DeviceAdminInfo | API reference | Android Developers

Archived: 2026-04-05 23:22:34 UTC

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

```
public final class DeviceAdminInfo
extends Object   implements Parcelable
```

This class is used to specify meta information of a device administrator component.

Summary

Constants	
int	HEADLESS_DEVICE_OWNER_MODE_AFFILIATED Value for getHeadlessDeviceOwnerMode() which indicates that this DPC should be provisioned into "affiliated" mode when on a Headless System User Mode device.
int	HEADLESS_DEVICE_OWNER_MODE_SINGLE_USER Value for getHeadlessDeviceOwnerMode() which indicates that this DPC should be provisioned into the first secondary user when on a Headless System User Mode device.
int	HEADLESS_DEVICE_OWNER_MODE_UNSUPPORTED Value for getHeadlessDeviceOwnerMode() which indicates that this DPC should not be provisioned into Device Owner mode on a Headless System User Mode device.
int	USES_ENCRYPTED_STORAGE A type of policy that this device admin can use: require encryption of stored data.

`int`	`USES_POLICY_DISABLE_CAMERA` A type of policy that this device admin can use: disables use of all device cameras.
`int`	`USES_POLICY_DISABLE_KEYGUARD_FEATURES` A type of policy that this device admin can use: disables use of keyguard features.
`int`	`USES_POLICY_EXPIRE_PASSWORD` A type of policy that this device admin can use: force the user to change their password after an administrator-defined time limit.
`int`	`USES_POLICY_FORCE_LOCK` A type of policy that this device admin can use: able to force the device to lock via `DevicePolicyManager.lockNow` or limit the maximum lock timeout for the device via `DevicePolicyManager.setMaximumTimeToLock` .
`int`	`USES_POLICY_LIMIT_PASSWORD` A type of policy that this device admin can use: limit the passwords that the user can select, via `DevicePolicyManager.setPasswordQuality` and `DevicePolicyManager.setPasswordMinimumLength` .
`int`	`USES_POLICY_RESET_PASSWORD` A type of policy that this device admin can use: able to reset the user's password via `DevicePolicyManager.resetPassword` .
`int`	`USES_POLICY_WATCH_LOGIN` A type of policy that this device admin can use: able to watch login attempts from the user, via `DeviceAdminReceiver.ACTION_PASSWORD_FAILED` , `DeviceAdminReceiver.ACTION_PASSWORD_SUCCEEDED` , and `DevicePolicyManager.getCurrentFailedPasswordAttempts` .
`int`	`USES_POLICY_WIPE_DATA` A type of policy that this device admin can use: able to factory reset the device, erasing all of the user's data, via `DevicePolicyManager.wipeData` .

Inherited constants

From interface `android.os.Parcelable`

`int`	`CONTENTS_FILE_DESCRIPTOR` Descriptor bit used with `describeContents()` : indicates that the Parcelable object's flattened representation includes a file descriptor.
`int`	`PARCELABLE_WRITE_RETURN_VALUE` Flag for use with `writeToParcel(Parcel, int)` : the object being written is a return value, that is the result of a function such as " `Parcelable someFunction()` ", " `void someFunction(out Parcelable)` ", or " `void someFunction(inout Parcelable)` ".

Fields

`public static final Creator<DeviceAdminInfo>`	`CREATOR` Used to make this class parcelable.

Public constructors

`DeviceAdminInfo(Context context, ResolveInfo resolveInfo)`

Constructor.

Public methods

`int`	`describeContents()` Describe the kinds of special objects contained in this Parcelable instance's marshaled representation.
`void`	`dump(Printer pw, String prefix)`
`ActivityInfo`	`getActivityInfo()` Return the raw information about the receiver implementing this device admin.

