<application>

SYNTAX:

```
<application android:allowTaskReparenting=["true" | "false"]
             android:allowBackup=["true" | "false"]
             android:backupAgent="string"
             android:backupInForeground=["true" | "false"]
             android:banner="drawable resource"
             android:debuggable=["true" | "false"]
             android:description="string resource"
             android:directBootAware=["true" | "false"]
             android:enabled=["true" | "false"]
             android:extractNativeLibs=["true" | "false"]
             android:fullBackupContent="string"
             android:fullBackupOnly=["true" | "false"]
             android:hasCode=["true" | "false"]
             android:hardwareAccelerated=["true" | "false"]
             android:icon="drawable resource"
             android:isGame=["true" | "false"]
             android:killAfterRestore=["true" | "false"]
             android:largeHeap=["true" | "false"]
             android:label="string resource"
             android:logo="drawable resource"
             android:manageSpaceActivity="string"
             android:name="string"
             android:networkSecurityConfig="xml resource"
             android:permission="string"
             android:persistent=["true" | "false"]
             android:process="string"
             android:restoreAnyVersion=["true" | "false"]
             android:requiredAccountType="string"
             android:resizeableActivity=["true" | "false"]
             android:restrictedAccountType="string"
             android:supportsRtl=["true" | "false"]
             android:taskAffinity="string"
             android:testOnly=["true" | "false"]
             android:theme="resource or theme"
             android:uiOptions=["none" | "splitActionBarWhenNarrow"]
             android:usesCleartextTraffic=["true" | "false"]
             android:vmSafeMode=["true" | "false"] >
    . . .
</application>
```

CONTAINED IN:

- <manifest>

CAN CONTAIN:

- <activity>
- <activity-alias>
- <meta-data>
- <service>
- <receiver>
- <provider>
- <uses-library>

DESCRIPTION:

The declaration of the application. This element contains subelements that declare each of the application's components and has attributes that can affect all the components. Many of these attributes (such as `icon`, `label`, `permission`, `process`, `taskAffinity`, and `allowTaskReparenting`) set default values for corresponding attributes of the component elements. Others (such as `debuggable`, `enabled`, `description`, and `allowClearUserData`) set values for the application as a whole and cannot be overridden by the components.

ATTRIBUTES

`android:allowTaskReparenting`

Whether or not activities that the application defines can move from the task that started them to the task they have an affinity for when that task is next brought to the front — `"true"` if they can move, and `"false"` if they must remain with the task where they started. The default value is `"false"`.

The `<activity>` element has its own `allowTaskReparenting` attribute that can override the value set here. See that attribute for more information.

`android:allowBackup`

Whether to allow the application to participate in the backup and restore infrastructure. If this attribute is set to false, no backup or restore of the application will ever be performed, even by a full-system backup that would otherwise cause all application data to be saved via adb. The default value of this attribute is true.

`android:backupAgent`

The name of the class that implement's the application's backup agent, a subclass of `BackupAgent`. The attribute value should be a fully qualified class name (such as, `"com.example.project.MyBackupAgent"`). However, as a shorthand, if the first character of the name is a period (for example, `".MyBackupAgent"`), it is appended to the package name specified in the `<manifest>` element.

There is no default. The name must be specified.

`android:backupInForeground`

Indicates that Auto Backup operations may be performed on this app even if the app is in a foreground-equivalent state. The system shuts down an app during auto backup operation, so use this attribute with caution. Setting this flag to true can impact app behavior while the app is active.

The default value is `false`, which means that the OS will avoid backing up the app while it is running in the foreground (such as a music app that is actively playing music via a service in the `startForeground()` state).

`android:banner`

A drawable resource providing an extended graphical banner for its associated item. Use with the `<application>` tag to supply a default banner for all application activities, or with the `<activity>` tag to supply a banner for a specific activity.

The system uses the banner to represent an app in the Android TV home screen. Since the banner is displayed only in the home screen, it should only be specified by applications with an activity that handles the `CATEGORY_LEANBACK_LAUNCHER` intent.

This attribute must be set as a reference to a drawable resource containing the image (for example `"@drawable/banner"`). There is no default banner.

See Banners in the UI Patterns for TV design guide, and Provide a home screen banner in Get Started with TV Apps for more information.

