Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

DevicePolicyManager

Kotlin (/reference/kotlin/android/app/admin/DevicePolicyManager) | **Java**

```
public class DevicePolicyManager
extends Object (/reference/java/lang/Object)
```

java.lang.Object (/reference/java/lang/Object)
↳ android.app.admin.DevicePolicyManager

Public interface for managing policies enforced on a device. Most clients of this class must be registered with the system as a device administrator (/guide/topics/admin/device-admin). Additionally, a device administrator may be registered as either a profile or device owner. A given method is accessible to all device administrators unless the documentation for that method specifies that it is restricted to either device or profile owners. Any application calling an api may only pass as an argument a device administrator component it owns. Otherwise, a `SecurityException` (/reference/java/lang/SecurityException) will be thrown.

Note: on `automotive builds` (/reference/android/content/pm/PackageManager#FEATURE_AUTOMOTIVE), some methods can throw an `UnsafeStateException` (/reference/android/app/admin/UnsafeStateException) exception (for example, if the vehicle is moving), so callers running on automotive builds should always check for that exception, otherwise they might crash.

Developer Guides

For more information about managing policies for device administration, read the Device Administration (/guide/topics/admin/device-admin) developer guide.

Requires the `PackageManager#FEATURE_DEVICE_ADMIN` (/reference/android/content/pm/PackageManager#FEATURE_DEVICE_ADMIN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Summary

Nested classes

`class`	`DevicePolicyManager.InstallSystemUpdateCallback` (/reference/ Callback used in `DevicePolicyManager.installSystemUpdate(Comp` (/reference/android/app/admin/DevicePolicyManager#installSystemUpdate that there was an error while trying to install an update.
`interface`	`DevicePolicyManager.OnClearApplicationUserDataListener` (/re Callback used in `DevicePolicyManager.clearApplicationUserData` (/reference/android/app/admin/DevicePolicyManager#clearApplicationUse to indicate that the clearing of an application's user data is done.

Constants

`String` (/reference/java/lang/String)	`ACTION_ADD_DEVICE_ADMIN` (/reference/android/app/admin/DevicePolicy Activity action: ask the user to add a new device administrator to the syste
`String` (/reference/java/lang/String)	`ACTION_ADMIN_POLICY_COMPLIANCE` (/reference/android/app/admin/Dev Activity action: Starts the administrator to show policy compliance for the p
`String` (/reference/java/lang/String)	`ACTION_APPLICATION_DELEGATION_SCOPES_CHANGED` (/reference/andı Broadcast Action: Sent after application delegation scopes are changed.

String (/reference/java/lang/String)	ACTION_CHECK_POLICY_COMPLIANCE (/reference/android/app/admin/Dev Activity action: launch the DPC to check policy compliance.
String (/reference/java/lang/String)	ACTION_DEVICE_ADMIN_SERVICE (/reference/android/app/admin/DeviceI Service action: Action for a service that device owner and profile owner car
String (/reference/java/lang/String)	ACTION_DEVICE_FINANCING_STATE_CHANGED (/reference/android/app/a Broadcast Action: Broadcast sent to indicate that the device financing state
String (/reference/java/lang/String)	ACTION_DEVICE_OWNER_CHANGED (/reference/android/app/admin/DeviceI Broadcast action: sent when the device owner is set, changed or cleared.
String (/reference/java/lang/String)	ACTION_DEVICE_POLICY_RESOURCE_UPDATED (/reference/android/app/a Broadcast action: notify system apps (e.g. settings, SysUI, etc) that the dev has been updated, the updated resources can be retrieved using DeviceP (/reference/android/app/admin/DevicePolicyResourcesManager#getDrawa DevicePolicyResourcesManager#getString (/reference/android/app
String (/reference/java/lang/String)	ACTION_GET_PROVISIONING_MODE (/reference/android/app/admin/Device Activity action: Starts the administrator to get the mode for the provisioning
String (/reference/java/lang/String)	ACTION_MANAGED_PROFILE_PROVISIONED (/reference/android/app/admi Broadcast Action: This broadcast is sent to indicate that provisioning of a m
String (/reference/java/lang/String)	ACTION_PROFILE_OWNER_CHANGED (/reference/android/app/admin/Device Broadcast action: sent when the profile owner is set, changed or cleared.
String (/reference/java/lang/String)	ACTION_PROVISIONING_SUCCESSFUL (/reference/android/app/admin/Dev Activity action: This activity action is sent to indicate that provisioning of a r
String (/reference/java/lang/String)	ACTION_PROVISION_MANAGED_DEVICE (/reference/android/app/admin/De

	This constant was deprecated in API level 31. to support Build.VERSION_ ACTION_GET_PROVISIONING_MODE (/reference/android/app/admin/Devic (/reference/android/app/admin/DevicePolicyManager#ACTION_ADMIN_PO (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION (/reference/android/os/Build.VERSION_CODES#S)*, admin apps must also c*
String (/reference/java/lang/String)	ACTION_PROVISION_MANAGED_PROFILE (/reference/android/app/admin/D Activity action: Starts the provisioning flow which sets up a managed profil
String (/reference/java/lang/String)	ACTION_SET_NEW_PARENT_PROFILE_PASSWORD (/reference/android/app Activity action: have the user enter a new password for the parent profile.
String (/reference/java/lang/String)	ACTION_SET_NEW_PASSWORD (/reference/android/app/admin/DevicePolicy Activity action: have the user enter a new password.
String (/reference/java/lang/String)	ACTION_START_ENCRYPTION (/reference/android/app/admin/DevicePolicy Activity action: begin the process of encrypting data on the device.
String (/reference/java/lang/String)	ACTION_SYSTEM_UPDATE_POLICY_CHANGED (/reference/android/app/adn Broadcast action: notify that a new local system update policy has been se
int	CONTENT_PROTECTION_DISABLED (/reference/android/app/admin/DeviceI Indicates that content protection is controlled and disabled by a policy (def
int	CONTENT_PROTECTION_ENABLED (/reference/android/app/admin/DevicePo Indicates that content protection is controlled and enabled by a policy.
int	CONTENT_PROTECTION_NOT_CONTROLLED_BY_POLICY (/reference/andro Indicates that content protection is not controlled by policy, allowing user t
String (/reference/java/lang/String)	DELEGATION_APP_RESTRICTIONS (/reference/android/app/admin/DeviceI Delegation of application restrictions management.