ComponentName	getComponent() Return the component of the receiver that implements this device admin.
int	getHeadlessDeviceOwnerMode() Returns the mode this DeviceAdmin wishes to use if provisioned as a Device Owner on a headless system user mode device.
String	getPackageName() Return the .apk package that implements this device admin.
String	getReceiverName() Return the class name of the receiver component that implements this device admin.
String	getTagForPolicy(int policyIdent) Return the XML tag name for the given policy identifier.
boolean	isVisible() Returns whether this device admin would like to be visible to the user, even when it is not enabled.
CharSequence	loadDescription(PackageManager pm) Load user-visible description associated with this device admin.
Drawable	loadIcon(PackageManager pm) Load the user-displayed icon for this device admin.
CharSequence	loadLabel(PackageManager pm) Load the user-displayed label for this device admin.
boolean	supportsTransferOwnership() Return true if this administrator can be a target in an ownership transfer.
String	toString()

	Returns a string representation of the object.
`boolean`	`usesPolicy(int policyIdent)` Return true if the device admin has requested that it be able to use the given policy control.
`void`	`writeToParcel(Parcel dest, int flags)` Used to package this object into a `Parcel`.

Inherited methods

From class `java.lang.Object`

`Object`	`clone()` Creates and returns a copy of this object.
`boolean`	`equals(Object obj)` Indicates whether some other object is "equal to" this one.
`void`	`finalize()` Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.
`final Class<?>`	`getClass()` Returns the runtime class of this `Object`.
`int`	`hashCode()` Returns a hash code value for the object.
`final void`	`notify()` Wakes up a single thread that is waiting on this object's monitor.

final void	notifyAll() Wakes up all threads that are waiting on this object's monitor.
String	toString() Returns a string representation of the object.
final void	wait(long timeoutMillis, int nanos) Causes the current thread to wait until it is awakened, typically by being *notified* or *interrupted*, or until a certain amount of real time has elapsed.
final void	wait(long timeoutMillis) Causes the current thread to wait until it is awakened, typically by being *notified* or *interrupted*, or until a certain amount of real time has elapsed.
final void	wait() Causes the current thread to wait until it is awakened, typically by being *notified* or *interrupted*.

From interface android.os.Parcelable

abstract int	describeContents() Describe the kinds of special objects contained in this Parcelable instance's marshaled representation.
abstract void	writeToParcel(Parcel dest, int flags) Flatten this object in to a Parcel.

Constants

HEADLESS_DEVICE_OWNER_MODE_AFFILIATED

```
public static final int HEADLESS_DEVICE_OWNER_MODE_AFFILIATED
```

Value for getHeadlessDeviceOwnerMode() which indicates that this DPC should be provisioned into "affiliated" mode when on a Headless System User Mode device.

This mode adds a Profile Owner to all users other than the user the Device Owner is on.

Starting from Android version `Build.VERSION_CODES.VANILLA_ICE_CREAM`, DPCs should set the value of attribute "headless-device-owner-mode" inside the "headless-system-user" tag as "affiliated".

Constant Value: 1 (0x00000001)

HEADLESS_DEVICE_OWNER_MODE_SINGLE_USER

```
public static final int HEADLESS_DEVICE_OWNER_MODE_SINGLE_USER
```

Value for `getHeadlessDeviceOwnerMode()` which indicates that this DPC should be provisioned into the first secondary user when on a Headless System User Mode device.

This mode only allows a single secondary user on the device blocking the creation of additional secondary users.

Starting from Android version `Build.VERSION_CODES.VANILLA_ICE_CREAM`, DPCs should set the value of attribute "headless-device-owner-mode" inside the "headless-system-user" tag as "single_user".

Constant Value: 2 (0x00000002)

HEADLESS_DEVICE_OWNER_MODE_UNSUPPORTED

```
public static final int HEADLESS_DEVICE_OWNER_MODE_UNSUPPORTED
```

Value for `getHeadlessDeviceOwnerMode()` which indicates that this DPC should not be provisioned into Device Owner mode on a Headless System User Mode device.

Constant Value: 0 (0x00000000)

USES_ENCRYPTED_STORAGE

```
public static final int USES_ENCRYPTED_STORAGE
```

A type of policy that this device admin can use: require encryption of stored data.

