LauncherApps | API reference | Android Developers

Archived: 2026-04-05 14:33:51 UTC

LauncherApps Stay organized with collections Save and categorize content based on your preferences.

```
open class LauncherApps
```

Class for retrieving a list of launchable activities for the current user and any associated managed profiles that are visible to the current user, which can be retrieved with `getProfiles`. This is mainly for use by launchers. Apps can be queried for each user profile. Since the PackageManager will not deliver package broadcasts for other profiles, you can register for package changes here.

To watch for managed profiles being added or removed, register for the following broadcasts: `Intent.ACTION_MANAGED_PROFILE_ADDED` and `Intent.ACTION_MANAGED_PROFILE_REMOVED`.

Note as of Android O, apps on a managed profile are no longer allowed to access apps on the main profile. Apps can only access profiles returned by `getProfiles()`.

Summary

Nested classes	
open	`ArchiveCompatibilityParams` Used to enable Archiving compatibility options with `setArchiveCompatibility`.
abstract	`Callback` Callbacks for package changes to this and related managed profiles.
	`PinItemRequest` Represents a "pin shortcut" or a "pin appwidget" request made by an app, which is sent with an `ACTION_CONFIRM_PIN_SHORTCUT` or `ACTION_CONFIRM_PIN_APPWIDGET` intent respectively to the default launcher app.
open	`ShortcutQuery` Represents a query passed to `getShortcuts(ShortcutQuery,UserHandle)`.
Constants	
static String	`ACTION_CONFIRM_PIN_APPWIDGET` Activity Action: For the default launcher to show the confirmation dialog to create a pinned app widget.

static String	`ACTION_CONFIRM_PIN_SHORTCUT` Activity Action: For the default launcher to show the confirmation dialog to create a pinned shortcut.
static String	`EXTRA_PIN_ITEM_REQUEST` An extra for `ACTION_CONFIRM_PIN_SHORTCUT` & `ACTION_CONFIRM_PIN_APPWIDGET` containing a `PinItemRequest` of appropriate type asked to pin.

Public methods

open MutableList<LauncherActivityInfo!>!	`getActivityList(packageName: String!, user: UserHandle!)` Retrieves a list of activities that specify `Intent.ACTION_MAIN` and `Intent.CATEGORY_LA` across all apps, for a specified user.
open MutableList<PackageInstaller.SessionInfo!>	`getAllPackageInstallerSessions()` Return list of all known install sessions in this user and managed profiles, regardless of th installer.
open IntentSender?	`getAppMarketActivityIntent(packageName: String?, user: UserHandle)` Returns an intent sender which can be used to start the App Market activity (Installer Act
open ApplicationInfo!	`getApplicationInfo(packageName: String, flags: Int, user: UserHandle)` Returns `ApplicationInfo` about an application installed for a specific user profile.
LauncherUserInfo?	`getLauncherUserInfo(userHandle: UserHandle)` Returns information related to a user which is useful for displaying UI elements to distin from other users (eg, badges).
open LauncherApps.PinItemRequest!	`getPinItemRequest(intent: Intent!)` A helper method to extract a `PinItemRequest` set to the `EXTRA_PIN_ITEM_REQUEST` extr
open MutableList<String!>	`getPreInstalledSystemPackages(userHandle: UserHandle)` Returns the list of the system packages that are installed at user creation.
open IntentSender?	`getPrivateSpaceSettingsIntent()` Returns `IntentSender` which can be used to start the Private Space Settings Activity.

open MutableList<UserHandle!>!	`getProfiles()` Return a list of profiles that the caller can access via the `LauncherApps` APIs.
open Drawable!	`getShortcutBadgedIconDrawable(shortcut: ShortcutInfo!, density: Int)` Returns the shortcut icon with badging appropriate for the profile.
open IntentSender?	`getShortcutConfigActivityIntent(info: LauncherActivityInfo)` Returns an intent sender which can be used to start the configure activity for creating cus shortcuts.
open MutableList<LauncherActivityInfo!>!	`getShortcutConfigActivityList(packageName: String?, user: UserHandle)` Retrieves a list of config activities for creating `ShortcutInfo` .
open Drawable!	`getShortcutIconDrawable(shortcut: ShortcutInfo, density: Int)` Returns the icon for this shortcut, without any badging for the profile.
open PendingIntent?	`getShortcutIntent(packageName: String, shortcutId: String, opts: Bundle?, user: UserHandle)` Returns PendingIntent associated with specified shortcut.
open MutableList<ShortcutInfo!>?	`getShortcuts(query: LauncherApps.ShortcutQuery, user: UserHandle)` Returns `ShortcutInfo` s that match `query` .
open Bundle?	`getSuspendedPackageLauncherExtras(packageName: String!, user: UserHandle!)` Gets the launcher extras supplied to the system when the given package was suspended v `PackageManager#setPackagesSuspended(String[], boolean, PersistableBundle, PersistableBundle, String)` .
open Boolean	`hasShortcutHostPermission()` Returns whether the caller can access the shortcut information.
open Boolean	`isActivityEnabled(component: ComponentName!, user: UserHandle!)` Checks if the activity exists and it enabled for a profile.
open Boolean	`isPackageEnabled(packageName: String!, user: UserHandle!)` Checks if the package is installed and enabled for a profile.