String (/reference/java/lang/String)	DELEGATION_BLOCK_UNINSTALL (/reference/android/app/admin/DevicePo Delegation of application uninstall block.
String (/reference/java/lang/String)	DELEGATION_CERT_INSTALL (/reference/android/app/admin/DevicePolicy Delegation of certificate installation and management.
String (/reference/java/lang/String)	DELEGATION_CERT_SELECTION (/reference/android/app/admin/DevicePol Grants access to selection of KeyChain certificates on behalf of requesting
String (/reference/java/lang/String)	DELEGATION_ENABLE_SYSTEM_APP (/reference/android/app/admin/Devic Delegation for enabling system apps.
String (/reference/java/lang/String)	DELEGATION_INSTALL_EXISTING_PACKAGE (/reference/android/app/adn Delegation for installing existing packages.
String (/reference/java/lang/String)	DELEGATION_KEEP_UNINSTALLED_PACKAGES (/reference/android/app/ac Delegation of management of uninstalled packages.
String (/reference/java/lang/String)	DELEGATION_NETWORK_LOGGING (/reference/android/app/admin/DevicePo Grants access to setNetworkLoggingEnabled(ComponentName, boo isNetworkLoggingEnabled(ComponentName) (/reference/android/app retrieveNetworkLogs(ComponentName, long) (/reference/android/a
String (/reference/java/lang/String)	DELEGATION_PACKAGE_ACCESS (/reference/android/app/admin/DevicePol Delegation of package access state.
String (/reference/java/lang/String)	DELEGATION_PERMISSION_GRANT (/reference/android/app/admin/DeviceP Delegation of permission policy and permission grant state.
String (/reference/java/lang/String)	DELEGATION_SECURITY_LOGGING (/reference/android/app/admin/DeviceP Grants access to setSecurityLoggingEnabled(ComponentName, boo isSecurityLoggingEnabled(ComponentName) (/reference/android/ap

	retrieveSecurityLogs(ComponentName) (/reference/android/app/adn retrievePreRebootSecurityLogs(ComponentName) (/reference/andı
int	ENCRYPTION_STATUS_ACTIVATING (/reference/android/app/admin/Device *This constant was deprecated in API level 34. This result code has never ac*
int	ENCRYPTION_STATUS_ACTIVE (/reference/android/app/admin/DevicePolic Result code for setStorageEncryption(ComponentName, boolean) getStorageEncryptionStatus() (/reference/android/app/admin/Device
int	ENCRYPTION_STATUS_ACTIVE_DEFAULT_KEY (/reference/android/app/ac Result code for getStorageEncryptionStatus() (/reference/android/a cryptographically protected by the user's credentials.
int	ENCRYPTION_STATUS_ACTIVE_PER_USER (/reference/android/app/admin Result code for getStorageEncryptionStatus() (/reference/android/a user or profile.
int	ENCRYPTION_STATUS_INACTIVE (/reference/android/app/admin/DevicePo Result code for setStorageEncryption(ComponentName, boolean) getStorageEncryptionStatus() (/reference/android/app/admin/Device
int	ENCRYPTION_STATUS_UNSUPPORTED (/reference/android/app/admin/Devi Result code for setStorageEncryption(ComponentName, boolean) getStorageEncryptionStatus() (/reference/android/app/admin/Device
String (/reference/java/lang/String)	EXTRA_ADD_EXPLANATION (/reference/android/app/admin/DevicePolicyMa An optional CharSequence providing additional explanation for why the adr
String (/reference/java/lang/String)	EXTRA_DELEGATION_SCOPES (/reference/android/app/admin/DevicePolicy An `ArrayList<String>` corresponding to the delegation scopes given to (/reference/android/app/admin/DevicePolicyManager#ACTION_APPLICATI

String (/reference/java/lang/String)	EXTRA_DEVICE_ADMIN (/reference/android/app/admin/DevicePolicyManag The ComponentName of the administrator component.
String (/reference/java/lang/String)	EXTRA_DEVICE_PASSWORD_REQUIREMENT_ONLY (/reference/android/app A boolean extra for ACTION_SET_NEW_PARENT_PROFILE_PASSWORD (/ref requirement is enforced during the parent profile password enrolment flow.
String (/reference/java/lang/String)	EXTRA_PASSWORD_COMPLEXITY (/reference/android/app/admin/DevicePol An integer indicating the complexity level of the new password an app wou (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_
String (/reference/java/lang/String)	EXTRA_PROVISIONING_ACCOUNT_TO_MIGRATE (/reference/android/app/a An Account (/reference/android/accounts/Account) extra holding the acco
String (/reference/java/lang/String)	EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE (/reference/android/app A Parcelable (/reference/android/os/Parcelable) extra of type Persista which starts managed provisioning to pass data to the management applic
String (/reference/java/lang/String)	EXTRA_PROVISIONING_ALLOWED_PROVISIONING_MODES (/reference/an An ArrayList (/reference/java/util/ArrayList) of Integer (/reference/java/
String (/reference/java/lang/String)	EXTRA_PROVISIONING_ALLOW_OFFLINE (/reference/android/app/admin/D A boolean extra indicating whether offline provisioning is allowed.
String (/reference/java/lang/String)	EXTRA_PROVISIONING_DEVICE_ADMIN_COMPONENT_NAME (/reference/a A ComponentName extra indicating the device admin receiver of the mobil
String (/reference/java/lang/String)	EXTRA_PROVISIONING_DEVICE_ADMIN_MINIMUM_VERSION_CODE (/refe An int extra holding a minimum required version code for the device admin
String (/reference/java/lang/String)	EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_CHECKSUM (/reference

