App Manager

User Manual

v4.0.5

August 21, 2025

Copyright © 2020–2025 Muntashir Al-Islam

“Wisely and slow. They stumble that run fast.” — Friar Laurence, *Romeo and Juliet*

Contents

Chapter 1

Introduction

App Manager is an advanced package manager for Android. It offers numerous features and thus, requires a user manual to assist its users. This document acts as a user manual for App Manager in the sense that it aims to describe every feature that App Manager has to offer. This document also serves as the "official" guidelines for App Manager and represents the expected behavior of App Manager. Translations can misinterpret this document (which is in English). Therefore, a capable user should read the English version of the document to get the most out of it. There may as well be other unofficial or third-party resources, such as blog articles, videos, forums, and chat groups. While these resources may be useful to many people, they may not be up-to-date with the latest version of App Manager. In case of any deviations detected in App Manager from this document, they should be reported in the App Manager issue tracker.

1.1 Terminologies

- **AM** — Short name for App Manager.
- **Block/Unblock** — Used for component blocking or unblocking. How components are blocked depends on the user preferences.
- **IFW** — Short form of Intent Firewall.
- **Ops** — Short name for operations, e.g. app ops, batch ops, 1-click ops
- **SAF** — Short for Storage Access Framework, an abstraction used by Android to allow apps to use or serve files without worrying about the underlying file system.
- **SSAID** — Short form of `Settings.Secure.ANDROID_ID`. It is a device identifier assigned to each app (Android Oreo onwards). It is generated from the combination of the signing certificate of the app and the SSAID set for the package `android`. As a result, it is guaranteed to be the same for an app unless the user chooses to format the device. It is widely used for tracking.
- **Tracker** — Denotes tracker components throughout the document and in App Manager except in the scanner page. Trackers include libraries such as crash reporters, analytics, profiling, identification, ad, location, etc. Thus, they are not equal in functions. There is no distinction or bias among the open-source and closed-source libraries that promote tracking.

1.2 Supported Versions

At present, the supported version is v4.0.1. Previous versions of App Manager may contain security vulnerabilities and should not be used.

1.3 Official Sources

1.3.1 Binary Distribution Sources

App Manager is distributed using the following sources. Unofficial sources may distribute modified versions of App Manager, and you alone shall be responsible for the consequences of using such a distribution.

1. Official F-Droid repository.[1]
   Link: `https://f-droid.org/packages/io.github.muntashirakon.AppManager`
2. GitHub repository.
   Normal releases: `https://github.com/MuntashirAkon/AppManager/releases`
   Debug releases: `https://github.com/MuntashirAkon/AMInsecureDebugBuilds`
3. Telegram.
   Normal releases: `https://t.me/AppManagerChannel`
   Debug releases: `https://t.me/AppManagerDebug`

1.3.2 Links to the Source Code

All but GitHub are mirrors. The tags should always be up-to-date, but the master branch may not. If you want to clone the master branch, use the GitHub link instead of the others.

1. GitHub: `https://github.com/MuntashirAkon/AppManager`
2. Codeberg: `https://codeberg.org/muntashir/AppManager`
3. GitLab: `https://gitlab.com/muntashir/AppManager`
4. Riseup: `https://0xacab.org/muntashir/AppManager`
5. sourcehut: `https://git.sr.ht/~muntashir/AppManager`

1.3.3 Translations

App Manager does not accept translations via pull/merge requests. Translations are managed through Hosted Weblate. To translate App Manager, visit `https://hosted.weblate.org/engage/app-manager/`. Please read the **Info** section before getting started.

1.4 Contributing

There are many ways a user can contribute, such as creating helpful issues, attending discussions, improving documentation and translations, reporting unknown libraries or trackers, reviewing the source code, and reporting security vulnerabilities.

1.4.1 Build Instructions

Build instructions are available in the BUILDING file located in the source root.

1.4.2 Submitting patches

Repositories located on sites other than GitHub are currently considered mirrors, and pull/merge requests submitted on those sites will not be accepted.[2] Instead, patches (as `.patch` files) can be submitted via email attachments. *Signing-off is a requirement.* See the CONTRIBUTING file located at the source root for more information.

[1] For distributing normal releases only

[2] GitHub pull requests will be merged manually using the corresponding patches. As a result, GitHub may wrongfully mark them closed instead of merged.

Notice. In the case of submitting patches via email, the whole conversation may be publicly accessible in the future. So, please do not include personally identifiable information (PII) other than your name or email address.

1.5 Donation & Funding

As of September 2024, App Manager is not accepting financial support until further notice. But you may still be able to send gifts (e.g., gift cards, subscriptions, food and drink, flowers, or even cash). Please reach out to the maintainer using the options given in §**??** for further assistance.

In addition, the maintainers and contributors of this project DO NOT consent to the creation, sale, or promotion of tokens, cryptocurrencies, NFTs, or any other financial instruments that claim to represent this project, its code, or its community. Any such attempts are unauthorized and not affiliated with this project in any way.

1.6 Contact

App Manager Community
Email: am4android [at] riseup [dot] net
GitHub: `https://github.com/AMCommunity`
Twitter/X: `https://x.com/AppManagerNews`
Mastodon: @appmanager@floss.social

Muntashir Al-Islam[3]
Email: muntashirakon [at] riseup [dot] net
GitHub: `https://github.com/MuntashirAkon`
Twitter/X: `https://x.com/Muntashir`
Mastodon: @muntashir@infosec.exchange

[3] You can also address me as "Muntashir Akon"

Chapter 2

Pages

2.1 Main Page

Main page lists all the installed, uninstalled and backed up applications. A single click on any installed application item opens the respective App Details page. For the uninstalled system applications, a dialog prompt is displayed with an option reinstall them. The applications uninstalled without removing their data and signatures are also displayed in this page with an option to perform a full uninstallation. For the uninstalled applications having one or more backups, the restore dialog is displayed. Using the sort option from the list options, the items can be sorted in various ways. It is also possible to filter items using the filter option in the list options. Filtering is also possible from the search bar with additional support for the regular expressions.

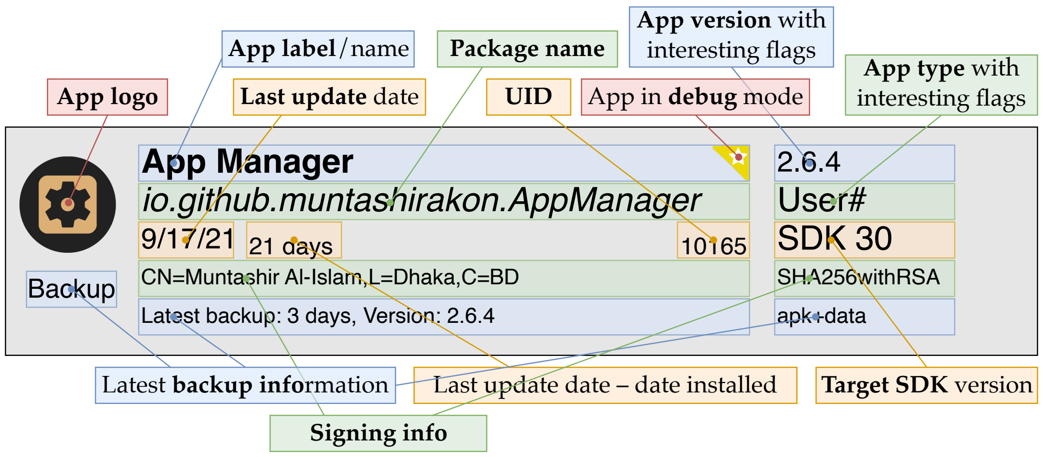
App label / name
Package name
App version with interesting flags
App logo
Last update date
UID
App in debug mode
App type with interesting flags
App Manager
io.github.muntashirakon.AppManager
9/17/21
21 days
10165
2.6.4
User#
SDK 30
CN=Muntashir Al-Islam,L=Dhaka,C=BD
SHA256withRSA
Backup
Latest backup: 3 days, Version: 2.6.4
apk+data
Latest backup information
Last update date – date installed
Target SDK version
Signing info

Figure 2.1: An application list item in the main page

2.1.1 Batch Operations

Batch operations or operations on multiple applications are also available within this page. Multiple selection mode can be activated by clicking on the app icon of an item or by long-clicking on any items in the list. Once activated, a click on an item selects it instead of opening the App Details page. In this mode, the batch operations are located in the multiple selection menu at the bottom of the page. The operations include:

- Adding the selected applications to a profile
- Backing up applications, or restoring and deleting the existing backups

- Blocking the trackers from the applications
- Clearing data or cache from the applications
- Exporting the blocking rules configured inside App Manager
- Exporting the list of applications in Markdown, CSV, JSON or XML format
- Freezing/unfreezing/force-stopping/uninstalling the applications
- Performing run-time optimization of the applications (Android 7 onwards)
- Preventing the background operations of the applications (Android 7 onwards)
- Saving the APK files to `AppManager/apks`
- Setting net policies

> ***Accessibility.*** After the multiple selection mode has been activated, it is possible to navigate in or out of the multiple selection menu using the right or left keys of the keyboard or remote.

2.1.2 Colour Codes

- Red (day) / dark red (night) – Uninstalled application
- Light red (day) / very dark red (night) – Frozen application
- Dark cyan – Force-stopped application
- Yellow Star – Debuggable application
- Orange *Date* – The application can read system logs
- Orange *UID* – The user ID is being shared among multiple applications
- Orange *SDK* – The application possibly uses cleartext (i.e. HTTP) traffic
- Light orange *package name* – The application has one or more trackers
- Red *app label* – The application does not allow clearing its data
- Dark cyan *version* – Inactive application
- Magenta type – Persistent application i.e. it remains running all the time
- Red *backup* – The uninstalled application with one or more backups present in App Manager
- Orange *backup* – Outdated backup, i.e. the base backup contains an older version of the installed application
- Dark cyan *backup* – Up to date backup, i.e. the base backup contains the same or higher version of the installed application.

2.1.3 Application Types

An application can be either a **User** or a **System** application along with the following suffixes:

- `X` – Supports multiple architectures
- `0` – No dex files present in the application
- `°` – Suspended application
- `#` – The application requested the system to allocate a large heap i.e. large runtime memory
- `?` – The application requested the virtual machine to be in the safe mode.

2.1.4 Version Info

Version name is followed by the prefixes below:

- `_` – No hardware acceleration (breaking the in-app animations or transparencies)
- `~` – Test-only application
- `debug` – Debuggable application

2.1.5 Options Menu

Options menu offers several options that can be used to sort and filter the listed applications as well as to navigate to different pages within or outside App Manager.

2.1.5.1 List Options

List options contain the options to sort and filter the list in the main page.

Sort The applications listed in the main page can be sorted in the following ways:

- **User apps first.** The user applications are listed on top
- **App label.** Sort the list in ascending order based on their application labels (also known as *application names*). This is the default sorting preference
- **Package name.** Sort the list in ascending order based on their package names
- **Last update.** Sort the list in descending order based on the date they were last updated
- **Shared user ID.** Sort the list in descending order based on their kernel user ID
- **Target SDK.** Sort the list in ascending order based on their target SDK
- **Signature.** Sort the list in ascending order based on their signing information
- **Frozen first.** The frozen applications are listed on the top
- **Blocked first.** Sort the list in descending order based on the number of blocked components each application has
- **Backed up first.** Display the applications with backups on the top
- **Trackers.** Sort the list in descending order based on the number of tracker components each application has
- **Last actions.** Sort the list in descending order based on the latest time and date of any actions made to the applications within App Manager.

- **Installation date.** Sort the list by the date of installation in descending order.
- **Total size.** Sort the list by the total size of the applications and their data in descending order. Requires `Usage Access` permission.
- **Data usage.** Sort the list by the total data usage in descending order. Requires `Usage Access` permission.
- **Times opened.** List frequently used applications on top. Requires `Usage Access` permission.
- **Screen time.** List the applications with the highest engagements on top. Requires `Usage Access` permission.
- **Last used.** List last used apps on top. Requires `Usage Access` permission.

In addition, there is the *reverse* option that can be used to sort the list in the reverse order. Regardless of the sorting preferences, the applications are sorted alphabetically at first in order to prevent producing any random sorting results.

Filter The applications listed in the main page can be filtered in the following ways:

- **User apps.** List only the user applications
- **System apps.** List only the system applications
- **Frozen apps.** List only the frozen applications
- **Stopped apps.** List only the applications that were forced-stopped
- **Installed apps.** List only the installed applications
- **Uninstalled apps.** List only the uninstalled applications
- **With rules.** List the applications with one or more blocking rules
- **With activities.** List the applications with one or more activities
- **With backups.** List the applications with one or more backups
- **Without backups.** List the applications with no backups present.
- **Running apps.** List the applications that are currently running
- **With splits.** List the applications with one or more split APK files
- **With KeyStore.** List only the applications with Android KeyStore.
- **With SAF.** List only the applications with SAF access.
- **With SSAID.** List only the applications with a valid SSAID.

Unlike sorting, it is possible to apply more than one filtering options at the same time. For example, the frozen user applications can be listed by selecting both *User apps* and *Frozen apps*. This can be particularly useful for batch operations where filtering the user applications may be necessary to carry out certain operations safely.

> ***Inconsistencies.*** App Manager extensively caches everything in this page. Therefore, certain states (e.g., freeze and stopped states) may not always be up-to-date.

Profile Name It is also possible to list the applications that are only present in a profile. This can be useful for carrying out certain operations on a profile (e.g., uninstalling all the applications in a profile) that cannot be done via the Profiles page.

2.1.5.2 User Manual

Clicking on the **User manual** opens the offline version of the App Manager user manual. It may also open the online version if the corresponding feature split i.e. `feat_docs` is not installed, or if an WebView is not present in the system to load the manual.

2.1.5.3 1-Click Ops

1-Click Ops stands for the single-click operations. It opens the corresponding page in a new activity.

2.1.5.4 App Usage

Application usage statistics, such as *screen time*, *data usage* (both mobile and Wi-Fi), *number of times an app was opened*, can be accessed by clicking on the **App Usage** option in the menu. However, it requires the *Usage Access* permission. This menu item will not be listed if the usage access feature is disabled in settings.

2.1.5.5 Running Apps

This menu item opens a new page where a list of running applications or processes are displayed. It also displays the current memory and cache (if available) usage. If root or ADB is not available to App Manager, it only displays itself in the recent versions of Android. The running applications or processes can also be force-stopped or killed within the resultant page. Logs for each process ID (PID) can also be viewed in the log viewer. In addition, it is also possible to carry out batch operations either by clicking on the icon or by long-clicking on an item. Normal click on any items opens a dialog where a more detailed information is displayed.

2.1.5.6 Profiles

This menu item opens the profiles page. Profiles are a way to configure regularly used tasks. They can also be invoked via shortcuts.

2.1.5.7 APK Updater

If the app APK Updater is installed in the system, it can be opened directly via this menu item. The option remains hidden if the app is not present in the system.

2.1.5.8 Termux

If the app Termux is installed in the system, the running session (or a new session) can be opened directly via this menu item. The option remains hidden if the app is not present in the system.

2.1.5.9 Debloater

This menu item opens the debloater page that lists all the bloatware available in the device and in App Manager. It also suggests alternative applications based on the criteria set by the Android Debloat List project.

2.1.5.10 Labs

This menu item opens the labs page that lists all the additional features. They include Log Viewer, System Config, Terminal, File Manager (as Files), UI Tracker, Interceptor, and Code Editor pages.

2.1.5.11 Settings

This menu item opens the in-app Settings page.

2.2 App Details Page

App Details page consists of 11 (eleven) tabs. It describes almost every bit of information an application usually has, including all attributes from its manifest, application operations (app ops), signing information, libraries, and so on.

2.2.1 Colour Codes

List of colours used in this page, and their meaning:

- Red (day) / dark red (night) – Denotes any app ops or permissions having the dangerous flag, or any components blocked within App Manager, or any unsupported but required features.
- Light red (day) / very dark red (night) – Denotes the components disabled outside of App Manager, or any unsupported but optional features.

> ***Note.*** A component marked as disabled does not always mean that it is disabled by the user: It could also be disabled by the system or marked as disabled in its manifest. The components of a disabled application are also considered disabled by the system (and App Manager).