`android:debuggable`

Whether or not the application can be debugged, even when running on a device in user mode — `"true"` if it can be, and `"false"` if not. The default value is `"false"`.

`android:description`

User-readable text about the application, longer and more descriptive than the application label. The value must be set as a reference to a string resource. Unlike the label, it cannot be a raw string. There is no default value.

`android:directBootAware`

Whether or not the application is *direct-boot aware*; that is, whether or not it can run before the user unlocks the device. If you're using a custom subclass of `Application`, and if any component inside your application is direct-boot aware, then your entire custom application is considered to be direct-boot aware.

> **Note:** During Direct Boot, your application can only access the data that is stored in *device protected* storage.

The default value is `"false"`.

`android:enabled`

Whether or not the Android system can instantiate components of the application — `"true"` if it can, and `"false"` if not. If the value is `"true"`, each component's `enabled` attribute determines whether that component is enabled or not. If the value is `"false"`, it overrides the component-specific values; all components are disabled.

The default value is `"true"`.

`android:extractNativeLibs`

Whether or not the package installer extracts native libraries from the APK to the filesystem. If set to `false`, then your native libraries must be page aligned and stored uncompressed in the APK. No code changes are required as the linker loads the libraries directly from the APK at runtime.

The default value is `"true"`.

`android:fullBackupContent`

This attribute points to an XML file that contains full backup rules for Auto Backup. These rules determine what files get backed up. For more information, see XML Config Syntax for Auto Backup.

This attribute is optional. If it is not specified, by default, Auto Backup includes most of your app's files. For more information, see Files that are backed up.

`android:fullBackupOnly`

This attribute indicates whether or not to use Auto Backup on devices where it is available. If set to `true`, then your app performs Auto Backup when installed on a device running Android 6.0 (API level 23) or higher. On older devices, your app ignores this attribute and performs Key/Value Backups.

The default value is `"false"`.

`android:hasCode`

Whether or not the application contains any code — `"true"` if it does, and `"false"` if not. When the value is `"false"`, the system does not try to load any application code when launching components. The default value is `"true"`.

An application would not have any code of its own only if it's using nothing but built-in component classes, such as an activity that uses the `AliasActivity` class, a rare occurrence.

`android:hardwareAccelerated`

Whether or not hardware-accelerated rendering should be enabled for all activities and views in this application — `"true"` if it should be enabled, and `"false"` if not. The default value is `"true"` if you've set either `minSdkVersion` or `targetSdkVersion` to `"14"` or higher; otherwise, it's `"false"`.

Starting from Android 3.0 (API level 11), a hardware-accelerated OpenGL renderer is available to applications, to improve

performance for many common 2D graphics operations. When the hardware-accelerated renderer is enabled, most operations in Canvas, Paint, Xfermode, ColorFilter, Shader, and Camera are accelerated. This results in smoother animations, smoother scrolling, and improved responsiveness overall, even for applications that do not explicitly make use the framework's OpenGL libraries.

Note that not all of the OpenGL 2D operations are accelerated. If you enable the hardware-accelerated renderer, test your application to ensure that it can make use of the renderer without errors.

For more information, read the Hardware Acceleration guide.

`android:icon`

An icon for the application as whole, and the default icon for each of the application's components. See the individual `icon` attributes for `<activity>`, `<activity-alias>`, `<service>`, `<receiver>`, and `<provider>` elements.

This attribute must be set as a reference to a drawable resource containing the image (for example `"@drawable/icon"`). There is no default icon.

`android:isGame`

Whether or not the application is a game. The system may group together applications classifed as games or display them separately from other applications.

The default is `false`.

`android:killAfterRestore`

Whether the application in question should be terminated after its settings have been restored during a full-system restore operation. Single-package restore operations will never cause the application to be shut down. Full-system restore operations typically only occur once, when the phone is first set up. Third-party applications will not normally need to use this attribute.

The default is `true`, which means that after the application has finished processing its data during a full-system restore, it will be terminated.

`android:largeHeap`

Whether your application's processes should be created with a large Dalvik heap. This applies to all processes created for the application. It only applies to the first application loaded into a process; if you're using a shared user ID to allow multiple applications to use a process, they all must use this option consistently or they will have unpredictable results.