	A String extra holding the URL-safe base64 encoded SHA-256 hash of the (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONII
String (/reference/java/lang/String)	EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_COOKIE_H (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONII A String extra holding a http cookie header which should be used in the htt (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONII
String (/reference/java/lang/String)	EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_LOCATION A String extra holding a url that specifies the download location of the devi
String (/reference/java/lang/String)	EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_NAME (/reference/andı *This constant was deprecated in API level 23. Use* EXTRA_PROVISIONING_ (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONII *requires the permission* Manifest.permission.BIND_DEVICE_ADMIN (/
String (/reference/java/lang/String)	EXTRA_PROVISIONING_DEVICE_ADMIN_SIGNATURE_CHECKSUM (/referer A String extra holding the URL-safe base64 encoded SHA-256 checksum o EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_LOCATION
String (/reference/java/lang/String)	EXTRA_PROVISIONING_DISCLAIMERS (/reference/android/app/admin/Dev A Bundle (/reference/android/os/Bundle)[] extra consisting of list of disclai
String (/reference/java/lang/String)	EXTRA_PROVISIONING_DISCLAIMER_CONTENT (/reference/android/app/a A Uri (/reference/android/net/Uri) extra pointing to disclaimer content.
String (/reference/java/lang/String)	EXTRA_PROVISIONING_DISCLAIMER_HEADER (/reference/android/app/ac A String extra of localized disclaimer header.
String (/reference/java/lang/String)	EXTRA_PROVISIONING_EMAIL_ADDRESS (/reference/android/app/admin/D *This constant was deprecated in API level 26. From* Build.VERSION_CODE
String (/reference/java/lang/String)	EXTRA_PROVISIONING_IMEI (/reference/android/app/admin/DevicePolicy

	A string extra holding the IMEI (International Mobile Equipment Identity) of
String (/reference/java/lang/String)	EXTRA_PROVISIONING_KEEP_ACCOUNT_ON_MIGRATION (/reference/andı Boolean extra to indicate that the migrated account should be kept.
String (/reference/java/lang/String)	EXTRA_PROVISIONING_KEEP_SCREEN_ON (/reference/android/app/admin *This constant was deprecated in API level 34. from* Build.VERSION_CODE. *kept on throughout the provisioning flow.*
String (/reference/java/lang/String)	EXTRA_PROVISIONING_LEAVE_ALL_SYSTEM_APPS_ENABLED (/reference A Boolean extra that can be used by the mobile device management applic
String (/reference/java/lang/String)	EXTRA_PROVISIONING_LOCALE (/reference/android/app/admin/DevicePol A String extra holding the Locale (/reference/java/util/Locale) that the dev
String (/reference/java/lang/String)	EXTRA_PROVISIONING_LOCAL_TIME (/reference/android/app/admin/Devi A Long extra holding the wall clock time (in milliseconds) to be set on the d
String (/reference/java/lang/String)	EXTRA_PROVISIONING_LOGO_URI (/reference/android/app/admin/DeviceI *This constant was deprecated in API level 33. Logo customization is no long*
String (/reference/java/lang/String)	EXTRA_PROVISIONING_MAIN_COLOR (/reference/android/app/admin/Devi *This constant was deprecated in API level 31. Color customization is no long*
String (/reference/java/lang/String)	EXTRA_PROVISIONING_MODE (/reference/android/app/admin/DevicePolicy An intent extra holding the provisioning mode returned by the administrato
String (/reference/java/lang/String)	EXTRA_PROVISIONING_SENSORS_PERMISSION_GRANT_OPT_OUT (/refere A boolean extra indicating the admin of a fully-managed device opts out of java.lang.String, java.lang.String, int) (/reference/android/app/admin/DevicePolicyManager#setPermissionGrantS

String (/reference/java/lang/String)	EXTRA_PROVISIONING_SERIAL_NUMBER (/reference/android/app/admin/D A string extra holding the serial number of the device.
String (/reference/java/lang/String)	EXTRA_PROVISIONING_SHOULD_LAUNCH_RESULT_INTENT (/reference/a A boolean extra that determines whether the provisioning flow should laun (/reference/android/app/admin/DevicePolicyManager#EXTRA_RESULT_LAU
String (/reference/java/lang/String)	EXTRA_PROVISIONING_SKIP_EDUCATION_SCREENS (/reference/android/ A boolean extra indicating if the education screens from the provisioning fl
String (/reference/java/lang/String)	EXTRA_PROVISIONING_SKIP_ENCRYPTION (/reference/android/app/admi A boolean extra indicating whether device encryption can be skipped as pa
String (/reference/java/lang/String)	EXTRA_PROVISIONING_SKIP_USER_CONSENT (/reference/android/app/ac *This constant was deprecated in API level 31. this extra is no longer relevant*
String (/reference/java/lang/String)	EXTRA_PROVISIONING_TIME_ZONE (/reference/android/app/admin/Device A String extra holding the time zone AlarmManager (/reference/android/ap
String (/reference/java/lang/String)	EXTRA_PROVISIONING_USE_MOBILE_DATA (/reference/android/app/admi A boolean extra indicating if mobile data should be used during the provisi
String (/reference/java/lang/String)	EXTRA_PROVISIONING_WIFI_ANONYMOUS_IDENTITY (/reference/androi The anonymous identity of the wifi network in EXTRA_PROVISIONING_WIF
String (/reference/java/lang/String)	EXTRA_PROVISIONING_WIFI_CA_CERTIFICATE (/reference/android/app The CA certificate of the wifi network in EXTRA_PROVISIONING_WIFI_SS
String (/reference/java/lang/String)	EXTRA_PROVISIONING_WIFI_DOMAIN (/reference/android/app/admin/Dev The domain of the wifi network in EXTRA_PROVISIONING_WIFI_SSID (/re

String (/reference/java/lang/String)	EXTRA_PROVISIONING_WIFI_EAP_METHOD (/reference/android/app/admi The EAP method of the wifi network in EXTRA_PROVISIONING_WIFI_SSI AKA or AKA_PRIME.
String (/reference/java/lang/String)	EXTRA_PROVISIONING_WIFI_HIDDEN (/reference/android/app/admin/Dev A boolean extra indicating whether the wifi network in EXTRA_PROVISION
String (/reference/java/lang/String)	EXTRA_PROVISIONING_WIFI_IDENTITY (/reference/android/app/admin/D The identity of the wifi network in EXTRA_PROVISIONING_WIFI_SSID (/re
String (/reference/java/lang/String)	EXTRA_PROVISIONING_WIFI_PAC_URL (/reference/android/app/admin/De A String extra holding the proxy auto-config (PAC) URL for the wifi network
String (/reference/java/lang/String)	EXTRA_PROVISIONING_WIFI_PASSWORD (/reference/android/app/admin/D A String extra holding the password of the wifi network in EXTRA_PROVISI
String (/reference/java/lang/String)	EXTRA_PROVISIONING_WIFI_PHASE2_AUTH (/reference/android/app/adm The phase 2 authentication of the wifi network in EXTRA_PROVISIONING_ MSCHAP, MSCHAPV2, GTC, SIM, AKA or AKA_PRIME.
String (/reference/java/lang/String)	EXTRA_PROVISIONING_WIFI_PROXY_BYPASS (/reference/android/app/ac A String extra holding the proxy bypass for the wifi network in EXTRA_PROV
String (/reference/java/lang/String)	EXTRA_PROVISIONING_WIFI_PROXY_HOST (/reference/android/app/admi A String extra holding the proxy host for the wifi network in EXTRA_PROVIS
String (/reference/java/lang/String)	EXTRA_PROVISIONING_WIFI_PROXY_PORT (/reference/android/app/admi An int extra holding the proxy port for the wifi network in EXTRA_PROVISI
String (/reference/java/lang/String)	EXTRA_PROVISIONING_WIFI_SECURITY_TYPE (/reference/android/app/a