- Vivid orange (day) / very dark orange (night) – Denotes the tracker components
- Soft magenta (day) / very dark violet (night) – Denotes the running services.
- Green – Used in the tracker-indicator tag to denote that all the trackers in the application are blocked.

2.2.2 App Info Tab

App Info tab contains general information about an application. It also lists many actions that can be performed within this tab.

2.2.2.1 General Information

The list below is in the same order as listed in the App Info tab.

- **Application Icon.** The application icon. If the application does not have an icon, the system default icon will be displayed. It is also possible to verify the APK signature via SHA or MD5 sums stored in the clipboard by simply clicking on it.
- **Application Label.** The application label or the name of the application.
- **Package Name.** The name of the application package. Clicking on the name stores it in the clipboard.
- **Version.** The application version is divided into two parts. The first part is called *version name*. The format of this part varies but often consists of multiple integers separated by dots. The second part is called *version code*. It is enclosed by the first brackets. The version code is an integer used to differentiate between application versions (since a version name can be unreadable to a machine). In general, a new version of an application has higher version code than the old ones. For example, if `123` and `125` are two version codes of an application, we can say that the latter is more updated than the former because the version code of the latter is higher. An application that serves different APK files for the same version on different platforms (mobile, tabs, desktops, etc.) or architectures (32/64 bit, ARM or Intel), the version numbers can be misleading as they often add prefixes for each platform.
- **Tags.** (Also known as tag clouds) Tags include the most basic, concise and useful information of an application. See §**??** for a complete list of tags shown here.

- **Horizontal Action Panel.** An action panel consisting of various actions that can be carried out for the application. See §?? for a complete list of actions available here. Additional actions are available in the options menu.

- **Paths & Directories.** Contains various information regarding application paths including *app directory* (where the APK files reside), *data directories* (internal, device protected and externals), and *JNI library directory* (if present). JNI libraries are used to invoke native codes usually written in C/C++. Use of native library can make the application run faster or help an application use third-party libraries written using languages other than Java like in most games. The directories can be opened via file managers provided they support it and have the necessary permissions, by clicking on the launch button on the right-hand side of a directory item.

- **Data Usage.** Amount of data used by the application as reported by the operating system. Depending on Android version, this may require a wide range of permissions including *Usage Access* and *Telephony* permissions.

- **Storage & Cache.** Displays information regarding the size of the application (APK files, optimised files), data and cache. In older devices, size of external data, cache, media and OBB folders are also displayed. This part remains hidden if *Usage Access* permission is not granted in the newer devices.

- **More Info.** Displays other information such as–

  - **SDK.** Displays information related to the Android SDK: *Max* denotes the target SDK and *Min* denotes the minimum SDK (the latter is not available in Android Lollipop). If the target SDK value is less than the platform SDK (i.e., the highest SDK the current operating system supports), the application will run in the compatibility mode. This means the application may have access to certain features that are unavailable or restricted in a newer version of Android, which can be a security and/or privacy issue. SDK is also known as **API Level**.
    See also: Android Version History
  - **Flags.** The application flags used at the time of building the application. For a complete list of flags and what they do, read the official documentation.
  - **Date Installed.** The date when the application was first installed.
  - **Date Updated.** The date when the application was last updated. This is the same as *Date Installed* if the application hasn't been updated.
  - **Process Name.** The name of the process if it is different from the package name. Process name is set when an application is being started by the system, and is usually the same as the package name.
  - **Installer App.** The application that installed this application. The installer application may not always be the same as the application that installed this application, because Android allows setting an arbitrary value for this field. In Android 11 onwards, the actual installer application is also stored by the system which can be accessed by clicking the "Info" button on the right-hand side of the item. The field will not be visible if the installer application is not reported by the system (e.g., due to the installer application being uninstalled or hidden). Installer application may be granted additional privileges by the system so that it can control certain behaviour of the application it installs.
  - **User ID.** The unique user ID set by the system to the application. For shared applications, the same user ID is assigned to multiple applications having the same *Shared User ID*.
  - **Shared User ID.** Applicable for applications that are shared together. The shared application must have the same signatures.
  - **Primary ABI.** Architecture supported by this platform for this application.
  - **Zygote preload name.** Responsible for preloading application code and data shared across all the isolated services that uses app zygote.

- **Hidden API enforcement policy.** Since Android 9, many methods and classes in Android framework have been made inaccessible to the third-party applications through hidden API enforcement policy. It has the following options:
  - *Default.* Based on the type of application. For system applications, it should be disabled, and for others, it should be enforced.
  - *None/disabled.* The application has full access to the hidden API as it used to be before Android 9.
  - *Warn.* Same as above, except that warnings will be logged each time the application accesses the hidden API. This is mostly unused.
  - *Enforce.* The application cannot access hidden API, either dark-grey list or blacklist, or both of them. This is the default option for the third-party applications in Android 9 onwards unless the application is whitelisted by the OEM or the vendor.

  > ***Warning.*** Hidden API enforcement policy is not properly implemented in Android and can be bypassed by the application. As a result, this value should not be trusted.

- **SELinux.** Mandatory access control (MAC) policy set by the operating system via SELinux.
- **Main Activity.** The main entry point to the application. This is only visible if the application has activities and any of those are openable from the Launcher. There's also a launch button on the right-hand side which can be used to launch this activity.

2.2.2.2 Tags

- **Tracker info.** Number of tracker components in the application (e.g., *5 trackers*) The colour of the tag appears orange if the trackers are unblocked and dark cyan if they are blocked in App Manager. Clicking on the tag opens a dialog containing the list of tracker components which can also be blocked or unblocked provided App Manager has sufficient privileges.

- **Application type.** User application or system application. For a system application, it displays whether the application is an updated version of the system application or the application is installed systemless-ly via Magisk.

- **Split APK info.** Number of splits in the APK excluding the base APK (e.g., *5 splits*). Clicking on the tag opens a dialog containing a few additional information, such as name, type, and size.

- **Debuggable.** The application can be debugged over ADB. Debuggable applications can enjoy certain functions unavailable to a regular application. The data of the application might be accessible via ADB (e.g. using `run-as` command) without any additional permissions.

- **Test only.** The application is a test-only application. Test-only applications can enjoy certain functions unavailable to a regular application. The data of the application might be accessible via ADB (e.g. using `run-as` command) without any additional permissions.

- **No code.** The application does not have any code associated with it i.e., DEX files aren't present. In some system applications, the actual code might be located in another place.

- **Large heap.** The application has requested large heap size i.e., more space in memory (RAM) is requested for dynamic allocation. It is still up to the operating system to decide whether to allocate large space for the application. App Manager, for example, requests large heap size because it needs to load an entire APK into memory while scanning an APK.

- **Open links.** The application may open certain links from any applications. If the application can open any links by default, the color of the tag would be red, dark cyan if otherwise. Clicking on the tag opens a dialog with the supported hosts or domains listed. In Android 12 onwards, it displays an option to enable or disable opening links by default provided App Manager has sufficient permissions. Otherwise, it displays an option to open the system settings page for the application.

- **Running.** One or more services of the application is currently running in the background. Clicking on the tag opens a dialog containing the list of running services. Clicking on any services opens the log viewer with default filter set to the UID associated with the service (which can be different from the UID of the application if it is running in a separate or isolated process) provided the log viewer feature is enabled. There's also an option to force-stop the application.
- **Stopped.** The application is force stopped. This may not prevent it from running automatically later.
- **Disabled.** Denotes that the application is disabled (hidden from the launcher).
- **Suspended.** Denotes that the application is suspended (grayed out in the launcher).
- **Hidden.** Denotes that the application is hidden (hidden from the launcher).
- **MagiskHide.** MagiskHide is enabled for the application. Clicking on the tag opens a dialog containing the list of process names within the application that can be added to or removed from the MagiskHide list.
- **MagiskDenyList.** The application is present in Magisk DenyList. Clicking on the tag opens a dialog containing the list of process names within the application that can be added to or removed from Magisk DenyList.
- **WX.** The application violates the "W^X policy" and is capable of writing and executing in the same directory or in the same portion of memory. This allows the execution of arbitrary executables either by the modification of executables embedded within the application or by downloading them from the Internet.
  See also: W^X in Wikipedia
- **Bloatware.** The application may be a known bloatware. Clicking on the tag opens a dialog containing detailed information in addition to suggestions (if available).
- **KeyStore.** The application has items in the Android KeyStore. Clicking on the tag opens a dialog containing all the KeyStore files that belong to the application.
- **Backup.** The application was backed up using App Manager at least once. Clicking on the tag opens a dialog containing all the available backups along with metadata.
- **No battery optimisation.** Battery optimisation is disabled for the application. It is possible to re-enable battery optimisation by clicking on the tag.
- **Sensors disabled.** Sensors are disabled for the application. Sensors are disabled for most applications by default.
- **Net policy.** Network policy (e.g., background data usage) is configured for the application. Clicking on the tag displays a dialog containing the supported policies for the platform along with the options to configure them.
- **SSAID.** Clicking on the tag opens a dialog containing the current SSAID assigned to the application. It is also possible to reset/regenerate the SSAID if needed.
- **SAF.** Denotes that the application has been granted to access one or more storage locations or files i.e. URIs via Storage Access Framework (SAF). Clicking on the tag opens a dialog containing the list of granted URIs.
- **Play App Signing.** Indicates that the application might be signed by Google.
- **Xposed.** Indicates that the application may be an Xposed module. Clicking on the tag displays a dialog with additional information.
- **Static Shared Library.** Denotes that the application serves as a static shared library for one or more applications. Clicking on the tag opens a dialog containing a list of installed versions of the library along with an option to uninstall them.

> ***Notice.*** The trichrome library (supplied by Google Chrome, Vanadium or similar projects) is currently the only known static shared library on Android.

2.2.2.3 Horizontal Action Panel

Horizontal Action Panel, as described in the previous section, consists of various application-related actions, such as–

- **Launch.** Launch the application provided it has a launcher activity.
- **Freeze.** Freeze the application. This button is not displayed if it is already frozen or the user does not have enough privileges. After the application is frozen, it may be hidden from the app drawer depending on how it was configured. Shortcuts configured by the application may also be removed. The application may only be unfrozen via App Manager, `pm` command or any other tools that offer such a feature. Long clicking on the button opens a dialog where a shortcut can be configured to quickly freeze or unfreeze the application.
- **Uninstall.** Uninstall the application with a prompt. In the dialog prompt, it is possible to uninstall updates of a system application, or if App Manager has enough privileges or the operating system supports it, it is possible to uninstall the application without clearing its data and signature. For the latter case, the installed application must match the signature with the previously installed application if it is installed again.

  > ***Tips.*** A better way to reinstall an application with a different signature would be to back up its data using App Manager and restore it again after installing the application instead of opting to preserving data and signature of the application during uninstallation as this option may cause undefined behaviour in the future.

- **Unfreeze.** Unfreeze the application. This button is not displayed if it is already enabled or the user does not have enough privileges. Similar to the *Freeze* button, long clicking on the button opens a dialog where a shortcut can be configured to quickly freeze or unfreeze the application.
- **Force Stop.** Force-stop the application.
- **Clear Data.** Clear data from the application. This includes any information stored in the internal and, recently, the external directories, including accounts (if set by the application), cache, etc. Clearing data from App Manager, for example, removes all the rules (the blocking is not removed though) saved within the application (Which is why you should always take backups of your rules). This button is not displayed if the user does not have enough privileges.
- **Clear Cache.** Clear the application cache. If the application is running during the operation, the cache may not be cleared as expected.
- **Install.** Install the application, only displayed if the application hasn't already been installed.
- **What's New.** Displayed for an external application if an older version of it is already installed. Clicking on this button opens a dialog containing the differences between this and the installed version in a version control manner. Changes include *version*, *trackers*, *permissions*, *components*, *signatures* (only checksum changes), *features*, *shared libraries* and *SDK*.
- **Update.** Displayed if the application has a higher version code than the installed application.
- **Reinstall.** Displayed if the application has the same version code as the installed application.
- **Downgrade.** Displayed if the application has a lower version code than the installed application.
- **Manifest.** Opens the application's manifest file in a separate page. If the application has more than one split, it will display the list of split APK files, and clicking on an item will open the corresponding manifest file instead.

- **Scanner.** Scan the application in order to list potential trackers and libraries. It also scans the file using VirusTotal and fetch results from Pithus if configured.
  See also: Scanner page
- **Shared Prefs.** Displays a list of shared preferences used by the application. Clicking on a preference item in the list opens the Shared Preferences Editor page. This option is only visible if the user has the required privileges.
- **Databases.** Displays a list of databases used by the application. Clicking on an item opens a list of activities that can open the database. This option is only visible if the user has the required privileges.
- **F-Droid.** Open the application in the selected *F-Droid* client.
- **Store.** Open the application in *Aurora Store*. The option is only visible if *Aurora Store* is installed.

2.2.2.4 Options Menu

Options-menu is located in the top-right corner of the page. A complete description of the options present there are given below:

- **Share.** Share button can be used to share the APK or (if the application is has multiple splits, OBB files or any dependencies) *APKS* file extracted from the application.
- **Refresh.** Refresh the App Info tab.
- **View in Settings.** Open the application in Android Settings.
- **Backup/restore.** Open the backup/restore dialog.
- **Export blocking rules.** Export rules configured for the application within App Manager.
- **Open in Termux.** Open the application in Termux. This actually runs `su - user_id` where `user_id` denotes the application's kernel user ID (described in §**??**). This option is only visible to the root users. See §**??** to learn how to configure Termux to run commands from third-party applications.
- **Run in Termux.** Open the application via `run-as package_name` in Termux. This is only applicable to the debuggable applications and works for both root and ADB users. See §**??** to learn how to configure Termux to run commands from third-party applications.
- **MagiskHide.** Open a dialog containing the list of process names within the application that can be added or removed from the MagiskHide list.
- **Magisk DenyList.** Open a dialog containing the list of process namees within the application that can be added or removed from Magisk DenyList.
- **Battery optimisation.** Enable/disable battery optimisation for this application.
- **Sensors.** Enable/disable sensors for this application.
- **Net policy.** Configure network policy (e.g., background data usage) for the application.
- **Extract Icon.** Extract and save the application's icon in the shared storage.
- **Optimize.** Perform optimisation for this application. This option is for advanced users only.
- **Add to profile.** Add the application to one of the configured profiles.
- **Install for. . . .** Install the application for another user and/or in the work profile if configured.

2.2.2.5 Configuring Termux

By default, Termux does not allow running commands from a third-party application. To use this option, Termux v0.96 or later is required and `allow-external-apps=true` must be added in `~/.termux/termux.properties`.

> ***Info.*** Enabling this option does not weaken Termux' security. The third-party applications still need to ask the user to allow running arbitrary commands in Termux.

2.2.3 Component Tabs

Activities, **Services**, **Receivers** (i.e., broadcast receivers) and **Providers** (e.g., content providers) are collectively known as the application components, because they offer similar features and share similar properties. For example, they all have a *name*, a *label*, an *icon*, can be enabled or disabled, and can be executed via *Intent*. Application components are the building blocks of an application and must be declared in the application manifest (with a few exceptions). Application manifest is a file where application specific metadata are stored. The Android operating system learns what to do with the application by reading the metadata.

Colours used in these tabs are explained in §**??**. It is also possible to sort the list of components to display blocked or tracker components on top of the list via the **Sort** option located in the three-dots menu.

2.2.3.1 Activities

Activities are the windows or pages that can be uniquely identified by the Android operating system (e.g., *Main page* and *App Details page* are two activities). Each activity can have multiple UI components known as *widgets* or *fragments*, and each component can be nested or placed on top of each other. The developer can also choose to open external files, links, etc. within an activity using a method called *intent filters*. For example, when you open a file in your file manager, either your file manager or the operating system scans the intent filters via PackageManager, find the activities capable of opening the file, and list those activities so that you can choose your preferred activity.

Activities that are *exportable* can be opened by any third-party applications. However, Some activities may require permissions, and only an application having those permissions can open them. In the *Activities* tab, certain activities can be launched via the **Launch** button. If it is necessary to supply additional information, such as Intent extras, data or action, long clicking on the **Launch** button opens the Activity Interceptor page which provides such features.

> ***Tip.*** No-root users can grant `android.permission.WRITE_SECURE_SETTINGS` via ADB to launch *non-exportable* activities.