To control this policy, the device admin must have a "encrypted-storage" tag in the "uses-policies" section of its meta-data.

Constant Value: 7 (0x00000007)

USES_POLICY_DISABLE_CAMERA

```
public static final int USES_POLICY_DISABLE_CAMERA
```

A type of policy that this device admin can use: disables use of all device cameras.

To control this policy, the device admin must be a device owner or profile owner, and must have a "disable-camera" tag in the "uses-policies" section of its meta-data. If used by a device owner, the policy affects all users on the device.

Constant Value: 8 (0x00000008)

USES_POLICY_DISABLE_KEYGUARD_FEATURES

```
public static final int USES_POLICY_DISABLE_KEYGUARD_FEATURES
```

A type of policy that this device admin can use: disables use of keyguard features.

To control this policy, the device admin must be a device owner or profile owner, and must have a "disable-keyguard-features" tag in the "uses-policies" section of its meta-data. If used by a device owner, the policy only affects the primary user and its profiles, but not any secondary users on the device.

Constant Value: 9 (0x00000009)

USES_POLICY_EXPIRE_PASSWORD

```
public static final int USES_POLICY_EXPIRE_PASSWORD
```

A type of policy that this device admin can use: force the user to change their password after an administrator-defined time limit.

To control this policy, the device admin must be a device owner or profile owner, and must have an "expire-password" tag in the "uses-policies" section of its meta-data. If used by a device owner, the policy only affects the primary user and its profiles, but not any secondary users on the device.

Constant Value: 6 (0x00000006)

USES_POLICY_FORCE_LOCK

```
public static final int USES_POLICY_FORCE_LOCK
```

A type of policy that this device admin can use: able to force the device to lock via `DevicePolicyManager.lockNow` or limit the maximum lock timeout for the device via `DevicePolicyManager.setMaximumTimeToLock`.

To control this policy, the device admin must have a "force-lock" tag in the "uses-policies" section of its meta-data.

Constant Value: 3 (0x00000003)

USES_POLICY_LIMIT_PASSWORD

```
public static final int USES_POLICY_LIMIT_PASSWORD
```

A type of policy that this device admin can use: limit the passwords that the user can select, via `DevicePolicyManager.setPasswordQuality` and `DevicePolicyManager.setPasswordMinimumLength`.

To control this policy, the device admin must be a device owner or profile owner, and must have a "limit-password" tag in the "uses-policies" section of its meta-data. If used by a device owner, the policy only affects the primary user and its profiles, but not any secondary users on the device.

Constant Value: 0 (0x00000000)

USES_POLICY_RESET_PASSWORD

```
public static final int USES_POLICY_RESET_PASSWORD
```

A type of policy that this device admin can use: able to reset the user's password via `DevicePolicyManager.resetPassword`.

To control this policy, the device admin must have a "reset-password" tag in the "uses-policies" section of its meta-data.

Constant Value: 2 (0x00000002)

USES_POLICY_WIPE_DATA

```
public static final int USES_POLICY_WIPE_DATA
```

A type of policy that this device admin can use: able to factory reset the device, erasing all of the user's data, via `DevicePolicyManager.wipeData`.

To control this policy, the device admin must have a "wipe-data" tag in the "uses-policies" section of its meta-data.

Constant Value: 4 (0x00000004)

Fields

Public constructors

DeviceAdminInfo

```
public DeviceAdminInfo (Context context,
                ResolveInfo resolveInfo)
```

Constructor.

Parameters	
`context`	`Context` : The Context in which we are parsing the device admin.
`resolveInfo`	`ResolveInfo` : The ResolveInfo returned from the package manager about this device admin's component.

Throws	
`IOException`	
`XmlPullParserException`	

Public methods

describeContents

```
public int describeContents ()
```

Describe the kinds of special objects contained in this Parcelable instance's marshaled representation. For example, if the object will include a file descriptor in the output of `writeToParcel(Parcel,int)`, the return value of this method must include the `CONTENTS_FILE_DESCRIPTOR` bit.