	A String extra indicating the security type of the wifi network in EXTRA_PRO NONE, WPA, WEP or EAP.
String (/reference/java/lang/String)	EXTRA_PROVISIONING_WIFI_SSID (/reference/android/app/admin/Device A String extra holding the ssid of the wifi network that should be used durin
String (/reference/java/lang/String)	EXTRA_PROVISIONING_WIFI_USER_CERTIFICATE (/reference/android/a The user certificate of the wifi network in EXTRA_PROVISIONING_WIFI_S
String (/reference/java/lang/String)	EXTRA_RESOURCE_IDS (/reference/android/app/admin/DevicePolicyManag An integer array extra for ACTION_DEVICE_POLICY_RESOURCE_UPDATED
String (/reference/java/lang/String)	EXTRA_RESOURCE_TYPE (/reference/android/app/admin/DevicePolicyMana An int extra for ACTION_DEVICE_POLICY_RESOURCE_UPDATED (/referer updated, the type can be EXTRA_RESOURCE_TYPE_DRAWABLE (/reference (/reference/android/app/admin/DevicePolicyManager#EXTRA_RESOURCE_
int	EXTRA_RESOURCE_TYPE_DRAWABLE (/reference/android/app/admin/Device A int value for EXTRA_RESOURCE_TYPE (/reference/android/app/admin/De (/reference/android/graphics/drawable/Drawable) is being updated.
int	EXTRA_RESOURCE_TYPE_STRING (/reference/android/app/admin/DevicePo A int value for EXTRA_RESOURCE_TYPE (/reference/android/app/admin/De
String (/reference/java/lang/String)	EXTRA_RESULT_LAUNCH_INTENT (/reference/android/app/admin/DevicePo An Intent (/reference/android/content/Intent) result extra specifying the I
int	FLAG_EVICT_CREDENTIAL_ENCRYPTION_KEY (/reference/android/app/ac Flag for lockNow(int) (/reference/android/app/admin/DevicePolicyManag
int	FLAG_MANAGED_CAN_ACCESS_PARENT (/reference/android/app/admin/Dev

	Flag used by **addCrossProfileIntentFilter(ComponentName, Inte** (/reference/android/app/admin/DevicePolicyManager#addCrossProfileInte intents sent from the parent profile.
int	FLAG_PARENT_CAN_ACCESS_MANAGED (/reference/android/app/admin/Dev Flag used by **addCrossProfileIntentFilter(ComponentName, Inte** (/reference/android/app/admin/DevicePolicyManager#addCrossProfileInte sent from the managed profile.
int	ID_TYPE_BASE_INFO (/reference/android/app/admin/DevicePolicyManage Specifies that the device should attest its manufacturer details.
int	ID_TYPE_IMEI (/reference/android/app/admin/DevicePolicyManager#ID_T Specifies that the device should attest its IMEI.
int	ID_TYPE_INDIVIDUAL_ATTESTATION (/reference/android/app/admin/Dev Specifies that the device should attest using an individual attestation certif
int	ID_TYPE_MEID (/reference/android/app/admin/DevicePolicyManager#ID_T Specifies that the device should attest its MEID.
int	ID_TYPE_SERIAL (/reference/android/app/admin/DevicePolicyManager#ID Specifies that the device should attest its serial number.
int	INSTALLKEY_REQUEST_CREDENTIALS_ACCESS (/reference/android/app/a Specifies that the calling app should be granted access to the installed cre
int	INSTALLKEY_SET_USER_SELECTABLE (/reference/android/app/admin/Dev Specifies that a user can select the key via the Certificate Selection promp
int	KEYGUARD_DISABLE_BIOMETRICS (/reference/android/app/admin/DeviceP Disable all biometric authentication on keyguard secure screens (e.g. PIN/P

int	KEYGUARD_DISABLE_FACE (/reference/android/app/admin/DevicePolicyMa Disable face authentication on keyguard secure screens (e.g. PIN/Pattern/F
int	KEYGUARD_DISABLE_FEATURES_ALL (/reference/android/app/admin/Devi Disable all current and future keyguard customizations.
int	KEYGUARD_DISABLE_FEATURES_NONE (/reference/android/app/admin/Dev Widgets are enabled in keyguard
int	KEYGUARD_DISABLE_FINGERPRINT (/reference/android/app/admin/Devic Disable fingerprint authentication on keyguard secure screens (e.g. PIN/Pat
int	KEYGUARD_DISABLE_IRIS (/reference/android/app/admin/DevicePolicyMa Disable iris authentication on keyguard secure screens (e.g. PIN/Pattern/Pa
int	KEYGUARD_DISABLE_REMOTE_INPUT (/reference/android/app/admin/Devi *This constant was deprecated in API level 33. This flag was added in versior*
int	KEYGUARD_DISABLE_SECURE_CAMERA (/reference/android/app/admin/Dev Disable the camera on secure keyguard screens (e.g. PIN/Pattern/Passwor
int	KEYGUARD_DISABLE_SECURE_NOTIFICATIONS (/reference/android/app/a Disable showing all notifications on secure keyguard screens (e.g. PIN/Patt
int	KEYGUARD_DISABLE_SHORTCUTS_ALL (/reference/android/app/admin/Dev Disable all keyguard shortcuts.
int	KEYGUARD_DISABLE_TRUST_AGENTS (/reference/android/app/admin/Devi Disable trust agents on secure keyguard screens (e.g. PIN/Pattern/Passwor
int	KEYGUARD_DISABLE_UNREDACTED_NOTIFICATIONS (/reference/android/