> ***Notice.*** If launching an activity throws an error, it may have certain dependencies which are not met (e.g., *App Details* page in App Manager cannot be launched using the launch button, because it requires a package name). Since the dependencies cannot be inferred programmatically, the activity may not be opened from App Manager by default.

It is also possible to create shortcuts of an activity-launch using the **Create shortcut** button. If you need to supply additional information, you can create a shortcut from the Activity Interceptor page instead.

> ***Caution.*** If you uninstall App Manager, all shortcuts created by App Manager will be lost.

2.2.3.2 Services

Unlike activities that users can see, **Services** handle background tasks. For example, if you're downloading a video from the internet using your phone's Internet browser, the Internet browser is using a *foreground service* to download the content.

When an activity is closed or removed from the *Recents* page, it may be destroyed immediately depending on the amount free memory the phone has, battery statistics, or how the activity is configured. But services can be run indefinitely if desired. If more services run in the background, the phone may become slower due to the shortage of memory and/or processing power, and the phone's battery will be drained more quickly. Newer versions of Android come with a battery optimisation feature enabled by default for all applications. With this feature enabled, the system can randomly terminate any service depending on the amount of resources the system has or the service requires. However, foreground services (i.e., services that run with a fixed notification, such as music player or downloader) are not typically terminated unless the system is very low on resources (memory, battery, etc.). Certain stock ROMs can offer more aggressive optimisation. MIUI, for example, has a very aggressive optimisation feature known as the *MIUI optimisation*.

Both activities and services are run in the same looper called the main looper, which means the services do not really run in the background. It is the task of the developer to ensure this. How do the application communicate with the services? It uses broadcast receiver or Binder.

2.2.3.3 Receivers

Receivers (also called *broadcast receivers*) can be used to trigger execution of certain tasks when certain events occur. These components are called broadcast receivers, because they are executed as soon as a broadcast message is received. These broadcast messages are sent using a method called *Intent*. Intent is a special feature in Android that can be used to open applications (i.e., activities), run services and send broadcast messages. Therefore, like activities, broadcast receivers use *intent filters* to receive the desired broadcast messages. Broadcast messages can be sent by the system or the application itself. When a broadcast message is sent, the corresponding receivers are activated by the system so that they can execute tasks. For example, if your phone is low on resources, it may freeze or experience lags for a moment after you enable mobile data or connect it to the Wi-Fi. This is because broadcast receivers that can receive `android.net.conn.CONNECTIVITY_CHANGE` are activated by the system as soon as the data connection is enabled. Since many applications typically use this intent filter, they are all activated almost immediately by the system which causes the freezing or lags.

Receivers can also be used for inter-process communication (IPC), i.e., it can be used to communicate across multiple applications or even different components of a single application.

2.2.3.4 Providers

Providers are primarily used for data management. For example, when you save an APK file or export rules in App Manager, it uses a content provider called `.fm.FmProvider` to save the APK or export the rules. There are many providers, including the ones provided by the system, that can be used to manage various content-related tasks, such as database management, tracking, searching, etc. Each provider has a field called *Authority* which is unique to the application in the entire Android ecosystem just as the package name.

2.2.3.5 Additional Features for Rooted Phones

Unlike the no-root users who are mostly spectators in these tabs, root users can perform various operations.

Blocking Components. On the right-most side of each component item, there is a switch which can be used to toggle the blocking status of that particular component. If Instant Component Blocking is not enabled or blocking is never applied to the application before, it is required to apply the changes using the **Apply rules** option in three-dots menu. It is also possible to remove the already-applied rules using the same option (which would be read as **Remove rules** this time).

It is also possible to block the component using one of the several methods by long clicking on the button.
See also: FAQ: App Components

Blocking Trackers. It is possible to disable tracker components using the **Block tracker** option in the three-dots menu. All tracker components will be blocked regardless of the tab you're currently in.

Info. Tracker components are a subset of application components. Therefore, they are blocked using the same method used for blocking any other components.

See also:

- *FAQ: Tracker classes versus tracker components*
- *Scanner Page*
- *1-Click Ops Page: Block/Unblock Trackers*

2.2.4 Permission Tabs

App Ops, **Uses Permissions** and **Permissions** tabs are related to permissions. In Android communication across applications or processes not having the same identity (known as *shared ID*) often require permissions. These permissions are managed by the permission controller. Some permissions are considered *normal* permissions which are granted automatically if they appear in the application manifest, but *dangerous* and *development* permissions require confirmation from the user. Colours used in these tabs are explained in §**??**.

2.2.4.1 App Ops

App Ops stands for **Application Operations**. Since Android 4.3, *App Ops* are used by Android to control many system permissions. Each app op has a unique number associated with it which is displayed along with the private name of the operation in the App Ops tab. Some app ops also have a public name. A large number of app ops are also associated with *permissions*. In this tab, an app op is considered dangerous if its associated permission is marked as dangerous. Other information such as *flags*, *permission name*, *permission description*, *package name*, *group* are also taken from the associated permission. Others may include the following:

- **Mode.** It describes the current authorisation status which can be *allow*, *deny* (a rather misnomer, it simply means error), *ignore* (it actually means deny), *default* (inferred from a list of defaults set internally by the vendor or the AOSP), *foreground* (in newer Android versions, it means the app op can only be used when the application is running in foreground), and some custom modes set by the vendors (MIUI uses *ask*, for example).
- **Duration.** The amount of time this app op has been used (there can be negative durations whose use cases are currently unknown to me).
- **Accept Time.** Last time the app op was accepted.
- **Reject Time.** Last time the app op was rejected.

Info. Contents of this tab are visible to no-root users if `android.permission.GET_APP_OPS_STATS` is granted via ADB.

There is a toggle button next to each app op item which can be used to allow or deny (ignore) it. Other supported modes can also be set by long clicking on the toggle button. If the desired app op is not listed in the tab, *Set custom app op* option in the menu can be used instead. It is also possible to reset the changes using the *Reset to default* option, or deny all the dangerous app ops using the corresponding option in the menu. Due to the nature how app ops work, the system may take some time to apply them.

Tip. Denying certain app ops may cause the application to misbehave. If all attempts fail, *reset to default* option can be used as the last resort.

It is possible to sort the list in ascending order by app op names and the associated unique numbers (or values), or list the denied app ops first using the corresponding sorting options.
See also: Appendix: App Ops

2.2.4.2 Uses Permissions

Uses Permissions are the permissions used by the application. This is named so because they are specified in the manifest using `uses-permission` tags. Information such as *flags*, *permission name*, *permission description*, *package name*, *group* are taken from the associated permission.

Privileged users can grant or revoke the *dangerous* and *development* permissions via the toggle button on the right side of each permission item. It is also possible revoke dangerous permissions all at once using the corresponding option in the menu. Only these two types of permissions can be revoked because Android does not allow the modification of *normal* permissions (which most of them are). It might still be possible to revoke them by editing `runtime-permissions.xml` itself, but whether this is a possibility is still being investigated.

> ***Info.*** Since dangerous permissions are revoked by default by the system, revoking all dangerous permissions is the same as resetting all the permissions.

It is possible to sort the permissions by their name (in ascending order) or choose to display denied or dangerous permissions at first using the corresponding options in the menu.

2.2.4.3 Permissions

Permissions are usually custom permissions defined by the application itself. These type of permissions are marked as *Internal* permissions. It also contains permissions declared by other applications which are marked as *External* permissions. An external permission can be specified in an *exported* application component so that another application may invoke the component only if it holds the permission. Below is a complete description of each item displayed in this tab:

- **Name.** A permission has a unique name (e.g., `android.permission.INTERNET`) that multiple applications can request. The application that declared the permission is automatically granted and cannot be revoked.
- **Icon.** Each permission can have a custom icon. The other permission tabs do not have any icon because they do not contain any icon in the application manifest.
- **Description.** This optional field describes the permission. If there isn't any description associated with the permission, the field is not displayed.
- **Flags.** (Uses the flag symbol or **Protection Level** name) This describes various permission flags such as *normal*, *development*, *dangerous*, *instant*, *granted*, *revoked*, *signature*, *privileged*, etc.
- **Package Name.** Denotes the package name associated with the permission, i.e. the package that defined the permission.
- **Group.** The group name associated with the permission (if any). Several related permissions can often be grouped together.

2.2.5 Signatures Tab

Signatures are actually called signing information. An application is signed with one or more signing keys by its developer before publishing it. The integrity of an application, i.e., whether the application is from the actual developer and has not been modified by another person, can be checked using the signing certificate included in the APK files. This is because when an application is modified by an unauthorised entity, the application can longer be signed with the original signing keys since the signing keys are unknown to the entity. One way to verify the integrity of an application is via the checksums generated from the certificates. If the developer supplies the checksums for the signing certificates, they can be compared against the checksums generated in the **Signatures** tab to verify the application. For example, if you have downloaded App Manager from GitHub or Telegram Channel, you can verify whether the application was actually released by me by simply matching the following *SHA256* checksum with the one displayed in this tab:

```
320c0c0fe8cef873f2b554cb88c837f1512589dcced50c5b25c43c04596760ab
```

Several hashing algorithms are used to generate checksums in this tab. They include *MD5*, *SHA1*, *SHA256* and *SHA512*.

> ***Caution.*** Signing information should be verified using a reliable hashing algorithm, such as *SHA256*. DO NOT rely on *MD5* or *SHA1* checksums as they are known to generate the same checksums for multiple certificates.

2.2.6 Uses Features tab

Uses Features tab lists the features declared by the application, such as OpenGL ES, telephony, and leanback. Some features can be required by the application, and some features can be optional. Required features must be present in the system along with the required version. Otherwise, any attempt to install the application will be denied by the system. Colours used in this tab are explained in §**??**.

2.2.7 Configurations tab

Configurations tab lists the configurations required by the application, such as input method type (qwerty, 12 key), touch screen type (finger, stylus, etc.), and navigation type (dial pad, trackball, wheel). This tab is going to be empty for most applications.

2.2.8 Shared Libraries Tab

Shared libraries tab lists the legacy JAR dependencies, any static shared library dependencies (currently, the only known cases are the Chromium-based browsers and WebViews) as well as the JNI (Java native interface) libraries. For JNI libraries, it specifies platform (x86/x86_64/ARM/AArch64), architecture (32/64 bit), object type (shared object or executable), etc.

2.3 1-Click Ops Page

This page is displayed on selecting the 1-Click Ops option in the main menu.

2.3.1 Block/Unblock Trackers

This option can be used to block or unblock the ad/tracker components from the installed applications. On selecting this option, App Manager will ask if it should list trackers from all the applications or only from the user applications. Novice users should avoid blocking trackers from the system applications in order to avoid bad consequences. After that, a multi-choice dialog box will appear where it is possible to exclude one or more applications from this operation. The changes are applied immediately on pressing the *block* or *unblock* button.

> ***Notice.*** Certain applications may not function as expected after blocking their trackers. If that is the case, remove the blocking rules all at once or one by one in the component tabs of the App Details page for the corresponding application.

See also: App Details Page: Blocking Trackers

2.3.2 Block Components…

This option can be used to block certain application components as specified by their signatures. A signature of a component is the full name or partial name of the component. For safety, it is recommended to add a `.` (dot) at the end of each partial signature, because the underlying algorithm searches and matches the components in a greedy manner. It is also possible to insert more than one signature in which case all the signatures have to be separated by white spaces. Similar to the option above, there is also an option to apply blocking to the system applications.

> ***Caution.*** If you are not aware of the consequences of blocking applcations components by their signatures, you should avoid using this option as it may result in bootloop or soft brick, and you may have to apply factory reset as a result.

2.3.3 Set Mode for App Ops…

This option can be used to configure certain applcation operations of all or selected applications. There are two fields. The first field can be used to insert more than one app op constants (either names or values) separated by white spaces. It is not always possible to know in advance about all the app op constants as they vary from device to device and from OS to OS. Desired app op constant can be found in the *App Ops* tab located in the App Details page. The second field can be used to insert or select one of the modes that will be set against the specified app ops.

> ***Caution.*** Unless you are well-informed about app ops and the consequences of blocking them, you should avoid using this option.

2.3.4 Back up

1-Click options for back up. As a precaution, it lists the affected backups before performing any operation.

Back up all apps. Back up all the installed applications.

Redo existing backups. Back up all the installed applications that have a previous backup.

Back up apps without backups. Back up all the installed applications without a previous backup.

Verify and redo backups. Verify the recently made backups of the installed applications and redo backup if necessary.

Back up apps with changes. If an app has changed since the last backup, redo its backup. It checks a number of indices including application version, last update date, last launch date, integrity and file hashes. Directory hashes are taken during the backup process and are stored in a database. On running this operation, new hashes are taken and compared with the ones kept in the database.

2.3.5 Restore

1-Click options for restore. As a precaution, it lists the affected backups before performing any operation.

Restore all apps. Restore *base backup* of all the backed up applications.

Restore not installed apps. Restore *base backup* of all the backed up applications that are not currently installed.

Restore latest backups. Restore *base backup* of already installed applications whose version codes are higher than the installed version code.

2.3.6 Trim Caches in All Apps

Delete caches from all applications, including Android system. During this operation, caches of all the running applications may not be cleared as expected.

2.4 Profiles Page

Profiles page can be accessed from the options-menu in the main page. It primarily displays a list of configured profiles along with the typical options to perform operations on them. New profiles can also be added using the *plus* button at the bottom-right corner. Profiles can be imported, duplicated or deleted. Clicking on a profile item opens its profile page.

2.4.1 Options Menu

The three-dots menu in the top-right corner opens the global option menu. It has an option to import an existing profile that was previously exported from App Manager.

Long clicking on any profile item brings up another options-menu. It offers the following options:

- **Apply now. . . .** This option can be used to apply the profile directly. When clicked, a dialog is displayed where it is possible to select a profile state. On selecting one of the states, the profile will be applied immediately.
- **Delete.** Clicking on this option will remove the profile immediately without a warning.
- **Duplicate.** This option can be used to duplicate the profile. When clicked, a dialog is displayed where it is possible to set a name for the new profile. On clicking "OK", the profile page will be loaded by duplicating all the configurations that this profile have. However, the profile will not be saved until it is saved manually.
- **Copy profile ID.** This option is used to copy the unique profile ID of the profile. The profile ID can be used to trigger the profile from a third-party application.
- **Export.** Export the profile to an external storage. Profiles exported this way can be imported via the *import* option as mentioned above.
- **Create shortcut.** This option can be used to create a shortcut for the profile. There are two options: *Simple* and *Advanced*. When configured with the latter option, it prompts the user to select a profile state when the shortcut is invoked. The former option, on the other hand, always uses the default state that was configured when the profile was last saved.

2.5 Profile Page

Profile page displays the configurations for a profile. It also offers the options to edit them.

2.5.1 Options Menu

The three dots menu in the top-right corner opens the options-menu. It contains several options such as–

- **Apply.** This option can be used to apply the profile directly. When clicked, a dialog is displayed where it is possible to select a profile state. On selecting one of the states, the profile will be applied immediately.

  > ***Notice.*** When you apply a profile, if some packages do not match the criteria, they will simply be ignored.

- **Save.** Save changes to the profile.
- **Discard.** Discard any modifications made since the last time it was saved.
- **Delete.** Clicking on this option will remove the profile immediately without a warning.

- **Duplicate.** This option can be used to duplicate the profile. When clicked, a dialog is displayed where it is possible to set a name for the new profile. On clicking "OK", this page will be reloaded by duplicating all the configurations that this profile have. However, the new profile will not be saved until it is saved manually.

- **Create shortcut.** This option can be used to create a shortcut for the profile. There are two options: *Simple* and *Advanced*. When configured with the latter option, it prompts the user to select a profile state when the shortcut is invoked. The former option, on the other hand, always uses the default state that was configured when the profile was last saved.

2.5.2 Apps Tab

Apps tab lists the packages configured for this profile. Packages can be added or removed using the *plus* button located near the bottom of the screen. A package can also be removed by long clicking on it (in which case, a popup will be displayed with the only option, *delete*).

2.5.3 Configurations Tab

Configurations tab can be used to configure the selected packages.

> ***Uninstalling a List of Applications..*** Since uninstalling is a one time event, it is not part of the configurations. However, it is still possible to uninstall the applications belonging to a profile. To do this, you can filter the applications in the Main page using a profile, select all the filtered applications, and uninstall them.

