With Native Android Toolkit your application will have access to several native Android functions. The way that Native Android Toolkit brings Android features try to be as faithful as possible to native Java implementation, but without giving up the ease of implementation. All features listed below are separated into classes by Native Android Toolkit, for organization, and will be accessible by your application. In addition, all these listed functions are highly well documented clearly and with examples of use.

Dialogs

- Simple Alert Dialogs
- Confirmation Alert Dialogs
- Confirmation Alert Dialogs with Neutral button
- Radial List Alert Dialogs
- Checkbox List Alert Dialogs
- Events so that your application knows everything the user has done with dialogues

Notifications

- Show Toast Notifications
- Send Push Notifications with Actions
- Add Buttons to Push Notifications
- Register Callbacks to know which Button the user touched on Push Notifications

Scheduled Notifications

- Schedule up to 20 Push Notifications to be delivered at any time in the future
- Scheduled Notifications are delivered even if your application is closed or the phone is blocked
- Scheduled Notifications are stored in up to 20 different Channels, until deliver to the user
- Schedule Repetitive Notifications and set time intervals for execution
- Check if a particular Channel has a Scheduled Notification (or repetitive) to be delivered
- Cancel Scheduled Notifications that have not yet been delivered to the user
- Get a list of Channels that do not have Scheduled Notifications to be delivered
- Customize the Notification icon and color of your application in the system Notification Tray
- Deliver Notifications with a trusted, tested and quality algorithm

Sharing

- Share Texture2D with other applications or people, using the default system Sharing flow
- Share String with other applications or people, using the default system Sharing flow
- Copy a String to Clipboard of device
- Get the String from the Clipboard of device
- Take a Screenshot and get the Texture2D asynchronously

Webview

- Open a Chromium based Webview on a Pop-Up inside your application
- Open a full screen Webview
- Access a page asychronously, using POST/GET and get all the Cookies returned by page
- Clean all Webview Cookies data from your application
- Customize the Webview of your application in many ways, changing icons, colors, layout and etc.
- Provide parameters on starting the Webvie, such as Cookies to use, Javascript block and etc.
- Register Callbacks to access to everything the user is doing on the Webview, such as browsed pages

Permissions

- Check if the application has the permission of “Camera”

● Check if the application has the permission of “Coarse Location”
● Check if the application has the permission of “Fine Location”
● Check if the application has the permission of “Microphone”
● Check if the application has the permission of “Files And Media”
● Request the user permission to "Camera"
● Request the user permission to "Coarse Location"
● Request the user permission to "Fine Location"
● Request the user permission to "Microphone"
● Request the user permission to "Files And Media"
● Use the "Permission Requester Wizard". An interface prepared by NAT to explain and request permissions from the user automatically and treated
● Request multiple permissions in a single C# API call

Utils

● Restart the application fully (as if the user closed it and open again)
● Vibrate the device
● Vibrate the device with a pattern
● Vibrate the device in a optimized way for multiple calls per second
● Get device manufacturer name
● Get device model name
● Get device Android API version code
● Make the device Speak (in several languages) a String, using the device Text-To-Speech engine
● Get device current locale name (includes Language code, Currency, Country and much more)
● Enable the Anti-Screenshot for the application (prevents Screen Recording too)
● Convert DP to Pixels
● Convert Pixels to DP
● Convert Pixels Size to Canvas Size
● Get device Notch size in Pixels
● Open the Play Store In App Review pop-up
● Check if the Vibration is available
● Check if the Wi-Fi is enabled
● Check if the device is connected to a Wi-Fi
● Check if has a Headset connected
● Check if the Internet is available
● Check if the Developer Mode is enabled
● Check if the Google Play Services is available in the device
● Check if the device is Rooted
● Check if the Anti-Screenshot is enabled

Settings

● Open the device General Settings
● Open the Settings of your application
● Open device Wi-Fi Settings
● Open device Bluetooth Settings
● Open device Location Settings
● Open the Network Operator Settings
● Open the Internet Toggle Settings

Location

● Check if the GPS Location is enabled on device
● Check if the Network Location is enabled on device
● Check if the Mock Location is enabled on device

● Check if the NAT is currently tracking the device Location
● Check if the NAT is currently using Network or GPS Locations
● Start to track the device Location using GPS or Network as source
● Stop to track the device Location
● Register Callbacks to receive position updates and various other tracking information
● Check if the received position is Emulated/Mock
● Get information such as Address, Country, Coordinates and more, during the position updates that NAT returns
● Check if the Google Maps is currently open
● Open the Google Maps pop-up
● Add/Remove markers on Google Maps
● Display markers with custom Sprite icons
● Register Callbacks to know everything the user does in Google Maps pop-up as touch positions and so on
● Customize the Google Maps interface and use your Google Cloud API key

Camera

● Check if the Camera is supported on device
● Open the Camera
● Open the Video Camera
● Allow the user to Take Photos or Record Videos
● Customize the Camera interface and layout in many different ways
● Provide parameters when opening the Camera, such as disable the flash, allow only the Front/Rear Camera and much more
● Open the reader of QR Code and Barcode. This reader does not depend of no third-party application installed on the user's device
● Generate Texture2D of QR Codes from any String you want
● Set maximum Video Recording times
● Register Callbacks to know everything the user does at the Camera interface
● Have easy access to all Photos/Videos resulting from NAT Camera

Microphone

● Check if the Microphone is supported on device
● Check if the NAT is recording the Microphone
● Start to record the device Microphone
● Stop to record the device Microphone
● Register Callbacks to receive all Microphone recording Events
● Have easy access to all Audios resulting from the NAT Microphone
● Check if the Speech-To-Text is supported on device
● Check if is listening to User Speech
● Start to listening the Speech-To-Text
● Register Callbacks to know what the user was said during the Speech-To-Text