	Only allow redacted notifications on secure keyguard screens (e.g. PIN/Pat
int	KEYGUARD_DISABLE_WIDGETS_ALL (/reference/android/app/admin/Devic Disable all keyguard widgets.
int	LEAVE_ALL_SYSTEM_APPS_ENABLED (/reference/android/app/admin/Devi Flag used by createAndManageUser(ComponentName, String, Comp (/reference/android/app/admin/DevicePolicyManager#createAndManageU to specify that the newly created user should skip the disabling of system a
int	LOCK_TASK_FEATURE_BLOCK_ACTIVITY_START_IN_TASK (/reference/a Enable blocking of non-allowlisted activities from being started into a locke
int	LOCK_TASK_FEATURE_GLOBAL_ACTIONS (/reference/android/app/admin/D Enable the global actions dialog during LockTask mode.
int	LOCK_TASK_FEATURE_HOME (/reference/android/app/admin/DevicePolicyM Enable the Home button during LockTask mode.
int	LOCK_TASK_FEATURE_KEYGUARD (/reference/android/app/admin/DevicePo Enable the keyguard during LockTask mode.
int	LOCK_TASK_FEATURE_NONE (/reference/android/app/admin/DevicePolicyM Disable all configurable SystemUI features during LockTask mode.
int	LOCK_TASK_FEATURE_NOTIFICATIONS (/reference/android/app/admin/De Enable notifications during LockTask mode.
int	LOCK_TASK_FEATURE_OVERVIEW (/reference/android/app/admin/DevicePo Enable the Overview button and the Overview screen during LockTask mod
int	LOCK_TASK_FEATURE_SYSTEM_INFO (/reference/android/app/admin/Devi

	Enable the system info area in the status bar during LockTask mode.
`int`	`MAKE_USER_EPHEMERAL` (/reference/android/app/admin/DevicePolicyMana Flag used by `createAndManageUser(ComponentName, String, Comp` (/reference/android/app/admin/DevicePolicyManager#createAndManageU to specify that the user should be created ephemeral.
`String` (/reference/java/lang/String)	`MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePo This MIME type is used for starting the device owner provisioning.
`int`	`MTE_DISABLED` (/reference/android/app/admin/DevicePolicyManager#MTE Require that MTE be disabled on the device.
`int`	`MTE_ENABLED` (/reference/android/app/admin/DevicePolicyManager#MTE_ Require that MTE be enabled on the device, if supported.
`int`	`MTE_NOT_CONTROLLED_BY_POLICY` (/reference/android/app/admin/Devic Allow the user to choose whether to enable MTE on the device.
`int`	`NEARBY_STREAMING_DISABLED` (/reference/android/app/admin/DevicePol Indicates that nearby streaming is disabled.
`int`	`NEARBY_STREAMING_ENABLED` (/reference/android/app/admin/DevicePolic Indicates that nearby streaming is enabled.
`int`	`NEARBY_STREAMING_NOT_CONTROLLED_BY_POLICY` (/reference/android/ Indicates that nearby streaming is not controlled by policy, which means ne
`int`	`NEARBY_STREAMING_SAME_MANAGED_ACCOUNT_ONLY` (/reference/androi Indicates that nearby streaming is enabled only to devices offering a comp
`int`	`OPERATION_SAFETY_REASON_DRIVING_DISTRACTION` (/reference/andro

	Indicates that a `UnsafeStateException` (/reference/android/app/admin/U
`int`	`PASSWORD_COMPLEXITY_HIGH` (/reference/android/app/admin/DevicePolic Constant for `getPasswordComplexity()` (/reference/android/app/admin/ (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswo
`int`	`PASSWORD_COMPLEXITY_LOW` (/reference/android/app/admin/DevicePolicy Constant for `getPasswordComplexity()` (/reference/android/app/admin/ (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswo
`int`	`PASSWORD_COMPLEXITY_MEDIUM` (/reference/android/app/admin/DevicePo Constant for `getPasswordComplexity()` (/reference/android/app/admin/ (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswo
`int`	`PASSWORD_COMPLEXITY_NONE` (/reference/android/app/admin/DevicePolic Constant for `getPasswordComplexity()` (/reference/android/app/admin/ (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswo
`int`	`PASSWORD_QUALITY_ALPHABETIC` (/reference/android/app/admin/DeviceP Constant for `setPasswordQuality(ComponentName, int)` (/reference password containing at least alphabetic (or other symbol) characters.
`int`	`PASSWORD_QUALITY_ALPHANUMERIC` (/reference/android/app/admin/Devi Constant for `setPasswordQuality(ComponentName, int)` (/reference password containing at least *both>* numeric *and* alphabetic (or other symb
`int`	`PASSWORD_QUALITY_BIOMETRIC_WEAK` (/reference/android/app/admin/De Constant for `setPasswordQuality(ComponentName, int)` (/reference security biometric recognition technology.
`int`	`PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolic Constant for `setPasswordQuality(ComponentName, int)` (/reference precisely how many characters of various types the password should conta

int	PASSWORD_QUALITY_NUMERIC (/reference/android/app/admin/DevicePolic Constant for setPasswordQuality(ComponentName, int) (/reference password containing at least numeric characters.
int	PASSWORD_QUALITY_NUMERIC_COMPLEX (/reference/android/app/admin/D Constant for setPasswordQuality(ComponentName, int) (/reference password containing at least numeric characters with no repeating (4444)
int	PASSWORD_QUALITY_SOMETHING (/reference/android/app/admin/DevicePo Constant for setPasswordQuality(ComponentName, int) (/reference password or pattern, but doesn't care what it is.
int	PASSWORD_QUALITY_UNSPECIFIED (/reference/android/app/admin/Device Constant for setPasswordQuality(ComponentName, int) (/reference for the password.
int	PERMISSION_GRANT_STATE_DEFAULT (/reference/android/app/admin/Dev Runtime permission state: The user can manage the permission through the
int	PERMISSION_GRANT_STATE_DENIED (/reference/android/app/admin/Devi Runtime permission state: The permission is denied to the app and the user
int	PERMISSION_GRANT_STATE_GRANTED (/reference/android/app/admin/Dev Runtime permission state: The permission is granted to the app and the use
int	PERMISSION_POLICY_AUTO_DENY (/reference/android/app/admin/DeviceP Permission policy to always deny new permission requests for runtime perm
int	PERMISSION_POLICY_AUTO_GRANT (/reference/android/app/admin/Device Permission policy to always grant new permission requests for runtime perm
int	PERMISSION_POLICY_PROMPT (/reference/android/app/admin/DevicePolic