2.5.3.1 Profile ID

The unique ID for this profile, currently set based on the profile name. The profile ID can be used to trigger the profile from a third-party application.

2.5.3.2 Comment

This is the text that will be displayed in the profiles page. If not set, the current configurations will be displayed instead.

2.5.3.3 State

Denotes how certain configured options will behave by default. For instance, if *disable* option is turned on, the applications will be disabled if the state is *on* and will be enabled if the state is *off*. Currently, it only supports *on* and *off* values.

2.5.3.4 Users

Select users for which is the profile will be applied. All users are selected by default.

2.5.3.5 Components

This behaves the same way as the Block Components... option does in the 1-Click Ops page. However, the blocking here is only applied to the selected packages. If the state is *on*, the components will be blocked, and if the state is *off*, the components will be unblocked. The option can be disabled (regardless of the inserted values) by clicking on the *disabled* button on the input dialog.
See also: What are the app components?

2.5.3.6 App Ops

This behaves the same way as the Set Mode for App Ops… option does in the 1-Click Ops page. However, the operation here is only applied to the selected packages. If the state is *on*, the app ops will be denied (i.e. ignored), and if the state is *off*, the app ops will be allowed. The option can be disabled (regardless of the inserted values) by clicking on the *disable* button in the input dialog.

2.5.3.7 Permissions

This option can be used to grant or revoke certain permissions from the selected packages. Like others above, permissions must be separated by white spaces. If the state is *on*, the permissions will be revoked, and if the state is *off*, the permissions will be allowed. The option can be disabled (regardless of the inserted values) by clicking on the *disable* button in the input dialog.

2.5.3.8 Backup/Restore

This option can be used to take a backup of the selected applications and its data or restore them. Two options are available here: *Backup options* and *backup name*.

- **Backup options.** Same as the backup options of the backup/restore feature. If not set, the default options will be used.
- **Backup name.** Set a custom name for the backup. If the backup name is set, each time a backup is made, it will be given a unique name with backup-name as the suffix. This behaviour will be fixed in a future release. Leave this field empty for regular "base" backups (also, make sure not to enable *backup multiple* in the backup options).

If the state is *on*, the packages will be backed up, and if the state is *off*, the packages will be restored. The option can be disabled by clicking on the *disable* button in the input dialog.

2.5.3.9 Export Blocking Rules

Danger. This option is not yet implemented.

2.5.3.10 Freeze

Allow freezing or unfreezing the selected packages depending on the value of the state. If the state is *on*, the packages will be frozen, and if the state is *off*, they will be unfrozen.

2.5.3.11 Force-stop

Allow the selected packages to be force-stopped.

2.5.3.12 Clear Cache

Enable clearing cache for the selected packages.

2.5.3.13 Clear Data

Enable clearing data for the selected packages.

2.5.3.14 Block Trackers

Enable blocking or unblocking of the tracker components from the selected packages depending on the value of the state. If the state is *on*, the trackers will be blocked, and if the state is *off*, the trackers will be unblocked.

2.5.3.15 Save APK

Enable saving APK files at `AppManager/apks` (or in the directory selected in the settings page) of the selected packages.

2.6 Settings Page

Settings page can be used to customise the behaviour of App Manager.

2.6.1 Language

Configure in-app language. App Manager currently supports 22 (twenty-two) languages.

2.6.2 Appearance

2.6.2.1 App Theme

Configure in-app theme.

2.6.2.2 Pure Black Theme

Whether to use a black background instead of the material themed background.

2.6.2.3 Layout Direction

Change layout direction, either left to right or right to left. This is usually set using the selected language but not everybody prefers the same direction.

2.6.2.4 Enable/Disable Features

Enable or disable certain features in App Manager, such as

- **Interceptor**
- **Manifest viewer**
- **Scanner**
- **Package installer**
- **Usage access.** With this feature turned off, App Manager will never ask for the *Usage Access* permission.
- **Log viewer**
- **App explorer.** The "Explore" option will not be available while trying to open an APK file.
- **App info.** The "App info" option displayed while trying to open an APK file.
- **Code Editor**
- **VirusTotal**

2.6.3 Privacy

2.6.3.1 Screen Lock

Lock App Manager using Android screen lock provided a screen lock is configured.

> ***Warning.*** If screen lock is disabled in Android after enabling this setting, App Manager will not open until it is enabled again.

2.6.3.2 Run App Manager in the Background

Whether to run App Manager in the background to reduce the initialisation delay. On certain devices, this can also help if you're frequently disconnected from ADB.

2.6.3.3 Use the Internet

Whether to activate the Internet features in App Manager. This currently include VirusTotal and Pithus scanning in the Scanner page.

2.6.3.4 Authorization Manager

This setting allows a third-party application to get access to certain features, such as profiles.

2.6.4 Mode of Operation

Mode of operation defines how App Manager works as a whole. It has the following options:

- **Auto.** Let App Manager decide the suitable option. Although this is the default option, non-rooted users should use the *no-root* mode.
- **Root.** Operate App Manager in root mode. App Manager will fall back to *no-root* mode if root is not detected, or in rare cases when Binder communication through root is disabled (e.g. in Phh SuperUser).
- **ADB over TCP.** Operate App Manager in ADB mode via ADB over TCP. App Manager will fall back to *no-root* mode if ADB over TCP is not enabled.
- **Wireless debugging.** Enable ADB via Wireless Debugging. It will try to connect to the configured port automatically at first. On failure, it will ask the user to either pair or connect to the ADB daemon manually. App Manager will fall back to *no-root* mode if it fails to connect to the ADB daemon this way.

  > ***Info.*** This option is only displayed in devices running Android 11 or later as Wireless Debugging was introduced in Android 11.

- **No-root.** Operate App Manager in no-root mode. While App Manager performs better in this mode, all the root- or ADB-specific features will be disabled.

It also displays the actual mode of operation at the top. The actual mode of operations are *root*, *ABD* and *no-root*.

> ***Notice.*** "Remote service" is only required for ADB users or when you use the custom commands.

2.6.5 APK Signing

2.6.5.1 Signature Schemes

Configure the signature schemes to be used when APK signing is enabled. v1 and v2 signature schemes are enabled by default, but v3 should also be enabled to ensure proper security in Android 9 or later.

2.6.5.2 Signing Key

Configure the signing key for signing APK files. Keys from an existing KeyStore can be imported to App Manager, or a new key can be generated.

> ***Tip.*** If you need to use the key in the future, it is recommended that you create a KeyStore yourself and import the key here. Without a proper backup, keys generated within App Manager are at the risk of being deleted.

2.6.5.3 Align APK Files

Performs zip alignment when App Manager signs an APK file. Zip alignment stores the files in the APK file (which is actually a zip file) in a way that a zip file reader can access the files quite easily using random access instead of loading the entire APK file in the memory, which results in the reduction of Android's memory usage. Note that this step is required if an application's manifest has `extractNativeLibs` set to `true`.

2.6.6 Installer

Configure the default behaviour of the installer. You can also find most of the settings by clicking on the *cog* icon when you install an application.

2.6.6.1 Install Location

Define APK installation location. This can be one of *auto*, *internal only* and *prefer external*. In newer Android versions, selecting the last option does not guarantee that the application will be installed in the external storage.

2.6.6.2 Block Trackers

Whether to block the tracking components immediately after installing the application.

2.6.6.3 Display Changes

Whether to display changes in version, trackers, components, permissions, signatures, SDK, etc. in a version controlled style before installing the application if the application has already been installed.

2.6.6.4 Installer App

Select the installer application. This is useful for applications that explicitly checks the installer as a way to verify if the application is installed legitimately. This only works for root or ADB users.

> ***Notice.*** While checking for the installer might seem a legitimate concern for an application, the Android framework already deals with this during the installation. Checking for the installer is simply the wrong way to prove the legitimacy of the source of an application.

2.6.6.5 Sign APK

Whether to sign the APK files before installing the application. A signing key has to be added or generated before this option can be enabled. This can be done in the APK signing page.

2.6.6.6 Immediately Perform DEX Optimization

Whether to perform DEX optimization immediately after installing the application. This can be useful for heavy applications, such as the gaming applications.

2.6.6.7 Install in the Background

Whether to always install applications in the background. A notification will be issued once the installation is finished.

2.6.7 Back up/Restore

Settings related to back up/restore.

2.6.7.1 Compression method

Set the compression method to be used during backups. App Manager supports GZip, BZip2 and Zstandard compression methods, GZip being the default compression method. It doesn't affect the restore of an existing backup.

2.6.7.2 Backup Options

Customise the *back up/restore dialog* displayed while taking a backup.
See also: Backup options

2.6.7.3 Backup apps with Android KeyStore

Allow backup of applications that has entries in the Android KeyStore. This option is disabled by default because a few apps (such as Signal or Element) may crash if restored.

2.6.7.4 Encryption

Set an encryption method for the backups. App Manager currently supports OpenPGP (via OpenKeyChain), AES, RSA and ECC. Like APK signing, The AES, RSA and ECC keys are stored in the KeyStore and can be imported from other KeyStores.

> ***Danger.*** For your own safety, it is not recommended generating RSA and ECC keys inside App Manager. Instead, they should be imported from a KeyStore stored in a secure place.
> In case of AES, the generated key should be stored in a secure place, such as in a password manager.

2.6.7.5 Backup Volume

Select the storage where the backups will be stored. This is also where logs and exported APK files are saved.

> ***Notice.*** The backup volume only specifies the storage, not the path. Backups are traditionally stored in the `AppManager` folder inside the storage path. But when the path is selected using Storage Access Framework (SAF), the selected path or directory is used directly.

2.6.7.6 Import Backups

Import backups from old and discontinued projects such as Titanium Backup, OAndBackup, and Swift Backup (version 3.0 to 3.2). The backups are not deleted after importing to prevent data loss in case the imported backups cannot be restored properly.

2.6.8 Rules

2.6.8.1 Instant Component Blocking

By default, blocking rules are not applied unless they are applied explicitly in the App Details page for any application. After enabling this option, all (old and new) rules are applied immediately for all applications without explicitly enabling blocking for an application.
See also: FAQ: What is instant component blocking?

2.6.8.2 Import/Export Blocking Rules

It is possible to import or export blocking rules within App Manager for all applications. The types of rules (components, app ops or permissions) that should be imported or exported can also be selected. It is also possible to import blocking rules from Blocker and Watt. If it is necessary to export blocking rules for a single application, the corresponding App Details page can be used to export rules, or for multiple apps, batch operations can be used.
See also: Rules Specification

Export Export blocking rules for all applications configured within App Manager. This may include app components, app ops and permissions based on the options selected in the multi-choice options.

Import Import previously exported blocking rules from App Manager. Similar to export, this may include app components, app ops and permissions based on the options selected in the multi-choice options.

Import Existing Rules Add components disabled by other applications to App Manager. App Manager only keeps track of the components disabled within App Manager. If application components are blocked or disabled by other tools or applications, this option can be utilised to import them. On clicking this option, App Manager will find the components potentially disabled by other applications or tools and list only the name of the applications along with the number of matched components. For safety, all the applications are unselected by default. They have to be selected manually, and the blocking has to be re-applied via App Manager.

> ***Caution.*** Be careful when using this tool as there can be many false positives. Choose only the applications that you are certain about.

Import from Watt Import configuration files from Watt, each file containing rules for a single package and file name being the name of the package with `.xml` extension.

> ***Tip.*** Location of configuration files in Watt: `/sdcard/Android/data/com.tuyafeng.watt/files/ifw`

Import from Blocker Import blocking rules from Blocker, each file containing rules for a single package. These files have a `.json` extension.

2.6.8.3 Remove all rules

One-click option to remove all rules configured within App Manager. This will enable all blocked components, app ops will be set to their default values and permissions will be granted.

2.6.9 Advanced

2.6.9.1 Selected Users

This option lets you control the users App Manager should operate on. App Manager operates on all users in root or ADB mode by default.

2.6.9.2 Saved APK Name Format

Defines the format of the APK name to be used while saving it via batch operations or through profiles. App Manager offers some special keywords enclosed inside `%` (percentage) signs and available below the input box. These keywords are:

- `label`. Denotes the name or label of the application. This can be localised to the configured language depending on the app.

- `package_name`. Denotes the name of the package or application ID, the unique identifier that each application has.
- `version`. Denotes the current version of the application extracted from its manifest.
- `version_code`. Denotes the current version code of the application that can be used to separate two versions of the same application.
- `min_sdk`. Denotes the minimum SDK (i.e. Android framework version) that the application can operate on. This data is only available since Android 7 (Nougat).
- `target_sdk`. Denotes the SDK that this application targets. The application can operate on higher SDK but only in the compatibility mode.
- `datetime`. Denotes the time and date when the APK is exported.

2.6.9.3 Import/Export Keystore

Import or export the KeyStore used by App Manager. This is a Bouncy Castle KeyStore with `bks` extension. Therefore, other KeyStore such as Java KeyStore (JKS) or PKCS #12 are not supported. If a key is needed to be imported from such a KeyStore, the relevant options should be should as specified above.

2.6.10 About the device

Display Android version, security, CPU, GPU, battery, memory, screen, languages, user info, etc.

2.7 Scanner Page

Scanner page appears after clicking on the *scanner* button in the App Info tab. External APK files can also be opened for scanning from file managers, web browsers, etc.

It scans for trackers and libraries, and displays the number of trackers and libraries as a summary. It also displays checksums of the APK file as well as the signing certificates. If VirusTotal is configured in the settings, it also attempts to retrieve reports from VirusTotal, or uploads the APK file if it is not in the database. It also display a link to the Pithus report provided the Internet features are enabled.

> ***Disclaimer.*** App Manager only scans an application statically without prejudice. The application may provide the options for opting out, or in some cases, certain features of the tracker may not be used at all by the application (e.g. F-Droid), or some applications may simply use them as placeholders to prevent the breaking of certain features (e.g. Fennec F-Droid). **The intention of the scanner is to give you an idea about what the APK might contain. It should be taken as an initial step for further investigations.**

Clicking on the first item (i.e. number of classes) opens a new page containing a list of tracker classes for the application. All classes can also be viewed by clicking on the *Toggle Class Listing* menu. The SMALI or Java version of the class can be viewed by simply clicking on an item.

> ***Notice.*** Due to various limitations, it is not possible to scan all the components of an APK file. This is especially true if an APK is highly obfuscated or packed. The scanner also does not check strings (or website signatures).

The second item lists the number of trackers along with their names. Clicking on the item displays a dialog containing the name of trackers, matched signatures, and the number of classes against each signature. Some tracker names may have ^{2} prefix which indicates that the trackers are in the ETIP stand-by list, i.e., whether they are actual trackers is still being investigated.

The third item lists the number of libraries along with their names. The information are mostly taken from IzzyOnDroid repo.

See also: FAQ: Tracker classes vs tracker components

2.8 Interceptor Page

Interceptor can be used to intercept communication between applications using `Intent`. It works as a man-in-the-middle between the source and the destination applications. It offers a feature-complete user interface for editing `Intents`.

> ***Warning.*** Interceptor only works for *implicit* intents where the app component isn't specified.

See also:

- *Common Intents*
- *Intents and Intent Filters*

2.8.1 Intent Filters

Intent filters are used by the applications to specify the tasks they are able to perform or the tasks they are going to perform using other applications. For example, when you're opening a PDF file using a file manager, the file manager will try to find the applications to open the PDF with. To find the right applications, the file manager will create an Intent with filters such as the MIME type and ask the system to retrieve the applications capable of opening this filter. The system will search through the Manifest of the installed applications to match the filter and list the application components that are able to open this filter (in our case the PDF). At this, either the file manager will open the desired application component all by itself or use a system provided option to open it. If multiple application components are able to open it and no default is set, you may get a prompt where you have to choose the right application component.

2.8.1.1 Action

Action specifies the generic action to perform such as `android.intent.action.VIEW`. Applications often declare the relevant actions in the Manifest file to catch the desired Intents. The action is particularly useful for broadcast Intent where it plays a vital rule. In other cases, it works as an initial way to filter out the relevant application components. Generic actions such as `android.intent.action.VIEW` and `android.intent.action.SEND` are widely used by applications. Hence, setting this alone may match many application components.

2.8.1.2 Data

Data is originally known as URI (Uniform Resource Identifier) defined in RFC 2396. It can be web links, file location, or a special feature called *content*. Contents are an Android feature managed by the content providers. Data are often associated with a MIME type.