Applications

● Check if a application is installed, using the application Package Name
● Open a application using the Package Name
● Open a application using the Package Name and passing startup parameters
● Get the Package Name of your application
● Check if the Google Play Store is available on the device
● Open a application in the Play Store using Package Name

Date Time

● Open a Hour Picker

● Open a Date Picker
● Optionally specify the start time and end time intervals of Date/Hour Pickers
● Register Callbacks to access the date/hours selected by the user
● Load current time from any NTP server of Internet
● Get elapsed Real Time since the device boot
● Register a Callback to know how much time has passed (reliably) whenever the user minimizes, lose focus, blocking or suspending your application
● Register a Callback to know how much time has passed while your application was closed

Files

● Have access and manage the user device Files
● File Management API Completely compatible with Scope Storage of Android 10+
● Completely backward compatible File Management API with Android 9 or previous versions
● API that obeys all good practices of Android File Management
● API fully structured and developed around Scoped Storage File Management mechanics
● Extremely easy and highly well documented API, with examples and explanations of all Android rules and File Management mechanics
● Access all File Scopes
● Get the real path for any File Scope in the device storage
● Get the Scope Availability
● Get the internal memory usage informations
● Save a Media (image, audio or video) and do it available to the device Gallery
● Set or Unset a folder as Scannable by the System
● Check if a Folder is Scannable by the System
● List all Files/Folders presents in a Directory/Scope
● Check if a Folder/File exists
● Get all Attributes (like size, date of creation, extension and more) of a File or Folder
● Copy a Folder/File to another path
● Move a Folder/File to another path
● Rename a Folder/File
● Delete a Filer or Folder (recursively or not)
● Create Folders
● Create Files (using String, Bytes and more)
● Load all Bytes of a File
● Write All Text for a File
● Read All Text from a File
● Write All Lines for a File
● Read All Lines from a File
● Open a File with the device Default Application
● Start the System File Picker to create or open files (Similar to the "Open" or "Save As" interface of Windows OS)

Tasks

● Using Unity Editor, create Tasks using the same programming logic that you use in C# to create Scripts that can be runned on the background of the device. These scripts are called as "Tasks"
● Tasks can be Scheduled by your application to be runned at any future moment
● Scripts ("Tasks") can do whatever you want, from consulting Internet servers, send Notifications, contain logical comparators like IF, ELSE and WHERE, create or read files and more!
● Scheduled Tasks will run even if your application is closed or the phone is blocked
● Run the Scheduled Tasks with a trusted, tested and quality algorithm
● Customize your Task and make it do whatever you want!
● Schedule the execution of your Task for any future moment, you can even program it to self re-schedule and repeat yourself if you want!
● Check if have a Task running now

● Check if have a Task Scheduled now
● Cancel a Scheduled Task
● Allow your C# code to send or define parameters that will be used by Tasks
● Use Constant and useful values such as "isRooted", "lastExecutionTimeMillis", "applicationVersion" and more
● Use dozens of different Components in the code of your Task
● Easily access Files generated by your Task execution, or create files to your Task access

Audio Player

● Play Audios or Songs using the system Media Player (supports the sound file streamming to play)
● The best option to Play/Preview Audios recorded by the NAT Microphone or MP3 without loading them to Unity
● Get all Audio Metadata without playing it
● Check if the NAT Audio Player is playing something
● Starts to Play a Audio
● Pause a Playing Audio
● Stop Playing a Audio
● Get current Playing Audio duration
● Set a new duration for current Playing Audio
● Set the NAT Audio Player volume
● Register Callbacks so that your application knows everything that is happening while the Audio is Playing

Power Manager

● Check if the Exact Alarms and Reminders is enabled for your application
● Open Alarms and Reminders access Setting of the System
● Check if the device is considered "Problematic" regarding energy savings (because of possible extremely drastic tactics to save battery)
● Request AutoStart ON if the device is considered “Problematic” (Usually this can help with problems on problematic devices)

Google Play Games

● Implement Google Play Games in your application extremely easily and well documented way, with a simple and powerful C# API
● Reduce the amount of libraries you will need to use on your project, because using NAT and NAT Play Games you will only need to use the Unity In App Purchase and Unity Ads/Unity Monetization!
● Check if the user is Signed In
● Do Manual Login
● The user is automatically logged in when entering on your application
● Access the user Icon
● Show all Achievements of the user
● Revel Achievements for the user
● Unlock Achievements for the user
● Increment Achievements for the user
● Load all Event Data
● Increment a Event
● Show Leaderboards for the user
● Submit user scores to Leaderboards
● Check if the Friend List of the user is accessible
● Request Friend List Access for the user
● Load the user Friend List (with informations about each Friend and his icon)
● Open the user Profile Display/Comparation
● Save the game to Google Drive of the user
● Load the game from Google Drive of the user

● Register Callbacks to receive all types of responses from Google Play Games, such as loaded games to cloud, friends list and much more

NAT Dependencies Resolver

● This is the module of passive dependencies resolution of Native Android Toolkit
● It will act whenever you detect a new AAR/JAR added or removed to the project
● As he is a passive dependency manager, he will never install anything new on your project
● It will manage and control only the dependencies of Native Android Toolkit that are already included with Native Android Toolkit when installing it
● Highly well documented
● It does the job of preventing NAT AARs/JARS libraries from conflicting with any other libraries that you have on your project!
● Completely compatible with "Play Services Resolver" or "External Dependency Manager for Unity"
● Thanks to this module, Native Android Toolkit should work without problems with any other libraries/SDKs you have in your project, like Unity Ads, Unity Monetization, Firebase, Facebook SDK, Unity IAP, AdMob and others.

There are much more resources! See this link to a complete list, as it is not possible to include the full list here, as it is very large!