	Permission policy to prompt user for new permission requests for runtime p
`int`	PERSONAL_APPS_NOT_SUSPENDED (/reference/android/app/admin/DeviceI Return value for getPersonalAppsSuspendedReasons(ComponentNam personal apps are not suspended.
`int`	PERSONAL_APPS_SUSPENDED_EXPLICITLY (/reference/android/app/admi Flag for getPersonalAppsSuspendedReasons(ComponentName) (/refe
`int`	PERSONAL_APPS_SUSPENDED_PROFILE_TIMEOUT (/reference/android/ap Flag for getPersonalAppsSuspendedReasons(ComponentName) (/refe
String (/reference/java/lang/String)	POLICY_DISABLE_CAMERA (/reference/android/app/admin/DevicePolicyMa Constant to indicate the feature of disabling the camera.
String (/reference/java/lang/String)	POLICY_DISABLE_SCREEN_CAPTURE (/reference/android/app/admin/Devi Constant to indicate the feature of disabling screen captures.
`int`	PRIVATE_DNS_MODE_OFF (/reference/android/app/admin/DevicePolicyMar Specifies that Private DNS was turned off completely.
`int`	PRIVATE_DNS_MODE_OPPORTUNISTIC (/reference/android/app/admin/Dev Specifies that the device owner requested opportunistic DNS over TLS
`int`	PRIVATE_DNS_MODE_PROVIDER_HOSTNAME (/reference/android/app/admi Specifies that the device owner configured a specific host to use for Privat
`int`	PRIVATE_DNS_MODE_UNKNOWN (/reference/android/app/admin/DevicePolic Specifies that the Private DNS setting is in an unknown state.
`int`	PRIVATE_DNS_SET_ERROR_FAILURE_SETTING (/reference/android/app/a

	General failure to set the Private DNS mode, not due to one of the reasons
int	PRIVATE_DNS_SET_ERROR_HOST_NOT_SERVING (/reference/android/app If the `privateDnsHost` provided was of a valid hostname but that host wa
int	PRIVATE_DNS_SET_NO_ERROR (/reference/android/app/admin/DevicePolic The selected mode has been set successfully.
int	PROVISIONING_MODE_FULLY_MANAGED_DEVICE (/reference/android/app The provisioning mode for fully managed device.
int	PROVISIONING_MODE_MANAGED_PROFILE (/reference/android/app/admin The provisioning mode for managed profile.
int	PROVISIONING_MODE_MANAGED_PROFILE_ON_PERSONAL_DEVICE (/refe The provisioning mode for a managed profile on a personal device.
int	RESET_PASSWORD_DO_NOT_ASK_CREDENTIALS_ON_BOOT (/reference/an Flag for resetPasswordWithToken(ComponentName, String, byte (/reference/android/app/admin/DevicePolicyManager#resetPasswordWithT (/reference/android/app/admin/DevicePolicyManager#resetPassword(java.
int	RESET_PASSWORD_REQUIRE_ENTRY (/reference/android/app/admin/Devic Flag for resetPasswordWithToken(ComponentName, String, byte (/reference/android/app/admin/DevicePolicyManager#resetPasswordWithT (/reference/android/app/admin/DevicePolicyManager#resetPassword(java.
int	SKIP_SETUP_WIZARD (/reference/android/app/admin/DevicePolicyManage Flag used by createAndManageUser(ComponentName, String, Comp (/reference/android/app/admin/DevicePolicyManager#createAndManageU to skip setup wizard after creating a new user.
int	WIFI_SECURITY_ENTERPRISE_192 (/reference/android/app/admin/Devic

	Constant for `getMinimumRequiredWifiSecurityLevel()` (/reference/a `setMinimumRequiredWifiSecurityLevel(int)` (/reference/android/a
`int`	`WIFI_SECURITY_ENTERPRISE_EAP` (/reference/android/app/admin/Devic Constant for `getMinimumRequiredWifiSecurityLevel()` (/reference/a `setMinimumRequiredWifiSecurityLevel(int)` (/reference/android/a
`int`	`WIFI_SECURITY_OPEN` (/reference/android/app/admin/DevicePolicyManag Constant for `getMinimumRequiredWifiSecurityLevel()` (/reference/a `setMinimumRequiredWifiSecurityLevel(int)` (/reference/android/a
`int`	`WIFI_SECURITY_PERSONAL` (/reference/android/app/admin/DevicePolicyM Constant for `getMinimumRequiredWifiSecurityLevel()` (/reference/a `setMinimumRequiredWifiSecurityLevel(int)` (/reference/android/a
`int`	`WIPE_EUICC` (/reference/android/app/admin/DevicePolicyManager#WIPE_E Flag for `wipeData(int)` (/reference/android/app/admin/DevicePolicyMana
`int`	`WIPE_EXTERNAL_STORAGE` (/reference/android/app/admin/DevicePolicyMa Flag for `wipeData(int)` (/reference/android/app/admin/DevicePolicyMana
`int`	`WIPE_RESET_PROTECTION_DATA` (/reference/android/app/admin/DevicePo Flag for `wipeData(int)` (/reference/android/app/admin/DevicePolicyMana
`int`	`WIPE_SILENTLY` (/reference/android/app/admin/DevicePolicyManager#WI Flag for `wipeData(int)` (/reference/android/app/admin/DevicePolicyMana