Most apps should not need this and should instead focus on reducing their overall memory usage for improved performance. Enabling this also does not guarantee a fixed increase in available memory, because some devices are constrained by their total available memory.

To query the available memory size at runtime, use the methods `getMemoryClass()` or `getLargeMemoryClass()`.

`android:label`

A user-readable label for the application as a whole, and a default label for each of the application's components. See the individual `label` attributes for `<activity>`, `<activity-alias>`, `<service>`, `<receiver>`, and `<provider>` elements.

The label should be set as a reference to a string resource, so that it can be localized like other strings in the user interface. However, as a convenience while you're developing the application, it can also be set as a raw string.

`android:logo`

A logo for the application as whole, and the default logo for activities.

This attribute must be set as a reference to a drawable resource containing the image (for example `"@drawable/logo"`). There is no default logo.

`android:manageSpaceActivity`

The fully qualified name of an Activity subclass that the system can launch to let users manage the memory occupied by the application on the device. The activity should also be declared with an `<activity>` element.

`android:name`

The fully qualified name of an `Application` subclass implemented for the application. When the application process is started, this class is instantiated before any of the application's components.

The subclass is optional; most applications won't need one. In the absence of a subclass, Android uses an instance of the base Application class.

`android:networkSecurityConfig`

Specifies the name of the XML file that contains your application's Network Security Configuration. The value must be a reference to the XML resource file containing the configuration.

This attribute was added in API level 24.

`android:permission`

The name of a permission that clients must have in order to interact with the application. This attribute is a convenient way to set a permission that applies to all of the application's components. It can be overwritten by setting the `permission` attributes of individual components.

For more information on permissions, see the Permissions section in the introduction and another document, Security and Permissions.

`android:persistent`

Whether or not the application should remain running at all times — `"true"` if it should, and `"false"` if not. The default value is `"false"`. Applications should not normally set this flag; persistence mode is intended only for certain system applications.

`android:process`

The name of a process where all components of the application should run. Each component can override this default by setting its own `process` attribute.

By default, Android creates a process for an application when the first of its components needs to run. All components then run in that process. The name of the default process matches the package name set by the `<manifest>` element.

By setting this attribute to a process name that's shared with another application, you can arrange for components of both applications to run in the same process — but only if the two applications also share a user ID and be signed with the same certificate.

If the name assigned to this attribute begins with a colon (':'), a new process, private to the application, is created when it's needed. If the process name begins with a lowercase character, a global process of that name is created. A global process can be shared with other applications, reducing resource usage.

`android:restoreAnyVersion`

Indicates that the application is prepared to attempt a restore of any backed-up data set, even if the backup was stored by a newer version of the application than is currently installed on the device. Setting this attribute to `true` will permit the Backup Manager to attempt restore even when a version mismatch suggests that the data are incompatible. *Use with caution!*

The default value of this attribute is `false`.

`android:requiredAccountType`

Specifies the account type required by the application in order to function. If your app requires an `Account`, the value for this attribute must correspond to the account authenticator type used by your app (as defined by `AuthenticatorDescription`), such as "com.google".

The default value is null and indicates that the application can work *without* any accounts.

Because restricted profiles currently cannot add accounts, specifying this attribute **makes your app unavailable from a restricted profile** unless you also declare `android:restrictedAccountType` with the same value.

> **Caution:** If the account data may reveal personally identifiable information, it's important that you declare this attribute and leave `android:restrictedAccountType` null, so that restricted profiles cannot use your app to access personal information that belongs to the owner user.

This attribute was added in API level 18.

`resizeableActivity`

Specifies whether the app supports multi-window display. You can set this attribute in either the `<activity>` or `<application>` element.

If you set this attribute to true, the user can launch the activity in split-screen and freeform modes. If you set the attribute to false, the activity does not support multi-window mode. If this value is false, and the user attempts to launch the activity in multi-window mode, the activity takes over the full screen.

If your app targets API level 24 or higher, but you do not specify a value for this attribute, the attribute's value defaults to true.

This attribute was added in API level 24.

`android:restrictedAccountType`

Specifies the account type required by this application and indicates that restricted profiles are allowed to access such accounts that belong to the owner user. If your app requires an `Account` and restricted profiles **are allowed to access** the primary user's accounts, the value for this attribute must correspond to the account authenticator type used by your app (as defined by `AuthenticatorDescription`), such as "com.google".