Examples:

```
http://search.disroot.org/?q=URI%20in%20Android%20scheme&categories=general&language=en-US
https://developer.android.com/reference/android/net/Uri
file:///sdcard/AppManager.apk
mailto:email@example.com
content://io.github.muntashirakon.AppManager.provider/23485af89b08d87e898a90c7e/AppManager.apk
```

2.8.1.3 MIME Type

MIME type of the data. For example, if the data field is set to `file:///sdcard/AppManager.apk`, the associated MIME type can be `application/vnd.android.package-archive`.

2.8.1.4 Categories

This is similar to action in the sense that it is also used by the system to filter application components. This has no further benefits. Unlike *action*, there can be more than one category. Clicking on the *plus* button next to the title allows adding more categories.

2.8.1.5 Flags

Flags are useful in determining how system should behave during the launch or after the launch of an activity. This should not be touched as it requires some technical background. The *plus* button next to the title can be used to add one or more flags.

2.8.1.6 Extras

Extras are the key-value pairs used for supplying additional information to the destination component. More extras can be added using the *plus* button next to the title.

2.8.1.7 URI

Represents the entire Intent as a URI (e.g. `intent://...`). Some data cannot be converted to string, and as a result, they might not appear here.

2.8.2 Matching Activities

List all the activity components that matches the Intent. This is internally determined by the system (rather than App Manager). The launch button next to each component can be used to launch them directly from App Manager.

2.8.3 Reset to Default

Reset the Intent to its initial state.

2.8.4 Send Edited Intent

Resend the edited Intent to the destination application. This may open a list of applications where the desired application is needed to be selected. The result received from the target application will be sent to the source application. As a result, the source application will not know if there was a man-in-the-middle.

2.9 Shared Preferences Editor Page

Shared preferences can be edited in this page. Clicking any item on the list opens the edit dialog where the item can be edited. The floating action button in the bottom-right corner can be used to add a new item. To save or delete the file, or to discard current changes, the respective options in the menu can be used.

Chapter 3

Guides

3.1 ADB over TCP

Many root-only features can still be used by enabling ADB over TCP. To do that, a PC or Mac is required with Android platform-tools installed, and an Android phone with developer options & USB debugging enabled.

> ***Root users.*** If superuser permission has been granted to App Manager, it can already execute privileged code without any problem. **Therefore, root users do not need to enable ADB over TCP.** But if you insist on using ADB over TCP, you must revoke superuser permission for App Manager.

See also: FAQ: ADB over TCP

3.1.1 Enable developer options

3.1.1.1 Location of developer options

Developer options is located in Android **Settings**, either directly near the bottom of the page (in most ROMs) or under some other settings, such as **System** (Google Pixel, Lineage OS, Asus Zenfone 8.0+), **Additional Settings** (Xiaomi MIUI, Oppo ColorOS), **More Settings** (Vivo FuntouchOS), **More** (ZTE Nubia). Unlike other options, it is not visible until explicitly enabled by the user. If it is already enabled, you can use the search box in Android **Settings** to locate it as well.

3.1.1.2 How to enable developer options

This option is available within Android **Settings** as well but like the location of the developer options, it also differs from device to device. But in general, you have to find **Build number** (or **MIUI version** for MIUI ROMs and **Software version** for Vivo FuntouchOS, **Version** for Oppo ColorOS) and tap it at least 7 (seven) times until you finally get a message saying *You are now a developer* (you may be prompted to insert pin/password/pattern or solve captchas at this point). In most devices, it is located at the bottom of the settings page, inside **About Phone**. But the best way to find it is to use the search box.

3.1.2 Enable USB debugging

After locating the developer options, enable **Developer option** (if not already). After that, scroll down a bit until you will find the option **USB debugging**. Use the toggle button on the right-hand side to enable it. At this point, you may get an alert prompt where you may have to click *OK* to actually enable it. You may also have to enable some other options depending on device vendor and ROM. Here are some examples:

3.1.2.1 Xiaomi (MIUI)

Enable **USB debugging (Security settings)** as well.

3.1.2.2 Huawei (EMUI)

Enable **Allow ADB debugging in charge only mode** as well. When connecting to your PC or Mac, you may get a prompt saying **Allow access to device data?** in which case click **YES, ALLOW ACCESS**.

> ***Notice.*** Often the **USB debugging** mode could be disabled automatically by the system. If that's the case, repeat the above procedure.

3.1.2.3 Realme

Depending on the device and the version of operating system, you have to enable **Disable Permission Monitoring**, or **USB debugging (Security settings)** along with **Install via USB**.

3.1.2.4 OnePlus (Oxygen OS)

Depending on the device and the version of operating system, you have to enable **Disable Permission Monitoring**.

3.1.2.5 LG

Make sure you have **USB tethering** enabled.

3.1.2.6 Troubleshooting

In case **USB Debugging** is greyed out, you can do the following:

1. Make sure you enabled USB debugging before connecting your phone to the PC or Mac via USB cable
2. Enable USB tethering after connecting to PC or Mac via USB cable
3. (For Samsung) If your device is running KNOX, you may have to follow some additional steps. See official documentations or consult support for further assistant

3.1.3 Setup ADB on PC or Mac

In order to enable ADB over TCP, you have to set up ADB in your PC or Mac. ***Lineage OS users can skip to §??.***

3.1.3.1 Windows

1. Download the latest version of Android SDK Platform-Tools for Windows
2. Extract the contents of the zip file into any directory (such as `C:\adb`) and navigate to that directory using *Explorer*
3. Open **Command Prompt**, **PowerShell**, or **Terminal** from this directory. You can do it manually from the start menu or by holding `Shift` and right clicking within the directory in *File Explorer* and then clicking either on *Open command window here*, or *Open PowerShell window here* (depending on what you have installed). You can now access ADB by typing `adb` (Command Prompt) or `./adb` (PowerShell). Do not close this window yet.

> ***Tip.*** If you have WinGet installed, you can install ADB using the following command:
> `<MINTED>`
> After that, you can simply type `adb` to access ADB.

3.1.3.2 macOS

1. Download the latest version of Android SDK Platform-Tools for macOS

2. Extract the contents of the zip file into a directory by clicking on it. After that, navigate to that directory using *Finder* and locate `adb`

3. Open **Terminal** using *Launchpad* or *Spotlight* and drag-and-drop `adb` from the *Finder* window into the *Terminal* window. Do not close the *Terminal* window yet

> ***Tip.*** If you have Homebrew installed, you can install ADB using the following command:
> <MINTED>
> After that, you can simply type `adb` in any *Terminal* window to access ADB.

3.1.3.3 Linux

1. In your favourite terminal emulator, run the following command:

   <MINTED>

2. If it is successful, you can simply type `./adb` in the in *same* terminal emulator window or type `~/Downloads/platform-tools/adb` in any terminal emulator to access ADB.

3.1.4 Configure ADB over TCP

3.1.4.1 Lineage OS 17.1 and Earlier

Lineage OS (or its derivatives) users can directly enable ADB over TCP using the developer options. To enable that, go to the **Developer options**, scroll down until you find **ADB over Network**. Now, use the toggle button on the right-hand side to enable it and skip to §**??**.

> ***Warning.*** You can turn off **ADB over Network** in developer options, but turning off this option will also stop App Manager's remote server. So, turn it off only when you're not going to use App Manager in ADB over TCP mode.

3.1.4.2 Enable ADB over TCP via PC or Mac

For other ROMs, you can do this using the command prompt/PowerShell/terminal emulator that you've opened in the step 3 of the previous section. In this section, I will use `adb` to denote `./adb`, `adb` or any other command that you needed to use based on your platform and software in the previous section.

1. Connect your device to your PC or Mac using a USB cable. For some devices, it is necessary to turn on *File transfer mode (MTP)* as well

2. To confirm that everything is working as expected, type `adb devices` in your terminal. If your device is connected successfully, you will see something like this:

   ```
   List of devices attached
   xxxxxxxx  device
   ```

   > ***Notice.*** In some Android phones, an alert prompt will be appeared with a message **Allow USB Debugging** in which case, check *Always allow from this computer* and click **Allow**.

3. Finally, run the following command to enable ADB over TCP:

   <MINTED>

Danger. You cannot disable developer options or USB debugging after enabling ADB over TCP.

3.1.4.3 Enable ADB mode in App Manager

After enabling ADB over TCP, relaunch App Manager. App Manager should detect ADB mode automatically. If it cannot, you can change the mode of operation to ADB over TCP in the settings page. There, you can also verify whether App Manager has correctly detected ADB as indicated by the *inferred mode*.

Notice. In some Android devices, the USB cable is needed to be disconnected from the PC before connecting to App Manager.

Warning. ADB over TCP will be disabled after a reboot. In that case, you have to follow §**??** again.

3.2 Wireless Debugging

If you are running Android 11 or later and capable of connecting to a Wi-Fi network for, at least, a few moments, Wireless Debugging is the recommended approach as it offers more protection than ADB over TCP. It requires two steps:

1. **ADB pairing.** The initial and a bit complex step for a novice user. Fortunately, this step is not required all the time.
2. **Connecting to ADB.** Needs to be done every time you reboot your phone. App Manager can also automate this step in most devices.

3.2.1 Enable developer options and USB Debugging

See §**??** and §**??**.

3.2.2 Enable Wireless Debugging

In the **Developer options** page, find **Wireless debugging** and click to open it. In the new page, turn on *Use wireless debugging*. Depending on the operating system, you might see a dialog prompt asking you to verify your decision. If that is the case, click *Allow*.

Tip. For easy access, you might want to add **Wireless debugging** in the notification tiles section. To do this, find **Quick settings developer tiles** in the **Developer options** page and click to open it. In the new window, enable *Wireless debugging*. In case you do not see this setting, you may find a **Wireless debugging** tile in the tile customization panel.

3.2.3 Pair ADB with App Manager

In App Manager, navigate to **Settings** > Mode of operation and then enable *Wireless debugging*. At this, App Manager will try to establish a wireless debugging connection automatically which will fail if it has not been paired before. Once it fails, it will ask you to either connect or pair ADB. Select *pair* and a new dialog will appear. It will ask you to navigate to the **Wireless debugging** page.

Note. As of v4.0.0, pairing is done using a notification prompt. So, if you have disabled notification for App Manager, you must enable it first.

In the **Wireless debugging** page, select **Pair device with pairing code**. At this, a dialog containing a pairing code will be displayed. A notification asking for the pairing code will also be visible almost instantly. Insert the pairing code in the input box in the notification and click *pair*. If the pairing is successful, App Manager will display notification with the message “paired”, and the dialog in the **Wireless debugging** page will be dismissed automatically. You will also be able to see App Manager listed as an ADB client in the same page.

> ***Notice.*** If you do not use App Manager in ADB mode for a while, App Manager might be removed from the list of clients. In that case, you have to repeat the above procedure.

3.2.4 Connect App Manager to ADB

App Manager should be able to connect to ADB automatically if the mode of operation is set to *auto*, *ADB over TCP* or *Wireless debugging*. If this is not the case, select *Wireless debugging* in **Settings** > Mode of operation. If App Manager fails to detect or connect to ADB, it will ask you to connect or pair ADB. Select *connect*.

Now, navigate to the **Wireless debugging** page in Android settings, and note down the port number displayed in the page. In App Manager's dialog prompt, replace the port number with the one you have noted earlier, and click *connect*.

Once a connection has been established, you can disable **Wireless debugging** in Android settings.

> ***Caution.*** Never disable **USB Debugging** or any other additional options described in §**??**. If you do this, the remote server used by App Manager will be stopped, and you may have to start all over again.

3.3 Back up/Restore

App Manager has a modern, advanced and easy-to-use backup/restore system implemented from the scratch. This is probably the only app that has the ability to restore not only the app or its data but also permissions and rules that you've configured within App Manager. You can also choose to back up an app multiple times (with custom names) or for all users.

See also:

- *1-Click Ops: Back up*
- *1-Click Ops: Restore*

3.3.1 Location

Back up/restore is a part of batch operations. It is also located inside the options menu in the App Info tab. Clicking on **Backup/Restore** opens the **Backup Options**. Backups are located at `/storage/emulated/0/AppManager` by default. You can configure custom backup location in the settings page in which case the backups will be located at the `AppManager` folder in the selected volume.

> ***Note.*** If one or more selected apps do not have any backup, the **Restore** and **Delete Backup** options will not be displayed.

3.3.2 Backup Options

Backup options (internally known as backup flags) let you customise the backups on the fly. However, the customisations will not be remembered for the future backups. If you want to customise this dialog, use Backup Options in the Settings page.

A complete description of the backup options is given below:

- **APK files.** Whether to back up the APK files. This includes the *base APK* file along with the `split APK` files if they exist.
- **Internal data.** Whether to back up the internal data directories. These directories are located at `/data/user/<user_id>` and (for Android N or later) `/data/user_de/<user_id>`.

- **External data.** Whether to back up data directories located in the internal memory as well as SD Card (if exists). External data directories often contain non-essential app data or media files (instead of using the dedicated media folder) and may increase the backup size. However, it might be essential for some apps. Although it isn't checked by default (as it might dramatically increase the size of the backups), you may have to check it in order to ensure a smooth restore of your backups.

  > ***Caution.*** Internal data folders should always be backed up if you are going to back up the external data folders. However, it could be useful to back up only the external folders if the app in question downloads a lot of assets from the Internet.

- **OBB and media.** Whether to back up or restore the OBB and the media directories located in the external storage or the SD Card. This is useful for games and the graphical software which actually use these folders.

- **Cache.** Android apps have multiple cache directories located at every data directories (both internal and external). There are two types of cache: **cache** and **code cache**. Disabling this option excludes both cache directories from all the data directories. It is generally advised to exclude cache directories since most apps do not clear the cache regularly and usually handled by the OS itself. Apps such as Telegram may use a very large cache (depending on the storage space) which may dramatically increase the backup size. When it is disabled, AM also ignores the **no_backup** directories.

- **Extras.** Backup/restore app permissions, net policy, battery optimization, SSAID, etc., enabled by default. Note that, blocking rules are applied *after* applying the extras. So, if an item is present in both places, it will be overwritten (i.e., the one from the blocking rules will be used).

- **Rules.** This option lets you back up blocking rules configured within App Manager. This might come in handy if you have customised permissions or block some components using App Manager as they will also be backed up or restored when you enable this option.

- **Backup Multiple.** Whether this is a multiple backup. By default, backups are saved using their user ID. Enabling this option allows you to create additional backups. These backups use the current date-time as the default backup name, but you can also specify custom backup name using the input field displayed when you click on the **Backup** button.

- **Custom users.** Backup or restore for the selected users instead of only the current user. This option is only displayed if the system has more than one user.

- **Skip signature checks.** When taking a backup, checksum of every file (as well as the signing certificate(s) of the base APK file) is generated and stored in the `checksums.txt` file. When you restore the backup, the checksums are generated again and are matched with the checksums stored in the said file. Enabling this option will disable the signature checks. This option is applied only when you restore a backup. During backup, the checksums are generated regardless of this option.

  > ***Caution.*** You should always disable this option to ensure that your backups are not modified by any third-party applications. However, this would only work if you enabled encryption.

See also: Settings: Encryption

3.3.3 Backup

Backup respects all the backup options except **Skip signature checks**. If base backups (i.e., backups that don't have the **Backup Multiple** option) already exist, you will get a warning as the backups will be overwritten. If **Backup Multiple** is set, you have an option to input the backup name, or you can leave it blank to use the current date-time.

3.3.4 Restore

Restore respects all the backup options and will fail if **APK files** option is set, but the backup doesn't contain such backups or in other cases, if the app isn't installed. When restoring backups for multiple packages, you can only restore the base backups (see backup section for an explanation). However, when restoring backups for a single package, you have the option to select which backup to restore. If **All users** option is set, AM will restore the selected backup for all users in the latter case but in the former case, it will restore base backups for the respective users.

> ***Notice.*** Apps that use storage access framework (SAF), SSAID or Android KeyStore works properly only after an immediate restart.

3.3.5 Delete Backup

Delete backup only respects **All users** option and when it is selected, only the base backups for all users will be deleted with a prompt. When deleting backups for a single package, another dialog will be displayed where you can select the backups to delete.