open Unit	`pinShortcuts(packageName: String, shortcutIds: MutableList<String!>, user: UserHandle)` Pin shortcuts on a package.
open Unit	`registerCallback(callback: LauncherApps.Callback!)` Registers a callback for changes to packages in this user and managed profiles.
open Unit	`registerCallback(callback: LauncherApps.Callback!, handler: Handler!)` Registers a callback for changes to packages in this user and managed profiles.
open Unit	`registerPackageInstallerSessionCallback(executor: Executor, callback: PackageInstaller.SessionCallback)` Register a callback to watch for session lifecycle events in this user and managed profiles
open LauncherActivityInfo!	`resolveActivity(intent: Intent!, user: UserHandle!)` Returns the activity info for a given intent and user handle, if it resolves.
open Unit	`setArchiveCompatibility(params: LauncherApps.ArchiveCompatibilityParams)` Disable different archive compatibility options of the launcher for the caller of this metho
open Boolean	`shouldHideFromSuggestions(packageName: String, user: UserHandle)` Returns whether a package should be hidden from suggestions to the user.
open Unit	`startAppDetailsActivity(component: ComponentName!, user: UserHandle!, sourceBounds: Rect!, opts: Bundle!)` Starts the settings activity to show the application details for a package in the specified pr
open Unit	`startMainActivity(component: ComponentName!, user: UserHandle!, sourceBounds: opts: Bundle!)` Starts a Main activity in the specified profile.
open Unit	`startPackageInstallerSessionDetailsActivity(sessionInfo: PackageInstaller.Sess sourceBounds: Rect?, opts: Bundle?)` Starts an activity to show the details of the specified session.
open Unit	`startShortcut(shortcut: ShortcutInfo, sourceBounds: Rect?, startActivityOptions: Bundle?)`

	Launches a shortcut.
open Unit	`startShortcut(packageName: String, shortcutId: String, sourceBounds: Rect?, startActivityOptions: Bundle?, user: UserHandle)` Starts a shortcut.
open Unit	`unregisterCallback(callback: LauncherApps.Callback!)` Unregisters a callback that was previously registered.
open Unit	`unregisterPackageInstallerSessionCallback(callback: PackageInstaller.SessionCa` Unregisters a callback that was previously registered.

Constants

ACTION_CONFIRM_PIN_SHORTCUT

```
static val ACTION_CONFIRM_PIN_SHORTCUT: String
```

Activity Action: For the default launcher to show the confirmation dialog to create a pinned shortcut.

See the `ShortcutManager` javadoc for details.

Use `getPinItemRequest(Intent)` to get a `PinItemRequest` object, and call `PinItemRequest.accept(Bundle)` if the user accepts. If the user doesn't accept, no further action is required.

```
Value:  "android.content.pm.action.CONFIRM_PIN_SHORTCUT"
```

```
static val EXTRA_PIN_ITEM_REQUEST: String
```

An extra for `ACTION_CONFIRM_PIN_SHORTCUT` & `ACTION_CONFIRM_PIN_APPWIDGET` containing a `PinItemRequest` of appropriate type asked to pin.

A helper function `getPinItemRequest(Intent)` can be used instead of using this constant directly.

```
Value:  "android.content.pm.extra.PIN_ITEM_REQUEST"
```

Public methods

getActivityList

```
open fun getActivityList(
    packageName: String!,
    user: UserHandle!
): MutableList<LauncherActivityInfo!>!
```

Retrieves a list of activities that specify `Intent.ACTION_MAIN` and `Intent.CATEGORY_LAUNCHER`, across all apps, for a specified user. If an app doesn't have any activities that specify `ACTION_MAIN` or `CATEGORY_LAUNCHER`, the system adds a synthesized activity to the list. This synthesized activity represents the app's details page within system settings.

Note: It's possible for system apps, such as app stores, to prevent the system from adding synthesized activities to the returned list.

As of Android Q, at least one of the app's activities or synthesized activities appears in the returned list unless the app satisfies at least one of the following conditions:

- The app is a system app.
- The app doesn't request any permissions.
- The app doesn't have a *launcher activity* that is enabled by default. A launcher activity has an intent containing the `ACTION_MAIN` action and the `CATEGORY_LAUNCHER` category.