The default value is null and indicates that the application can work *without* any accounts.

> **Caution:** Specifying this attribute allows restricted profiles to use your app with accounts that belong to the owner user, which may reveal personally identifiable information. If the account may reveal personal details, you **should not** use this attribute and you should instead declare the `android:requiredAccountType` attribute to make your app unavailable to restricted profiles.

This attribute was added in API level 18.

`android:supportsRtl`

Declares whether your application is willing to support right-to-left (RTL) layouts.

If set to `true` and `targetSdkVersion` is set to 17 or higher, various RTL APIs will be activated and used by the system so your app can display RTL layouts. If set to `false` or if `targetSdkVersion` is set to 16 or lower, the RTL APIs will be ignored or will have no effect and your app will behave the same regardless of the layout direction associated to the user's Locale choice (your layouts will always be left-to-right).

The default value of this attribute is `false`.

This attribute was added in API level 17.

`android:taskAffinity`

An affinity name that applies to all activities within the application, except for those that set a different affinity with their own `taskAffinity` attributes. See that attribute for more information.

By default, all activities within an application share the same affinity. The name of that affinity is the same as the package name set by the `<manifest>` element.

`android:testOnly`

Indicates whether this application is only for testing purposes. For example, it may expose functionality or data outside of itself that would cause a security hole, but is useful for testing. This kind of APK can be installed only through `adb`—you cannot publish it to Google Play.

Android Studio automatically adds this attribute when you click **Run** ▶.

`android:theme`

A reference to a style resource defining a default theme for all activities in the application. Individual activities can override the default by setting their own `theme` attributes. For more information, see the Styles and Themes developer guide.

`android:uiOptions`

Extra options for an activity's UI.

Must be one of the following values.

Value	Description
`"none"`	No extra UI options. This is the default.
`"splitActionBarWhenNarrow"`	Add a bar at the bottom of the screen to display action items in the *app bar* (also known as the *action bar*), when constrained for horizontal space (such as when in portrait mode on a handset). Instead of a small number of action items appearing in the app bar at the top of the screen, the app bar is split into the top navigation section and the bottom bar for action items. This ensures a reasonable amount of space is made available not only for the action items, but also for navigation and title elements at the top. Menu items are not split across the two bars; they always appear together.

For more information about the app bar, see the Adding the App Bar training class.

This attribute was added in API level 14.

`android:usesCleartextTraffic`

Indicates whether the app intends to use cleartext network traffic, such as cleartext HTTP. The default value is `"true"`.

When the attribute is set to `"false"`, platform components (for example, HTTP and FTP stacks, `DownloadManager`, `MediaPlayer`) will refuse the app's requests to use cleartext traffic. Third-party libraries are strongly encouraged to honor this setting as well. The key reason for avoiding cleartext traffic is the lack of confidentiality, authenticity, and protections against tampering: a network attacker can eavesdrop on transmitted data and also modify it without being detected.

This flag is honored on a best effort basis because it's impossible to prevent all cleartext traffic from Android applications given the level of access provided to them. For example, there's no expectation that the `Socket` API will honor this flag because it cannot determine whether its traffic is in cleartext. However, most network traffic from applications is handled by higher-level network stacks/components which can honor this flag by either reading it from `ApplicationInfo.flags` or `NetworkSecurityPolicy.isCleartextTrafficPermitted()`.

NOTE: `WebView` does not honor this flag.

During app development, StrictMode can be used to identify any cleartext traffic from the app: see `StrictMode.VmPolicy.Builder.detectCleartextNetwork()`.

This attribute was added in API level 23.

This flag is ignored on Android 7.0 (API level 24) and above if an Android Network Security Config is present.

`android:vmSafeMode`

Indicates whether the app would like the virtual machine (VM) to operate in safe mode. The default value is `"false"`.

This attribute was added in API level 8 where a value of "true" disabled the Dalvik just-in-time (JIT) compiler.

This attribute was adapted in API level 22 where a value of "true" disabled the ART ahead-of-time (AOT) compiler.

INTRODUCED IN:

API Level 1

SEE ALSO:

<activity>
<service>
<receiver>
<provider>