3.4 Automating Tasks

It is possible to trigger profiles configured inside App Manager via third-party applications such as **Automation** or **Tasker**. Traditionally, `Intents` are used to trigger such operations.

3.4.1 Generating authorization key

In order to ensure proper security, an authorization key is required. To generate a authorization key, go to **Settings** page and then **Privacy** > **Authorization Manager**. If an authorization key has not been generated, it will be generated automatically. The key can be regenerated as required.

> ***Caution.*** Regenerating the authorization key can have some side effects such as invalidation of all the previously configured Intents.

3.4.2 Configuring tasks

The activity `io.github.muntashirakon.AppManager.crypto.auth.AuthFeatureDemultiplexer` is responsible for handling all the automations. Sending an intent to the activity lets App Manager perform the designated operation by redirecting the `Intent` to the designated activity or service.

3.4.2.1 Required extras

It has two primary extras required in all conditions. The key names, data types are all follows:

1. **`auth`.** (String value) The authorization key as described in the earlier section.
2. **`feature`.** (String value) Name of the feature. Supported features are described in the next section.

3.4.3 Features

App Manager current support a single feature, namely `profile`.

3.4.4 Triggering a profile

In order to trigger a profile, `feature` must have the value `profile`. In addition, the following extras can be included:

1. **`prof`.** (String value – required) The name of the profile as displayed in the Profiles page.

2. `state`. (String value – optional) State of the profile – currently `on` or `off` – as specified in the documentation. If this extra is not set, App Manager will display a prompt where a state must be selected. Therefore, for complete automation, this option should be set.

3.5 Net Policy

Short for **Network policy** or network policies. It is usually located in the Android settings under **Mobile data & Wifi** section in the app info page of an app. Not all policies are guaranteed to be included in this page (e.g. Samsung), and not all settings are well-understood due to lack of documentation. App Manager can display all the net policies declared in the NetworkPolicyManager. Policies unknown to App Manager will have a *Unknown* prefix along with the policy constant name and number in the hexadecimal format. Unknown policies should be reported to App Manager for inclusion.

Net policy allows a user to configure certain networking behaviour of an app without modifying the ip tables directly and/or running a firewall app. However, the features it offers largely depend on Android version and ROM. A list of known net policies are listed below:

1. **None** or `POLICY_NONE`: (AOSP) No specific network policy is set. System can still assign rules depending on the nature of the app.
2. **Reject background data on metered networks** or `POLICY_REJECT_METERED_BACKGROUND`: (AOSP) Reject network usage on metered networks when the application is in background.
3. **Allow background data on metered networks even when Data Saver is on** or `POLICY_ALLOW_METERED_BACKGROUND`: (AOSP) Allow metered network use in the background even when data saving mode is enabled.
4. **Reject cellular data** or `POLICY_REJECT_CELLULAR` (Android 11+) or `POLICY_REJECT_ON_DATA` (up to Android 10): (Lineage OS) Reject mobile/cellular data. Signals network unavailable to the configured app as if the mobile data is inactive.
5. **Reject VPN data** or `POLICY_REJECT_VPN` (Android 11+) or `POLICY_REJECT_ON_VPN` (up to Android 10): (Lineage OS) Reject VPN data. Signals network unavailable to the configured app as if the VPN is inactive.
6. **Reject Wi-Fi data** or `POLICY_REJECT_WIFI` (Android 11+) or `POLICY_REJECT_ON_WLAN` (up to Android 10): (Lineage OS) Reject Wi-Fi data. Signals network unavailable to the configured app as if the device is not connected to a Wi-Fi network.
7. **Disable network access** or `POLICY_REJECT_ALL` (Android 11+) or `POLICY_NETWORK_ISOLATED` (up to Android 10): (Lineage OS) Reject network access in all circumstances. This is not the same as enforcing the other three policies above, and is the recommended policy for dodgy apps. If this policy is enforced, there is no need to enforce the other policies.
8. `POLICY_ALLOW_METERED_IN_ROAMING`: (Samsung) Possibly allow metered network use during roaming. Exact meaning is currently unknown.
9. `POLICY_ALLOW_WHITELIST_IN_ROAMING`: (Samsung) Possibly allow network use during roaming. Exact meaning is currently unknown.
10. **Reject data on metered networks** or `POLICY_REJECT_METERED`: (Motorola) Reject network usage if it is a metered network.
11. **Reject background data** or `POLICY_REJECT_BACKGROUND`: (Motorola) Reject network usage in the background.
12. **Disable network access** or `POLICY_REJECT_ALL`: (Motorola) Reject network access altogether. Like Lineage OS, it blocks internet connections via iptables. But whether it signals the unavailability of network to the configured app is not known.

Note. Corresponding Lineage OS patches are as follows:

1. fw/b: Squash of app fw restriction commits
2. fw/b: Add support for per app network isolation

Chapter 4

Frequently Asked Questions

4.1 App Components

4.1.1 What are the application components?

Activities, services, broadcast receivers (or only receivers) and content providers (or only providers) are jointly called application components. More technically, they all inherit the ComponentInfo class and can be launched via Intent.

4.1.2 How are the tracker and other components blocked in App Manager? What are its limitations?

App Manager typically blocks application components (or tracker components) using a method called Intent Firewall (IFW), it is superior to other methods such as *pm* (PackageManager), Shizuku or any other method that uses the package manager to enable or disable the components. If a component is disabled by the latter methods, the application itself can detect that the component is being blocked and can re-enable it as it has full access to its own components. (Many deceptive applications actually do this in order to keep the tracker components unblocked.) On the other hand, IFW is a true firewall and the application cannot detect if its components are being blocked. App Manager uses the term *block* rather than *disable* for this reason.

Even IFW has some limitations which are primarily applicable for the system applications:

- The application in question is whitelisted by the system i.e. the system cannot function properly without these applications and may cause random crashes. These applications include but not limited to Android System, System UI, Phone Services. They will continue to work even if they are disabled or blocked.

- Another system application or system process has activated a specific component of the application in question via interprocess communication (IPC). In this case, the component will be activated regardless of blocking status or even if the entire application is disabled. If there is such a system application that is not needed, the only way to prevent it from running is by getting rid of it.

4.1.3 Does app components blocked by other tools retained in App Manager?

No. But the application components blocked by the system or any other tools are displayed in the component tabs. These rules can be imported from Settings. However, it is not possible for App Manager to distinguish the components blocked by the third-party tools and components blocked by the system. Therefore, the applications listed in the import page should be selected with care.

4.1.4 What happens to the components blocked by App Manager which were previously blocked by other tools?

App Manager blocks the components again if requested. In case of unblocking, they will be reverted to the default state as specified in the manifest of the application. But if the components were blocked by MyAndroidTools (MAT) with IFW method, they will not be unblocked by App Manager as it uses a different format. To fix this issue, the rules have to be imported from Settings at first, in which case MAT's configurations will be permanently removed.

4.1.5 What is instant component blocking?

When you block a component in the App Details page, the blocking is not applied by default. It is only applied when you apply blocking using the *Apply rules* option in the top-right menu. If you enable instant component blocking, blocking will be applied as soon as you block a component. If you choose to block tracker components, however, blocking is applied automatically regardless of this setting. You can also remove blocking for an application by simply clicking on *Remove rules* in the same menu in the **App Details page**. Since the default behaviour gives you more control over applications, it is better to keep *instant component blocking* option disabled.

4.1.6 Tracker classes versus tracker components

All application components are classes but not all classes are components. In fact, only a few of the classes are components. That being said, scanner page displays a list of trackers along with the number of classes, not just the components. In all other pages, trackers and tracker components are used synonymously to denote tracker components, i.e. blocking tracker means blocking tracker components, not tracker classes.

> ***Info.*** Tracker classes that are not components cannot be blocked. They can only be removed by editing the application itself.

4.2 ADB over TCP

4.2.1 Do I have to enable ADB over TCP everytime I restart?

Unfortunately, yes. This is because the ADB daemon, the process responsible for ADB connection, is also restarted after a reboot, and it does not re-enable ADB over TCP.

4.2.2 Cannot enable USB debugging. What to do?

See §**??** in Chapter **??**.

4.2.3 Can I block tracker or any other application components using ADB over TCP?

ADB has limited number of permissions and controlling application components is not one of them. However, the components of a *test-only* app can be controlled via ADB. If App Manager detects such an application, it enables the blocking options automatically.

4.2.4 Which features can be used in ADB mode?

Supported features are enabled automatically in the ADB mode. Supported features include disabling, force-stopping, clearing application data, granting or revoking app ops and permissions, and so on. It is also possible to install or uninstall applications without any prompt from the system.

4.3 Miscellanea

4.3.1 I don't use root/ADB. Am I completely safe from any harms?

Yes. AM cannot modify any system settings without root or ADB.

4.3.2 How are the trackers and libraries are updated?

Trackers and libraries are updated manually before making a new release.

4.3.3 Are APKs deleted after installed?

No, APKs aren't deleted by App Manager after they are installed.

4.3.4 Any plans for Shizuku?

App Manager's use of hidden API and privileged code execution is now much more complex and cannot be integrated with other third party apps such as Shizuku. Here are some reasons for not considering Shizuku (which now has Apache 2.0 license) for App Manager:

1. Shizuku was initially non-free which led me to use a similar approach for App Manager to support both root and ADB
2. App Manager already supports both ADB and root which in some cases is more capable than Shizuku
3. Relying on a third-party app for the major functionalities is not a good design choice
4. Integration of Shizuku will increase the complexity of App Manager.

4.3.5 What are bloatware and how to remove them?

Bloatware are the unnecessary pre-installed apps, usually system apps. Some of the apps are often used to track users and collect user data which they might sell for profits. Many system apps do not need to request any permission to access device info, contacts and messaging data, and other usage info, such as your phone usage habits and everything you store on your shared storage(s).

The bloatware may also include Google apps, Meta apps, and Twitter/X which can also track users and/or collect user data without consent. You can disable a few permissions from Android settings but be aware that Android settings hides many permissions a security researcher would call potentially *dangerous* (e.g., internet, sensor).

Were the bloatware user apps, they could be easily uninstalled either from Android settings or AM. Uninstalling system apps is not possible without privileged permission, but even then, it cannot *remove* the system apps completely as they are located in the *system* partition which is a read-only partition. If you have root, you can remount this partition to manually *purge* these apps but this will break Over the Air (OTA) updates since data in the system partition has been modified. There are two kind of updates, delta (small-size, consisting of only the changes between two versions) and full updates. You may still be able to apply full updates, but the bloatware will be installed again, and consequently, you have to delete them all over again.

Another solution is to disable these apps either from Android settings or AM, but certain services can still run in the background as they can be started by other system apps using Inter-process Communication (IPC). One possible solution is to disable all bloatware until the service has finally stopped (after a restart). However, due to heavy modifications of the Android frameworks by the vendors, removing or disabling certain bloatware may cause the System UI to crash or even cause bootloop. From v4.0.0, AM has a new feature called **Debloater** which can be used as a starting point to monitor, disable, and remove the bloatware from a proprietary Android operating system.

Note. In most cases, you cannot completely debloat your device. Therefore, it is recommended that you use a custom ROM free from bloatware, such as Graphene OS, Lineage OS or their derivatives.

Appendix A

Specifications

A.1 Rules Specification

A.1.1 Background

AM currently supports blocking activities, broadcast receivers, content providers, services, app ops and permissions, and in future I may add more blocking options. In order to add more portability, it is necessary to import/export all these data.

Maintaining a database should be the best choice when it comes to storing data. For now, several `tsv` files with each file having the name of the package and a `.tsv` extension. The file/database will be queried/processed by the `RulesStorageManager` class. Due to this abstraction, it should be easier to switch to database or encrypted database systems in future without changing the design of the entire project. Currently, All configuration files are stored at `/data/data/io.github.muntashirakon.AppManager/Files/conf`.

A.1.2 Rules File Format

A.1.2.1 Internal

The format below is used internally within App Manager and is *not compatible with the external format.*

```
<name> <type> <mode>|<component_status>|<is_granted>
```

Here:

- `<name>` – Component/permission/app op name (in case of app op, it could be string or integer)
- `<type>` – One of the `ACTIVITY`, `RECEIVER`, `PROVIDER`, `SERVICE`, `APP_OP`, `PERMISSION`
- `<mode>` – (For app ops) The associated mode constant
- `<component_status>` – (For components) Component status
  - `true` – Component has been applied (`true` value is kept for compatibility)
  - `false` – Component hasn't been applied yet, but will be applied in future (`false` value is kept for compatibility)
  - `unblocked` – Component is scheduled to be unblocked
- `<is_granted>` – (For permissions) Whether the permission is granted or revoked

A.1.2.2 External

External format is used for importing or exporting rules in App Manager.

```
<package_name> <component_name> <type> <mode>|<component_status>|<is_granted>
```

This the format is essentially the same as above except for the first item which is the name of the package.

Caution. The exported rules have a different format than the internal one and should not be copied directly to the **conf** folder.

Appendix B

Changelogs

B.1 v4.0.5 (445)

- Added option to allow installing the existing applications in the installer page
- Display unsafe bloatware info
- Display vector icon on the splash screen in Android 7.1 and earlier
- Enabled “Clear data from uninstalled apps” to no-root users in 1-click ops page
- Enabled predictive back in Android 14 onwards
- Improved accessibility by updating the content description of the action items
- In app info tab, open “Open by default” setting in the “Open links” dialog
- In debloater page, sort by app label (or app name) rather than package name
- Fixed freezing an app with “Remember for this app” turned on
- Fixed selecting texts in the list items due to framework bugs
- Fixed setting app ops in custom ROMs with MIUI properties injected
- Fixed updating profile modification status when an application is deleted from the list
- Handled common colors and cursor movements in terminal output.

B.2 v4.0.4 (444)

- Optimized searching and filtering in the main page
- Adopted sentence-case for the titles
- Use the configured state to execute a profile for the simple shortcuts
- Prevented crashing while searching throughout the application
- Fixed integer overflow issue in the tar compression.

B.3 v4.0.3 (443)

- Updated translations
- Improved handling the list items throughout App Manager
- Fixed a regression error in file manager
- Fixed spinners in the App Usage and the System Config pages.

B.4 v4.0.2 (442)

- Updated bloatware
- Fixed fetching applications in multi-user environment in no-root mode
- Fixed opening `app-manager` URLs from the web browsers
- Fixed updating SSAID
- Prevented a crash in Android < 9.0 that occurs due to invalid app ops.

B.5 v4.0.1 (441)

B.5.1 Overlay management

In the App Details page, a new tab “Overlays” is added where per-app overlays are displayed. They can also be enabled or disabled using the toggle button. In addition, if the App Details page of an overlay package is opened, a “Overlay” tag will be displayed in the App Info tab. Clicking on the tag opens a dialog containing additional info along with a button that allows navigating to the App Details page of the overlay target package if it is installed.

Known limitation At present, it only works for root/ADB users in Android 8 (Oreo) and later.

B.5.2 Unfreeze option in activity shortcuts

If the application corresponding to the shortcut being launched is frozen, App Manager will now offer you to unfreeze the app temporarily so that the shortcut can be launched. The app will be frozen again once the screen is locked.

Known limitation This may not work on devices without a screen lock or if the screen is locked some time after the display goes off.

B.5.3 `market`-like URL support

Third-party applications can now open the App Details page of any installed package by invoking an Intent with an URL with the following format:

```
app-manager://details?id=<pkg>&user=<user_id>
```

where `<pkg>` stands for package name, and `<user_id>` stands for the user ID which is optional.

B.5.4 Updated color codes

In order to improve accessibility, certain color codes have been improved. Visit Settings > About > Version/Changelog for details.

B.5.5 Others

- Avoided waiting for the remote server to respond when no-root mode is set
- Fixed downgrading apps in Android 10 onwards
- Fixed installer issues in the Huawei stock operating systems
- Improved text formatting in the “What’s New” dialog
- In the UI tracker window, fixed clicking on the icon after it is iconified
- Updated bloatware and suggestions

B.6 v4.0.0 (440)

App Manager v4.0.0 comes with a lot of new features and improvements. Visit Settings > About > Version/Changelog for details.

B.6.1 New logo!

The new logo is just a cursive "A". The design is based on the Tengwar Telcontar font which was created to bring the Tengwar script, originally created by J. R. R. Tolkien, to the digital world. The letter has the classic App Manager color (i.e., #dcaf74) and uses a pure black background instead of a shade of grey.