Additionally, the system hides synthesized activities for some or all apps in the following enterprise-related cases:

- If the device is a fully managed device, no synthesized activities for any app appear in the returned list.
- If the current user has a work profile, no synthesized activities for the user's work apps appear in the returned list.

If the user in question is a hidden profile `UserManager.USER_TYPE_PROFILE_PRIVATE`, caller should have normal `android.Manifest.permission#ACCESS_HIDDEN_PROFILES` permission and the `android.app.role.RoleManager#ROLE_HOME` role.

Parameters	
`packageName`	String!: The specific package to query. If null, it checks all installed packages in the profile.
`user`	UserHandle!: The UserHandle of the profile.

getAppMarketActivityIntent

```
open fun getAppMarketActivityIntent(
    packageName: String?,
    user: UserHandle
): IntentSender?
```

Returns an intent sender which can be used to start the App Market activity (Installer Activity). This method is primarily used to get an intent sender which starts App Market activity for another profile, if the caller is not otherwise allowed to start activity in that profile.

When packageName is set, intent sender to start the App Market Activity which installed the package in calling user will be returned, but for the profile passed.

When packageName is not set, intent sender to launch the default App Market Activity for the profile will be returned. In case there are multiple App Market Activities available for the profile, IntentPicker will be started, allowing user to choose the preferred activity.

The method will fall back to the behaviour of not having the packageName set, in case:

- No activity for the packageName is found in calling user-space.
- The App Market Activity which installed the package in calling user-space is not present.
- The App Market Activity which installed the package in calling user-space is not present in the profile passed.

If the user in question is a hidden profile `UserManager.USER_TYPE_PROFILE_PRIVATE`, caller should have normal `android.Manifest.permission#ACCESS_HIDDEN_PROFILES` permission and the `android.app.role.RoleManager#ROLE_HOME` role.

Parameters	
`packageName`	String?: the package for which intent sender to launch App Market Activity is required. This value may be `null`.

`user`	UserHandle: the profile for which intent sender to launch App Market Activity is required. This value cannot be `null`.
Return	
`IntentSender?`	`IntentSender` object which launches the App Market Activity, null in case there is no such activity.

getShortcutBadgedIconDrawable

```
open fun getShortcutBadgedIconDrawable(
    shortcut: ShortcutInfo!,
    density: Int
): Drawable!
```

Returns the shortcut icon with badging appropriate for the profile.

The calling launcher application must be allowed to access the shortcut information, as defined in `hasShortcutHostPermission()`.

Parameters	
`density`	Int: Optional density for the icon, or 0 to use the default density. Use
Return	
`Drawable!`	A badged icon for the shortcut.
Exceptions	
`java.lang.IllegalStateException`	when the user is locked, or when the `user` user is locked or not running.

See Also

- `android.content.pm.ShortcutManager`
- `#getShortcutIconDrawable(android.content.pm.ShortcutInfo,int)`
- `android.util.DisplayMetrics`

getShortcutConfigActivityIntent

```
open fun getShortcutConfigActivityIntent(info: LauncherActivityInfo): IntentSender?
```

Returns an intent sender which can be used to start the configure activity for creating custom shortcuts. Use this method if the provider is in another profile as you are not allowed to start an activity in another profile.

The caller should receive `PinItemRequest` in onActivityResult on `android.app.Activity#RESULT_OK`.

Callers must be allowed to access the shortcut information, as defined in `hasShortcutHostPermission()`.

Exceptions	
`java.lang.IllegalStateException`	when the user is locked or not running.
`java.lang.SecurityException`	if `hasShortcutHostPermission()` is false.

See Also

- `#getPinItemRequest(android.content.Intent)`
- `android.content.Intent#ACTION_CREATE_SHORTCUT`

getShortcutIconDrawable

```
open fun getShortcutIconDrawable(
    shortcut: ShortcutInfo,
    density: Int
): Drawable!
```

Returns the icon for this shortcut, without any badging for the profile.

The calling launcher application must be allowed to access the shortcut information, as defined in `hasShortcutHostPermission()`.

Parameters	
`shortcut`	ShortcutInfo: This value cannot be `null`.
`density`	Int: The preferred density of the icon, zero for default density. Use density DPI values from `DisplayMetrics`.

Return	
`Drawable!`	The drawable associated with the shortcut.

Exceptions	
`java.lang.IllegalStateException`	when the user is locked, or when the `user` user is locked or not running.

See Also

- `android.content.pm.ShortcutManager`
- `#getShortcutBadgedIconDrawable(android.content.pm.ShortcutInfo,int)`
- `android.util.DisplayMetrics`

getShortcutIntent

```
open fun getShortcutIntent(
    packageName: String,
    shortcutId: String,
    opts: Bundle?,
    user: UserHandle
): PendingIntent?
```

Returns PendingIntent associated with specified shortcut.

