

C function	C++ method
device_recovery_start()	Device::RecoveryStart()
device_toggle_display() device_reboot_now()	RecoveryUI::CheckKey() (also RecoveryUI::IsKeyPressed())
device_handle_key()	Device::HandleMenuKey()
device_perform_action()	Device::InvokeMenuItem()
device_wipe_data()	Device::WipeData()
device_ui_init()	ScreenRecoveryUI::Init()

Conversion of old functions to new methods should be reasonably straightforward. Don't forget to add the new `make_device()` function to create and return an instance of your new Device subclass.

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 3.0 License (http://creativecommons.org/licenses/by/3.0/)*, and code samples are licensed under the Apache 2.0 License* (http://www.apache.org/licenses/LICENSE-2.0)*. For details, see our Site Policies* (https://developers.google.com/terms/site-policies)*. Java is a registered trademark of Oracle and/or its affiliates.*

Last updated May 1, 2017.