Public methods

`void`	`acknowledgeDeviceCompliant` (/refere Called by a profile owner of an organizatic

void	addCrossProfileIntentFilter (/refe admin, IntentFilter (/reference/andr Called by the profile owner of a managed
boolean	addCrossProfileWidgetProvider (/re (/reference/java/lang/String) packageNa Called by the profile owner of a managed widget providers from a given package to
int	addOverrideApn (/reference/android/ap (/reference/android/telephony/data/ApnSe Called by device owner or managed profil
void	addPersistentPreferredActivity (/ (/reference/android/content/ComponentN Called by a profile owner or device owner
void	addUserRestriction (/reference/andro (/reference/java/lang/String) key) Called by a profile owner, device owner or
void	addUserRestrictionGlobally (/refere Called by a profile owner, device owner or
boolean	bindDeviceAdminServiceAsUser (/ref (ComponentName (/reference/android/co (/reference/android/os/UserHandle) targ Called by a device owner to bind to a serv
boolean	bindDeviceAdminServiceAsUser (/reference/android/app/admin/DevicePoli (ComponentName (/reference/android/co (/reference/android/content/Context.Bind

	See bindDeviceAdminServiceAsUser (/reference/android/app/admin/DevicePoli
boolean	canAdminGrantSensorsPermissions (Returns true if the caller is running on a de
boolean	canUsbDataSignalingBeDisabled (/re Returns whether enabling or disabling USE
void	clearApplicationUserData (/reference/android/app/admin/DevicePoli (ComponentName (/reference/android/co OnClearApplicationUserDataLister Called by the device owner or profile owne
void	clearCrossProfileIntentFilters (/ Called by a profile owner of a managed pr
void	clearDeviceOwnerApp (/reference/andr *This method was deprecated in API level 2 the device owner factory resets the devic*
void	clearPackagePersistentPreferredA (/reference/android/content/ComponentN Called by a profile owner or device owner preferences associated with the given pac (/reference/android/app/admin/DevicePoli
void	clearProfileOwner (/reference/androic *This method was deprecated in API level 2 profile owner deletes it instead of calling t*
boolean	clearResetPasswordToken (/reference

	Called by a profile, device owner or holder token.
`void`	clearUserRestriction (/reference/and (/reference/java/lang/String) key) Called by a profile owner, device owner or
Intent (/reference/android/content/Intent)	createAdminSupportIntent (/referenc Called by any app to display a support dia
UserHandle (/reference/android/os/UserHandle)	createAndManageUser (/reference/andr (/reference/android/content/ComponentN adminExtras, int flags) Called by a device owner to create a user
`int`	enableSystemApp (/reference/android/a (/reference/android/content/Intent) inte Re-enable system apps by intent that were
`void`	enableSystemApp (/reference/android/a packageName) Re-enable a system app that was disabled
AttestedKeyPair (/reference/android/security/AttestedKeyPair)	generateKeyPair (/reference/android/a (/reference/android/content/ComponentN This API can be called by the following to g • Device owner • Profile owner • Delegated certificate installer • Credential management app • An app that holds the Manifest.perm If the device supports key generation via s

String[] (/reference/java/lang/String)	getAccountTypesWithManagementDis Gets the array of accounts for which acco
List (/reference/java/util/List)<ComponentName (/reference/android/content/ComponentName)>	getActiveAdmins (/reference/android/a Return a list of all currently active device a
Set (/reference/java/util/Set)<String (/reference/java/lang/String)>	getAffiliationIds (/reference/androic Returns the set of affiliation ids previously have been set.
Set (/reference/java/util/Set)<String (/reference/java/lang/String)>	getAlwaysOnVpnLockdownWhitelist Called by device or profile owner to query
String (/reference/java/lang/String)	getAlwaysOnVpnPackage (/reference/ar Called by a device or profile owner to reac
Bundle (/reference/android/os/Bundle)	getApplicationRestrictions (/refere (/reference/java/lang/String) packageNa Retrieves the application restrictions for a
String (/reference/java/lang/String)	getApplicationRestrictionsManagi *This method was deprecated in API level 2* (/reference/android/app/admin/DevicePoli (/reference/android/app/admin/DevicePoli
boolean	getAutoTimeEnabled (/reference/andro Returns true if auto time is enabled on the
boolean	getAutoTimeRequired (/reference/andr *This method was deprecated in API level 3* (/reference/android/app/admin/DevicePoli

`boolean`	`getAutoTimeZoneEnabled` (/reference/a Returns true if auto time zone is enabled c
`List` (/reference/java/util/List)`<UserHandle` (/reference/android/os/UserHandle)`>`	`getBindDeviceAdminTargetUsers` (/re Returns the list of target users that the ca (/reference/android/app/admin/DevicePoli .
`boolean`	`getBluetoothContactSharingDisabl` Called by a profile owner of a managed pr
`boolean`	`getCameraDisabled` (/reference/androic Determine whether or not the device's car
`String` (/reference/java/lang/String)	`getCertInstallerPackage` (/reference *This method was deprecated in API level 2* (/reference/android/app/admin/DevicePoli *instead.*
`int`	`getContentProtectionPolicy` (/refere Returns the current content protection po
`PackagePolicy` (/reference/android/app/admin/PackagePolicy)	`getCredentialManagerPolicy` (/refere Called by a device owner or profile owner
`Set` (/reference/java/util/Set)`<String` (/reference/java/lang/String)`>`	`getCrossProfileCalendarPackages` *This method was deprecated in API level 3* (/reference/android/app/admin/DevicePoli
`boolean`	`getCrossProfileCallerIdDisabled` *This method was deprecated in API level 3* (/reference/android/app/admin/DevicePoli

`boolean`	`getCrossProfileContactsSearchDis` *This method was deprecated in API level 3* (/reference/android/app/admin/DevicePoli
Set (/reference/java/util/Set)<String (/reference/java/lang/String)>	`getCrossProfilePackages` (/reference Returns the set of package names that the (/reference/android/app/admin/DevicePoli
List (/reference/java/util/List)<String (/reference/java/lang/String)>	`getCrossProfileWidgetProviders` (/ Called by the profile owner of a managed providers from which packages are availab
`int`	`getCurrentFailedPasswordAttempts` Retrieve the number of times the user has
List (/reference/java/util/List)<String (/reference/java/lang/String)>	`getDelegatePackages` (/reference/andr (/reference/java/lang/String) **delegatio** Called by a profile owner or device owner
List (/reference/java/util/List)<String (/reference/java/lang/String)>	`getDelegatedScopes` (/reference/andro (/reference/java/lang/String) **delegated** Called by a profile owner or device owner
CharSequence (/reference/java/lang/CharSequence)	`getDeviceOwnerLockScreenInfo` (/ref
String (/reference/java/lang/String)	`getDevicePolicyManagementRoleHol` Returns the package name of the device p
CharSequence (/reference/java/lang/CharSequence)	`getEndUserSessionMessage` (/referenc Returns the user session end message.
String (/reference/java/lang/String)	`getEnrollmentSpecificId` (/reference