B.6.2 Android 14 and 15 support

App Manager now targets Android 14 and fully supports Android 15.

Known issue KeyStore backup/restore is not working in Android 12 and later.

B.6.3 Revamped debloater

Debloating profiles were available as "Presets" in the Profiles page which has now been replaced with the Debloater page and can be accessed from the three-dots menu in the Main page. ADL is a new project that focuses on maintaining a list of bloatware as well as potential open source alternatives. Contributions are welcome!

B.6.4 Introducing file manager

App Manager offers an (almost) fully-featured file manager with basic file operations, such as copy, cut, rename, and delete along with the batch operations. It also offers an extensive "Open with…" dialog to open a file with another app, and a comprehensive file properties viewer. Folders can also be added to the list of favorites for quick access. And many more.

B.6.5 Integrated code editor

Manifest and code viewers have been replaced with this new editor. Among other regular features, it includes proper syntax highlighting and advanced searching options. In addition, files from third-party apps can also be opened for editing.

B.6.6 History of operations

All 1-click operations, batch operations, and profile invocations are now stored as history. The history items can also be executed from the History page. To ensure consistency, the profile state, configurations, package list are also stored, and this stored version is executed instead of the actual profile. As a result, this works even if the profile is deleted.

B.6.7 Per-app freezing, and more

Freeze/unfreeze feature now supports setting per-app freezing method which is beneficial in certain scenarios, such as when a user want to suspend some apps while using the disable method as the default. In addition, an "Advanced suspend" option is added which force-stops an application before suspending it, thus, prevent it's services from running in the background.

B.6.8 Log viewer enhancements

Log viewer now supports enhanced searching and filtering options, such as keyword- and regular expression-based searching and filtering. Please read the in-app changelog for details. Support for batch operations has also been added.

B.6.9 Launching non-exported activities

App Manager now supports launching non-exported activities in no-root and ADB mode. However, in no-root mode, `android.permission.WRITE_SECURE_SETTINGS` permission is required.

B.6.10 New tags in App Info tab

Five new tags are added in the App Info tab. They are: bloatware, Xposed, sensors disabled, open links, and static shared libs. Clicking on "bloatware" will display more information regarding the bloatware and suggest alternatives, "Xposed" tag will display dependency information, "open links" will display a list of links supported by the application, and "static shared libs" will display all version of the application installed in the system along with an option to uninstall them. The latter is useful for applications, such as Trichrome.

Known issue "Sensors disabled" only works real-time. That means if the application is not currently active, this tag will always display even though the applications may use sensors while it is running. This is a framework limitation and nothing can be done to avoid it effectively.

B.6.11 Per-session installer options

It is not possible to modify installer options during the installation by clicking on the "settings" button in the installation dialog. The installer options will be applied to all the applications installed in the same session (i.e., the installer queue).

B.6.12 Advanced mode of operations, ADB enhancements, ...

App Manager now supports running its remote server (which is used as a proxy for running privileged operations) as any supported user (UID). This includes root (0), system (1000), and shell/ADB (2000) through the custom commands. This is also useful for Fire TVs which have disabled connecting to ADB from localhost through socket connection. In addition, ADB pairing is now done using notifications rather than split screen. ADB connection speed can also be improved by choosing to run App Manager in the background which can be configured in the settings.

B.6.13 Data usage widget, and more

Data usage widget display the total data usage for the day, similar to the screen time widget which displays the total screen time for the day. In addition, existing widgets have been improved.

B.6.14 Others

- Replaced log viewer, sys config, Terminal, etc. with Labs page
- Added an option to disable sensors for each app in the App Info tab
- Added an option to perform runtime optimization of applications in the 1-click Ops page and in the App Info tab
- Added support for Zstandard compression for backup/restore
- Enabling APK signing now automatically enables zip align feature
- Support exporting application list as CSV or JSON in the batch operations
- Added pure black theme support
- Display current activity name (when possible) in the UI Tracker window
- Added an option to filter apps by user in the Main page
- Display a link to Pithus report in the scanner page if available.

B.7 v3.1.0 (423)

App Manager v3.1.0 comes with a few new features and a lot of improvements. Visit Settings > About > Version/Changelog for details.

B.7.1 Android 13 support

App Manager now targets Android 13 which means most issues in Android 12 and 13 has been addressed, including SSAID and SAF issues as well as monochrome icons and other theming issues.

Known issue KeyStore backup/restore not working in Android 12 and later.

B.7.2 Introducing freeze/unfreeze

Enable/disable is replaced with freeze/unfreeze to allow greater control on the behaviours of an app. It supports suspend, disable and hide functionalities which can be controlled at Settings > Rules > Default freezing method. In order to make it easy to freeze or unfreeze an app, shortcuts can also be created from the App Info tab by long clicking on the freeze or unfreeze button.

B.7.3 Export app list

In the Main page, it is now possible to export the list of apps in either XML or Markdown format using batch operations. In the future, the XML file may also be imported to App Manager.

B.7.4 Elliptic Curve Crypography (ECC)

App Manager now fully supports encrypting backups using ECC in addition to offering AES, RSA and OpenPGP.

B.7.5 New languages

Two new languages are added: Korean and Romanian.

B.7.6 More list options

In the main page, more sorting and filtering options are added. Sorting options include sorting the apps by total size, total data usage, launch count, screen time and last usage time. Filtering options include filtering the apps having at least one item in the Android KeyStore, filtering apps with URIs granted via SAF, and filtering apps with SSAID.

B.7.7 Improved handling of mode of operation

Fixed various issues with ADB pairing, handled incomplete USB debugging. Some rooting methods cannot allow interprocess communication via Binder. In those cases, ADB mode is used as a fallback method by enabling it automatically if possible.

B.7.8 Handling multiple users

When possible, App Manager will be able to display apps from work profile in no-root mode in addition to allowing basic operations such as launching the app or navigating to the system settings. For backups, it is now possible to restore backups for other users, but for work profile, some apps may only work properly after re-enabling the work profile. In the installer page, selecting *All users* will now install the app for all users instead of only the current user. Finally, in the app info tab, current app can be installed in another profile using the *Install for…* option available in the three-dots menu. This is analogous to the `pm install-existing` command, thereby, making the installation process a lot faster.

B.7.9 Explorer enhancements

Explorer can now open DEX and JAR files in addition to APK files. Several sorting options as well as folder options are also added as the list options.

B.7.10 New tag: WX

In app info tab, a new tag called WX is added. It is displayed in Android 10 and later if the application targets Android 9 or earlier. It indicates W^X violation which allows the app to execute arbitrary executable files either by the modification of executables embedded within the app or by downloading them from the Internet.

B.7.11 App ops management

App ops are now managed automatically to avoid various app ops related crashes in various platforms. This will also lessen the amount of crashes in an unsupported operating system.

B.7.12 Batch uninstallation

In the Main page, enabled batch uninstallation in no-root mode.

B.7.13 Running apps

Enabled advanced searching. Searching now matches not only app labels but also package names.

B.7.14 Interceptor

Copy the intercepted Intent as am command which can be run from either an ADB shell or a terminal using root with the same effectiveness.

B.7.15 Device-specific changes

Graphene OS Explicitly handle the Internet permission which is a runtime permission in the OS.

MIUI Fixed permission denied issues in the installer due to a framework issue introduced in MIUI 12.5.

Motorola Fixed crashes in the Interceptor page due to a framework issue introduced in Android 11.

B.7.16 Others

- Improved Java-Smali conversion by including all the subclasses during conversion
- Improved scanning performance in the Scanner page
- Improved updating the list of apps in the Main page
- Scan all the available paths to detect systemless-ly installed system apps
- `vacuum` SQLite database before opening it for viewing or editing.

B.8 v3.0.0 (410)

App Manager v3.0.0 comes with a lot of features and improvements. See Settings > About > Version/Changelog to see a more detailed changelog.

B.8.1 Material 3 and More

Material 3, somewhat similar to *Material You*, is a significant improvement over Material Design 2 with support for dynamic colours in Android 12 and later. In addition, many design changes have been made in App Manager without any significant changes in the overall user experience.

Known issue Switches are still based on Material Design 2 which will be fixed in a future release.

B.8.2 Wireless Debugging

Wireless debugging support has been fully implemented. Head over to §**??** for instructions on how to configure wireless debugging.

> ***No-root users.*** Due to auto-detection feature, startup time might be large for no-root users when the mode of operation is set to *auto*. Instead, no-root users should select *no-root* instead of *auto*.

B.8.3 Languages

App Manager is fully translated into Indonesian and Italian languages and can be enabled in settings. Bengali is removed due to lack of translators.

B.8.4 Introducing App Explorer

App Explorer can be used to browse the contents of an application. This includes binary XML files, DEX contents or any other media files. DEX contents can only be explored in Android Oreo (Android 8) and later. It's also possible to convert an `.smali` file into `.java` for a better understanding of the reversed code. This feature, if not needed, can be disabled in Settings > Enable/disable features.

B.8.5 Import Backups from Other Applications

It is possible to import backups from discontinued or obsolete applications such as Titanium Backup, OAndBackup and Swift Backup (version 3.0 to 3.2). Go to Setting > Backup/restore to find this option.

B.8.6 VirusTotal

VirusTotal is a widely used tool to scan files and URLs for viruses. In the scanner page and in the running apps page, an option to scan files with VirusTotal has been added. But the option is hidden by default. To enable the option, it is necessary to obtain an API key from VirusTotal. Go to Settings > VirusTotal API Key for more information.

> ***Internet feature.*** This is currently the only feature which require an Internet connection. If you wish to use any Internet feature that might also be added in the future, enable *Use the Internet* in Settings > Enable/disable features.

B.8.7 Trigger Profiles from the Automation Software

As the implementation of routine operations is being delayed, an option to trigger profiles from the external automation software is added. See §**??** for instructions on how to configure profile automation.

B.8.8 Improved Application Installer

Application installer includes several improvements including the ability to downgrade applications in no-root mode, installing multiple applications at once and blocking trackers after installation. In Android 12 and later, no-root users can update applications without any user interactions.

B.8.9 Component Blocking

It is now possible to configure how App Manager should block a component. Visit Settings > Rules > Default blocking method for more information. In the components tab, long clicking the block/unblock button opens a context menu which allows per-component blocking in a similar manner. ADB users can also block the components of a *Test only* app.

B.8.10 Advanced Searching

In some pages, the search bar supports additional searching which includes searching via prefix, suffix or even regular expressions. In the main page, it is also possible to search for applications using the first letters of each word, e.g. *App Manager* can be listed by searching for *am*.

B.8.11 Shared Libraries

Shared libraries tab has received a significant improvements. It can display three types of libraries, such as native, jar and APK files.

B.8.12 Make the Best Use of Interceptor

Activity interceptor can be opened directly from the activities tab by long clicking on the launch button, and similarly, activities can be launched from the activity interceptor page with or without root, for any users.

> ***Notice.*** Currently, activities opened via root cannot send the results back to the original applications.

B.8.13 Widget: Screen Time

Screen time widget is quite similar to Digital Wellbeing's widget by the same name. It displays the total screen time for the day along with the top three apps from all users.

B.8.14 Widget: Clear Cache

Clear cache widget can be to clear cache from all the applications directly from the home screen.

B.9 v2.6.0 (385)

B.9.1 Introducing Backups

Back up/restore feature is now finally out of beta! Read the corresponding guide to understand how it works.

B.9.2 Introducing Log Viewer

Log viewer is essentially a front-end for `logcat`. It can be used to filter logs by *tag* or *pid* (process ID), or even by custom filters. Log levels AKA verbosity can also be configured. You can also save, share and manage logs.

B.9.3 Lock App Manager

Lock App Manager with the screen lock configured for your device.

B.9.4 Extended Modes for App Ops

You can set any mode for any app ops that your device supports, either from the 1-click ops page or from the app ops tab.

B.9.5 New Batch Ops: Add to Profile

You can now easily add selected apps to an existing profile using the batch operations.

B.9.6 App Info: Improved

App info tab now has many options, including the ability to change SSAID, network policy (i.e. background network usage), battery optimization, etc. Most of the tags used in this tab are also clickable, and if you click on them, you will be able to look at the current state or configure them right away.

B.9.7 Advanced Sort and Filtering Options in the Main Page

Sort and filter options are now replaced by List Options which is highly configurable, including the ability to filter using profiles.

B.9.8 About This Device

Interested in knowing about your device in just one page? Go to the bottom of the settings page.

B.9.9 Enable/disable Features

Not interested in all the features that AM offers? You can disable some features in settings.

B.9.10 New Languages

AM now has more than 19 languages! New languages include Farsi, Japanese and Traditional Chinese.

B.9.11 Signing the APK Files

You can now import external signing keys in AM! For security, App Manager has its own encrypted Key-Store which can also be imported or exported.

B.9.12 New Extension: UnAPKM

Since APKMirror has removed encryption from their APKM files, it's no longer necessary to decrypt them. As a result, the option to decrypt APKM files has been removed. Instead, this option is now provided by the UnAPKM extension which you can grab from F-Droid. So, if you have an encrypted APKM file and have this extension installed, you can open the file directly in AM.

B.10 v2.5.20 (375)

B.10.1 Introducing Profiles

Profiles finally closes the related issue. Profiles can be used to execute certain tasks repeatedly without doing everything manually. A profile can be applied (or invoked) either from the Profiles page or from the home screen by creating shortcuts. There are also some presets which consist of debloating profiles taken from Universal Android Debloater.

Known limitations

- Exporting rules and applying permissions are not currently working.
- Profiles are applied for all users.

B.10.2 The Interceptor

Intent Intercept works as a man-in-the-middle between source and destination, that is, when you open a file or URL with another app, you can see what is being shared by opening it with Interceptor first. You can also add or modify the intents before sending them to the destination. Additionally, you can double-click on any exportable activities in the Activities tab in the App Details page to open them in the Interceptor to add more configurations.

Known limitation Editing extras is not currently possible.

B.10.3 UnAPKM: DeDRM the APKM files

When I released a small tool called UnAPKM, I promised that similar feature will be available in App Manager. I am proud to announce that you can open APKM files directly in the App Info page or convert them to APKS or install them directly.

B.10.4 Multiple user

App manager now supports multiple users! For now, this requires root or ADB. But no-root support is also being considered. If you have multiple users enabled and click on an app installed in multiple profiles, an alert prompt will be displayed where you can select the user.

B.10.5 Vive la France!

Thanks to the contributors, we have one more addition to the language club: French. You can add more languages or improve existing translations at Weblate.

B.10.6 Report crashes

If App Manager crashes, you can now easily report the crash from the notifications which opens the share options. Crashes are not reported by App Manager, it only redirects you to your favourite Email client.

B.10.7 Android 11

Added support for Android 11. Not everything may work as expected though.

B.10.8 App Installer Improvements

B.10.8.1 Set installation locations

In settings page, you can set install locations such as auto (default), internal only and prefer external.

B.10.8.2 Set APK installer

In settings page, you can also set default APK installer (root/ADB only) instead of App Manager.

B.10.8.3 Multiple users

In settings page, you can allow App Manager to display multiple users during APK installation.

B.10.8.4 Signing APK files

In settings page, you can choose to sign APK files before installing them. You can also select which signature scheme to use in the *APK signing* option in settings.

Known limitation Currently, only a generic key is used to sign APK files

B.11 v2.5.17 (368)

B.11.1 App Installer

As promised, it is now possible to select splits. AM also provides recommendations based on device configurations. If the app is already installed, recommendations are provided based on the installed app. It is also possible to downgrade to a lower version without data loss if the device has root or ADB. But it should be noted that not all app can be downgraded. Installer is also improved to speed up the installation process, especially, for root users. If the app has already been installed and the new (x)apk(s) is newer or older or the same version with a different signature, AM will display a list of changes similar to **What's New** before prompting the user to install the app. This is useful if the app has introduced tracker components, new permissions, etc.

Known Limitations

- Large app can take a long time to fetch app info, and therefore, it may take a long time display the installation prompt.
- If the apk is not located in the internal storage, the app has to be cached first which might also take a long time depending on the size of the apk.

B.11.2 Scanner: Replacement for Exodus Page

Exodus page is now replaced with scanner page. Scanner page contains not only a list of trackers but also a list of used libraries. This is just a start. In the future, this page will contain more in depth analysis of the app.