Returns	
`int`	a bitmask indicating the set of special object types marshaled by this Parcelable object instance. Value is either `0` or • `CONTENTS_FILE_DESCRIPTOR`

dump

```
public void dump (Printer pw,
                String prefix)
```

Parameters	
`pw`	`Printer`

`prefix`	`String`

getActivityInfo

```
public ActivityInfo getActivityInfo ()
```

Return the raw information about the receiver implementing this device admin. Do not modify the returned object.

Returns	
`ActivityInfo`	

getComponent

```
public ComponentName getComponent ()
```

Return the component of the receiver that implements this device admin.

Returns	
`ComponentName`	This value cannot be `null`.

getHeadlessDeviceOwnerMode

```
public int getHeadlessDeviceOwnerMode ()
```

Returns the mode this DeviceAdmin wishes to use if provisioned as a Device Owner on a headless system user mode device.

Returns	
`int`	Value is one of the following: • `HEADLESS_DEVICE_OWNER_MODE_UNSUPPORTED` • `HEADLESS_DEVICE_OWNER_MODE_AFFILIATED` • `HEADLESS_DEVICE_OWNER_MODE_SINGLE_USER`

getPackageName

```
public String getPackageName ()
```

Return the .apk package that implements this device admin.

Returns	
`String`	

getReceiverName

```
public String getReceiverName ()
```

Return the class name of the receiver component that implements this device admin.

Returns	
`String`	

getTagForPolicy

```
public String getTagForPolicy (int policyIdent)
```

Return the XML tag name for the given policy identifier. Valid identifiers are as per `usesPolicy(int)` . If the given identifier is not known, null is returned.

Parameters		
`policyIdent`	`int`	
Returns		
`String`		

isVisible

```
public boolean isVisible ()
```

Returns whether this device admin would like to be visible to the user, even when it is not enabled.

Returns	
`boolean`	

loadDescription

```
public CharSequence loadDescription (PackageManager pm)
```

Load user-visible description associated with this device admin.

Parameters	
pm	PackageManager : Supply a PackageManager used to load the device admin's resources.

Returns	
CharSequence	

Throws	
Resources.NotFoundException	

loadIcon

```
public Drawable loadIcon (PackageManager pm)
```

Load the user-displayed icon for this device admin.

Parameters	
pm	PackageManager : Supply a PackageManager used to load the device admin's resources.

Returns	
Drawable	

loadLabel

```
public CharSequence loadLabel (PackageManager pm)
```

Load the user-displayed label for this device admin.

Parameters	
pm	PackageManager : Supply a PackageManager used to load the device admin's resources.

Returns	
CharSequence	

supportsTransferOwnership

```
public boolean supportsTransferOwnership ()
```

Return true if this administrator can be a target in an ownership transfer.

Returns	
`boolean`	

toString

```
public String toString ()
```

Returns a string representation of the object.

Returns	
`String`	a string representation of the object.

usesPolicy

```
public boolean usesPolicy (int policyIdent)
```

Return true if the device admin has requested that it be able to use the given policy control. The possible policy identifier inputs are: `USES_POLICY_LIMIT_PASSWORD` , `USES_POLICY_WATCH_LOGIN` , `USES_POLICY_RESET_PASSWORD` , `USES_POLICY_FORCE_LOCK` , `USES_POLICY_WIPE_DATA` , `USES_POLICY_EXPIRE_PASSWORD` , `USES_ENCRYPTED_STORAGE` , `USES_POLICY_DISABLE_CAMERA` .

Parameters	
`policyIdent`	`int`
Returns	
`boolean`	

writeToParcel

```
public void writeToParcel (Parcel dest,
                int flags)
```

Used to package this object into a `Parcel` .

Parameters	
`dest`	`Parcel` : The `Parcel` to be written.

`flags`	`int` : The flags used for parceling.

Source: https://developer.android.com/reference/android/app/admin/DeviceAdminInfo