Parameters	
`packageName`	String: The packageName of the shortcut. This value cannot be `null`.
`shortcutId`	String: The id of the shortcut. This value cannot be `null`.
`opts`	Bundle?: This parameter is no longer supported. This value may be `null`.
`user`	UserHandle: The UserHandle of the profile. This value cannot be `null`.

hasShortcutHostPermission

```
open fun hasShortcutHostPermission(): Boolean
```

Returns whether the caller can access the shortcut information. Access is currently available to:

- The current launcher (or default launcher if there is no set current launcher).
- The currently active voice interaction service.

Note when this method returns `false`, it may be a temporary situation because the user is trying a new launcher application. The user may decide to change the default launcher back to the calling application again, so even if a launcher application loses this permission, it does **not** have to purge pinned shortcut information. If the calling launcher application contains pinned shortcuts, they will still work, even though the caller no longer has the shortcut host permission.

Exceptions	
`java.lang.IllegalStateException`	when the user is locked.

pinShortcuts

```
open fun pinShortcuts(
    packageName: String,
    shortcutIds: MutableList<String!>,
    user: UserHandle
): Unit
```

Pin shortcuts on a package.

This API is **NOT** cumulative; this will replace all pinned shortcuts for the package. However, different launchers may have different set of pinned shortcuts.

The calling launcher application must be allowed to access the shortcut information, as defined in `hasShortcutHostPermission()`.

For user-profiles with items restricted on home screen, caller must have the required permission.

Parameters	
`packageName`	String: The target package name. This value cannot be `null`.
`shortcutIds`	MutableList<String!>: The IDs of the shortcut to be pinned. This value cannot be `null`.
`user`	UserHandle: The UserHandle of the profile. This value cannot be `null`.
Exceptions	
`java.lang.IllegalStateException`	when the user is locked, or when the `user` user is locked or not running.

startPackageInstallerSessionDetailsActivity

```
open fun startPackageInstallerSessionDetailsActivity(
    sessionInfo: PackageInstaller.SessionInfo,
    sourceBounds: Rect?,
    opts: Bundle?
): Unit
```

Starts an activity to show the details of the specified session.

Parameters	
`sessionInfo`	PackageInstaller.SessionInfo: The SessionInfo of the session. This value cannot be `null`.
`sourceBounds`	Rect?: The Rect containing the source bounds of the clicked icon. This value may be `null`.
`opts`	Bundle?: Options to pass to startActivity. This value may be `null`.

startShortcut

```
open fun startShortcut(
    shortcut: ShortcutInfo,
    sourceBounds: Rect?,
    startActivityOptions: Bundle?
): Unit
```

Launches a shortcut.

The calling launcher application must be allowed to access the shortcut information, as defined in `hasShortcutHostPermission()`.

Parameters	
`shortcut`	ShortcutInfo: The target shortcut. This value cannot be `null`.
`sourceBounds`	Rect?: The Rect containing the source bounds of the clicked icon. This value may be `null`.
`startActivityOptions`	Bundle?: Options to pass to startActivity. This value may be `null`.
Exceptions	
`android.content.ActivityNotFoundException`	failed to start shortcut. (e.g. the shortcut no longer exists, is disabled, the intent receiver activity doesn't exist, etc)
`java.lang.IllegalStateException`	when the user is locked, or when the `user` user is locked or not running.

startShortcut

```
open fun startShortcut(
    packageName: String,
    shortcutId: String,
    sourceBounds: Rect?,
    startActivityOptions: Bundle?,
    user: UserHandle
): Unit
```

Starts a shortcut.

The calling launcher application must be allowed to access the shortcut information, as defined in `hasShortcutHostPermission()`.

Parameters	
`packageName`	String: The target shortcut package name. This value cannot be `null`.
`shortcutId`	String: The target shortcut ID. This value cannot be `null`.
`sourceBounds`	Rect?: The Rect containing the source bounds of the clicked icon. This value may be `null`.
`startActivityOptions`	Bundle?: Options to pass to startActivity. This value may be `null`.
`user`	UserHandle: The UserHandle of the profile. This value cannot be `null`.
Exceptions	
`android.content.ActivityNotFoundException`	failed to start shortcut. (e.g. the shortcut no longer exists, is disabled, the intent receiver activity doesn't exist, etc)
`java.lang.IllegalStateException`	when the user is locked, or when the `user` user is locked or not running.

unregisterCallback

```
open fun unregisterCallback(callback: LauncherApps.Callback!): Unit
```

Unregisters a callback that was previously registered.