B.11.3 Introducing System Config

System Config lists various system configurations and whitelists/blacklists included in Android by either OEM/vendor, AOSP or even some Magisk modules. Root users can access this option from the overflow menu in the main page. There isn't any official documentation for these options therefore it's difficult to write a complete documentation for this page. I will gradually add documentations using my own knowledge. However, some functions should be understandable by their name.

B.11.4 More Languages

Thanks to the contributors, AM now has more than 12 languages. New languages include Bengali, Hindi, Norwegian, Polish, Russian, Simplified Chinese, Turkish and Ukrainian.

B.11.5 App Info Tab

More tags are added in the app info tab such as **KeyStore** (apps with KeyStore items), **Systemless app** (apps installed via Magisk), **Running** (apps that are running). For external apk, two more options are added namely **Reinstall** and **Downgrade**. Now it is possible to share an apk via Bluetooth. For system apps, it is possible to uninstall updates for root/ADB users. But like the similar option in the system settings, this operation will clear all app data. As stated above, exodus has been replaced with scanner.

B.11.6 Navigation Improvements

It's now relatively easy to navigate to various UI components using keyboard. You can use up/down button to navigate between list items and tab button to navigate to UI components inside an item.

B.11.7 Running Apps Page

It is now possible to sort and filter processes in this tab. Also, the three big buttons are replaced with an easy-to-use three dot menu. Previously the memory usage was wrong which is fixed in this version.

B.11.8 Built-in Toybox

Toybox (an alternative to busybox) is bundled with AM. Although Android has this utility built-in from API 23, toybox is bundled in order to prevent buggy implementations and to support API < 23.

B.11.9 Component Blocker Improvements

Component blocker seemed to be problematic in the previous version, especially when global component blocking is enabled. The issues are mostly fixed now.

> ***Caution.*** The component blocking mechanism is no longer compatible with v2.5.6 due to various security issues. If you have this version, upgrade to v2.5.13 or earlier versions first. After that, enable global component blocking and disable it again.

B.11.10 Improvements in the App Details Page

Value of various app ops depend on their parent app ops. Therefore, when you allow/deny an app op, the parent of the app op gets modified. This fixes the issues some users have been complaining regarding some app ops that couldn't be changed.

If an app has the target API 23 or less, its permissions cannot be modified using the `pm grant ...` command. Therefore, for such apps, option to toggle permission has been disabled.

The signature tab is improved to support localization. It also displays multiple checksums for a signature.

B.11.11 App Manifest

Manifest no longer crashes if the size of the manifest is too long. Generated manifest are now more accurate than before.

B.12 v2.5.13 (348)

B.12.1 Bundled App (Split APK)

Bundled app formats such as **apks** and **xapk** are now supported. You can install these apps using the regular installation buttons. For root and adb users, apps are installed using shell, and for non-root users, the platform default method is used.

Known Limitations

- Currently *all* splits apks are installed. But this behaviour is going to change in the next release. If you only need a few splits instead of all, extract the **APKS** or **XAPK** file, and then, create a new zip file with your desired split apks and replace the **ZIP** extension with **APKS**. Now, open it with AM.
- There is no progress dialog to display the installation progress.

B.12.2 Direct Install Support

You can now install **APK**, **APKS** or **XAPK** directly from your favourite browser or file manager. For apps that need updates, a **What's New** dialog is displayed showing the changes in the new version.

Known Limitations

- Downgrade is not yet possible.
- There is no progress dialog to display the installation progress. If you cannot interact with the current page, wait until the installation is finished.

B.12.3 Remove All Blocking Rules

In the Settings page, a new option is added which can be used to remove all blocking rules configured within App Manager.

B.12.4 App Ops

- App Ops are now generated using a technique similar to AppOpsX. This should decrease the loading time significantly in the App Ops tab.
- In the App Ops tab, a menu item is added which can be used to list only active app ops without including the default app ops. The preference is saved in the shared preferences.

Known Limitation Often the App Ops tab may not be responsive. If that's the case, restart App Manager.

B.12.5 Enhanced ADB Support

ADB shell commands are now executed using a technique similar to AppOpsX (This is the *free* alternative of AppOps by Rikka.). This should dramatically increase the execution time.

Known Limitation AM can often crash or become not responsive. If that's the case, restart App Manager.

B.12.6 Filtering in Main Page

Add an option to filter apps that has at least one activity.

B.12.7 Apk Backup/Sharing

Apk files are now saved as `app name_version.extension` instead of `package.name.extension`.

B.12.8 Batch Ops

- Added a foreground service to run batch operations. The result of the operation is displayed in a notification. If an operation has failed for some packages, clicking on the notification will open a dialog box listing the failed packages. There is also a **Try Again** button on the bottom which can be used to perform the operation again for the failed packages.
- Replaced Linux *kill* with **force-stop**.

B.12.9 Translations

Added German and Portuguese (Brazilian) translations.

Known Limitation Not all translations are verified yet.

B.12.10 App Data Backup

Install app only for the current user at the time of restoring backups. Support for split apks is also added.

Data backup feature is now considered unstable. If you encounter any problem, please report to me without hesitation.

Appendix C

App Ops

C.1 Background

App Ops (short hand for **Application Operations**) are used by Android system (since Android 4.3) to control application permissions. The user *can* control some permissions, but only the permissions that are considered dangerous (and Google thinks knowing your phone number isn't a dangerous thing). So, app ops seems to be the one we need if we want to install apps like Facebook and it's Messenger (the latter literary records everything if you live outside the EU) and still want *some* privacy and/or security. Although certain features of app ops were available in Settings and later in hidden settings in older version of Android, it's completely hidden in newer versions of Android and is continued to be kept hidden. Now, any app with **android.Manifest.permission.GET_APP_OPS_STATS** permission can get the app ops information for other applications but this permission is hidden from users and can only be enabled using ADB or root. Still, the app with this permission cannot grant or revoke permissions (actually mode of operation) for apps other than itself (with limited capacity, of course). To modify the ops of other app, the app needs **android.Manifest.permission.UPDATE_APP_OPS_STATS** permissions which isn't accessible via `pm` command. So, you cannot grant it via root or ADB, the permission is only granted to the system apps. There are very few apps who support disabling permissions via app ops. The best one to my knowledge is AppOpsX. The main (visible) difference between my app (AppManager) and this app is that the latter also provides you the ability to revoke internet permissions (by writing ip tables). One crucial problem that I faced during the development of the app ops API is the lack of documentation in English language.

C.2 Introduction to App Ops

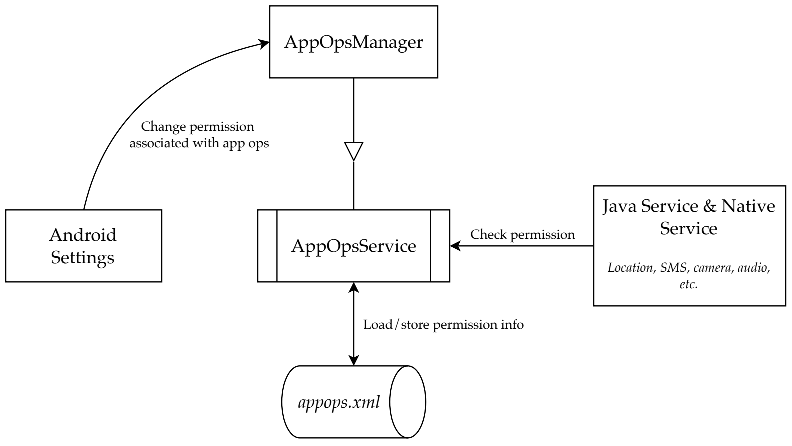
AppOpsManager
Change permission associated with app ops
Android Settings
AppOpsService
Check permission
Java Service & Native Service
Location, SMS, camera, audio, etc.
Load/store permission info
appops.xml

Figure C.1: How app ops work

Figure 1 describes the process of changing and processing permission. AppOpsManager can be used to manage permissions in Settings app. **AppOpsManager** is also useful in determining if a certain permission (or operation) is granted to the application. Most of the methods of **AppOpsManager** are accessible to the user app but unlike a system app, it can only be used to check permissions for any app or for the app itself and start or terminating certain operations. Moreover, not all operations are actually accessible from this Java class. **AppOpsManager** holds all the necessary constants such as `OP_*`, `OPSTR_*`, `MODE_*` which describes operation code, operation string and mode of operations respectively. It also holds necessary data structures such as PackageOps and **OpEntry**. **PackageOps** holds **OpEntry** for a package, and **OpEntry**, as the name suggests, describes each operation.

`AppOpService` is completely hidden from a user application but accessible to the system applications. As it can be seen in Figure 1, this is the class that does the actual management stuff. It contains data structures such as **Ops** to store basic package info and **Op** which is similar to **OpEntry** of **AppOpsManager**. It also has **Shell** which is actually the source code of the *appops* command line tool. It writes configurations to or read configurations from `/data/system/appops.xml`. System services calls **AppOpsService** to find out what an application is allowed and what is not allowed to perform, and **AppOpsService** determines these permissions by parsing `/data/system/appops.xml`. If no custom values are present in *appops.xml*, it returns the default mode available in **AppOpsManager**.

C.3 AppOpsManager

AppOpsManager stands for application operations manager. It consists of various constants and classes to modify app operations.
See also: AppOpsManager documentation

C.3.1 `OP_*` Constants

`OP_*` are the integer constants starting from `0`. `OP_NONE` implies that no operations are specified whereas `_NUM_OP` denotes the number of operations defined in `OP_*` prefix. While they denote each operation, the operations are not necessarily unique. In fact, there are many operations that are actually a single operation denoted by multiple `OP_*` constant (possibly for future use). Vendors may define their own op based on their requirements. MIUI is one of the vendors who are known to do that.

<MINTED>

Listing 1: Sneak-peek of `OP_*`

Whether an operation is unique is defined by `sOpToSwitch`. It maps each operation to another operation or to itself (if it's a unique operation). For instance, `OP_FINE_LOCATION` and `OP_GPS` are mapped to `OP_COARSE_LOCATION`.

Each operation has a private name which are described by `sOpNames`. These names are usually the same names as the constants without the `OP_` prefix. Some operations have public names as well which are described by `sOpToString`. For instance, `OP_COARSE_LOCATION` has the public name **android:coarse_location**.

As a gradual process of moving permissions to app ops, there are already many permissions that are defined under some operations. These permissions are mapped in `sOpPerms`. For example, the permission **android.Manifest.permission.ACCESS_COARSE_LOCATION** is mapped to `OP_COARSE_LOCATION`. Some operations may not have any associated permissions which have `null` values.

As described in the previous section, operations that are configured for an app are stored at `/data/system/appops.xml`. If an operation is not configured, then whether system will allow that operation is determined from `sOpDefaultMode`. It lists the *default mode* for each operation.

C.3.2 `MODE_*` Constants

`MODE_*` constants also integer constants starting from `0`. These constants are assigned to each operation describing whether an app is authorised to perform that operation. These modes usually have associated names such as **allow** for `MODE_ALLOWED`, **ignore** for `MODE_IGNORED`, **deny** for `MODE_ERRORED` (a rather misnomer), **default** for `MODE_DEFAULT` and **foreground** for `MODE_FOREGROUND`.

0. `MODE_ALLOWED`. The app is allowed to perform the given operation
1. `MODE_IGNORED`. The app is not allowed to perform the given operation, and any attempt to perform the operation should *silently fail*, i.e. it should not cause the app to crash
2. `MODE_ERRORED`. The app is not allowed to perform the given operation, and this attempt should cause it to have a fatal error, typically a `SecurityException`
3. `MODE_DEFAULT`. The app should use its default security check, specified in `AppOpsManager`
4. `MODE_FOREGROUND`. Special mode that means "allow only when app is in foreground." This mode was added in Android 10
5. `MODE_ASK`. This is a custom mode used by MIUI whose uses are unknown.

C.3.3 PackageOps

AppOpsManager.PackageOps is a data structure to store all the **OpEntry** for a package. In simple terms, it stores all the customised operations for a package.

<MINTED>

Listing 2: Class `PackageOps`

As can be seen in Listing 2, it stores all **OpEntry** for a package as well as the corresponding package name and its kernel user ID.

C.3.4 OpEntry

AppOpsManager.OpEntry is a data structure that stores a single operation for any package.

<MINTED>

Listing 3: Class `OpEntry`

Here:

- `mOp`: Denotes one of the `OP_*` constants
- `mRunning`: Whether the operation is in progress (i.e. the operation has started but not finished yet). Not all operations can be started or finished this way
- `mMOde`: One of the `MODE_*` constants
- `mAccessTimes`: Stores all the available access times
- `mRejectTimes`: Stores all the available reject times
- `mDurations`: All available access durations, checking this with `mRunning` will tell you for how long the app is performing a certain app operation
- `mProxyUids`: No documentation found
- `mProxyPackageNames`: No documentation found

C.3.5 Usage

TODO

C.4 AppOpsService

TODO

C.5 appops.xml

Latest `appops.xml` has the following format: (This DTD is made by me and by no means perfect, has compatibility issues.)

```
<!DOCTYPE app-ops [

<!ELEMENT app-ops (uid|pkg)*>
<!ATTLIST app-ops v CDATA #IMPLIED>

<!ELEMENT uid (op)*>
<!ATTLIST uid n CDATA #REQUIRED>

<!ELEMENT pkg (uid)*>
<!ATTLIST pkg n CDATA #REQUIRED>

<!ELEMENT uid (op)*>
<!ATTLIST uid
n CDATA #REQUIRED
p CDATA #IMPLIED>

<!ELEMENT op (st)*>
<!ATTLIST op
n CDATA #REQUIRED
m CDATA #REQUIRED>

<!ELEMENT st EMPTY>
<!ATTLIST st
n CDATA #REQUIRED
t CDATA #IMPLIED
r CDATA #IMPLIED
d CDATA #IMPLIED
pp CDATA #IMPLIED
pu CDATA #IMPLIED>

]>
```

The instruction below follows the exact order given above:

- `app-ops`: The root element. It can contain any number of `pkg` or package `uid`
  - `v`: (optional, integer) The version number (default: `NO_VERSION` or `-1`)
- `pkg`: Stores package info. It can contain any number of `uid`
  - `n`: (required, string) Name of the package
- Package `uid`: Stores package or packages info
  - `n`: (required, integer) The user ID
- `uid`: The package user ID. It can contain any number of `op`
  - `n`: (required, integer) The user ID

  - p: (optional, boolean) Is the app is a private/system app
- op: The operation, can contain st or nothing at all
  - n: (required, integer) The op name in integer, i.e. AppOpsManager.OP_*
  - m: (required, integer) The op mode, i.e. AppOpsManager.MODE_*
- st: State of operation: whether the operation is accessed, rejected or running (not available on old versions)
  - n: (required, long) Key containing flags and uid
  - t: (optional, long) Access time (default: 0)
  - r: (optional, long) Reject time (default: 0)
  - d: (optional, long) Access duration (default: 0)
  - pp: (optional, string) Proxy package name
  - pu: (optional, integer) Proxy package uid

This definition can be found at AppOpsService.

C.6 Command Line Interface

appops or cmd appops (on latest versions) can be accessible via ADB or root. This is an easier method to get or update any operation for a package (provided the package name is known). The help page of this command is self-explanatory:

```
AppOps service (appops) commands:
help
Print this help text.
start [--user <USER_ID>] <PACKAGE | UID> <OP>
Starts a given operation for a particular application.
stop [--user <USER_ID>] <PACKAGE | UID> <OP>
Stops a given operation for a particular application.
set [--user <USER_ID>] <[--uid] PACKAGE | UID> <OP> <MODE>
Set the mode for a particular application and operation.
get [--user <USER_ID>] <PACKAGE | UID> [<OP>]
Return the mode for a particular application and optional operation.
query-op [--user <USER_ID>] <OP> [<MODE>]
Print all packages that currently have the given op in the given mode.
reset [--user <USER_ID>] [<PACKAGE>]
Reset the given application or all applications to default modes.
write-settings
Immediately write pending changes to storage.
read-settings
Read the last written settings, replacing current state in RAM.
options:
<PACKAGE> an Android package name or its UID if prefixed by --uid
<OP>      an AppOps operation.
<MODE>    one of allow, ignore, deny, or default
<USER_ID> the user id under which the package is installed. If --user is not
specified, the current user is assumed.
```