	Returns an enrollment-specific identifier c
FactoryResetProtectionPolicy (/reference/android/app/admin/FactoryResetProtectionPolicy)	getFactoryResetProtectionPolicy (Callable by device owner or profile owner (/reference/android/app/admin/DevicePoli
String (/reference/java/lang/String)	getGlobalPrivateDnsHost (/reference Returns the system-wide Private DNS host
int	getGlobalPrivateDnsMode (/reference Returns the system-wide Private DNS mod
List (/reference/java/util/List)<byte[]>	getInstalledCaCerts (/reference/andr Returns all CA certificates that are current
List (/reference/java/util/List)<String (/reference/java/lang/String)>	getKeepUninstalledPackages (/refere Get the list of apps to keep around as APK
Map (/reference/java/util/Map)<Integer (/reference/java/lang/Integer), Set (/reference/java/util/Set)<String (/reference/java/lang/String)>>	getKeyPairGrants (/reference/android/ Called by a device or profile owner, or dele access to a given KeyChain key.
int	getKeyguardDisabledFeatures (/refe Determine whether or not features have b
int	getLockTaskFeatures (/reference/andr Gets which system features are enabled fo
String[] (/reference/java/lang/String)	getLockTaskPackages (/reference/andr Returns the list of packages allowed to sta
CharSequence (/reference/java/lang/CharSequence)	getLongSupportMessage (/reference/ar

	Called by a device admin to get the long s
PackagePolicy (/reference/android/app/admin/PackagePolicy)	getManagedProfileCallerIdAccessP Called by a profile owner of a managed pr
PackagePolicy (/reference/android/app/admin/PackagePolicy)	getManagedProfileContactsAccessP Called by a profile owner of a managed pr
long	getManagedProfileMaximumTimeOff Called by a profile owner of an organizatic
ManagedSubscriptionsPolicy (/reference/android/app/admin/ManagedSubscriptionsPolicy)	getManagedSubscriptionsPolicy (/re Returns the current ManagedSubscripti
int	getMaximumFailedPasswordsForWipe Retrieve the current maximum number of
long	getMaximumTimeToLock (/reference/and Retrieve the current maximum time to unlo
List (/reference/java/util/List)<String (/reference/java/lang/String)>	getMeteredDataDisabledPackages (/ Called by a device or profile owner to retri
int	getMinimumRequiredWifiSecurityLe Returns the current Wi-Fi minimum securit
int	getMtePolicy (/reference/android/app/a Called by a device owner, profile owner of
int	getNearbyAppStreamingPolicy (/refe Returns the current runtime nearby app st

`int`	`getNearbyNotificationStreamingPo` Returns the current runtime nearby notific
`int`	`getOrganizationColor` (/reference/and *This method was deprecated in API level 3*
`CharSequence` (/reference/java/lang/CharSequence)	`getOrganizationName` (/reference/andr Called by the device owner (since API 26)
`List` (/reference/java/util/List)<`ApnSetting` (/reference/android/telephony/data/ApnSetting)>	`getOverrideApns` (/reference/android/a Called by device owner or managed profile (/reference/android/app/admin/DevicePoli
`DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager)	`getParentProfileInstance` (/referenc Called by the profile owner of a managed
`int`	`getPasswordComplexity` (/reference/ar Returns how complex the current user's s
`long`	`getPasswordExpiration` (/reference/ar Get the current password expiration time
`long`	`getPasswordExpirationTimeout` (/ref Get the password expiration timeout for th
`int`	`getPasswordHistoryLength` (/referenc Retrieve the current password history leng
`int`	`getPasswordMaximumLength` (/referenc Return the maximum password length tha

`int`	`getPasswordMinimumLength` (/referenc *This method was deprecated in API level 3*
`int`	`getPasswordMinimumLetters` (/referer *This method was deprecated in API level 3*
`int`	`getPasswordMinimumLowerCase` (/refe *This method was deprecated in API level 3*
`int`	`getPasswordMinimumNonLetter` (/refe *This method was deprecated in API level 3*
`int`	`getPasswordMinimumNumeric` (/referer *This method was deprecated in API level 3*
`int`	`getPasswordMinimumSymbols` (/referer *This method was deprecated in API level 3*
`int`	`getPasswordMinimumUpperCase` (/refe *This method was deprecated in API level 3*
`int`	`getPasswordQuality` (/reference/andro *This method was deprecated in API level 3*
`SystemUpdateInfo` (/reference/android/app/admin/SystemUpdateInfo)	`getPendingSystemUpdate` (/reference/a Get information about a pending system u
`int`	`getPermissionGrantState` (/reference `String` (/reference/java/lang/String) **pac** Returns the current grant state of a runtim

`int`	`getPermissionPolicy` (/reference/andr Returns the current runtime permission pc
`List` (/reference/java/util/List)`<String` (/reference/java/lang/String)`>`	`getPermittedAccessibilityService` Returns the list of permitted accessibility s
`List` (/reference/java/util/List)`<String` (/reference/java/lang/String)`>`	`getPermittedCrossProfileNotifica admin)` Returns the list of packages installed on th
`List` (/reference/java/util/List)`<String` (/reference/java/lang/String)`>`	`getPermittedInputMethods` (/referenc Returns the list of permitted input method
`int`	`getPersonalAppsSuspendedReasons` Called by profile owner of an organization
`List` (/reference/java/util/List) `<PreferentialNetworkServiceConfig` (/reference/android/app/admin/PreferentialNetworkServiceConfig) `>`	`getPreferentialNetworkServiceCor` Get preferential network configuration
`int`	`getRequiredPasswordComplexity` (/re Gets the password complexity requiremer
`long`	`getRequiredStrongAuthTimeout` (/ref Determine for how long the user will be ab
`DevicePolicyResourcesManager` (/reference/android/app/admin/DevicePolicyResourcesManager)	`getResources` (/reference/android/app/a Returns a `DevicePolicyResourcesMan`
`boolean`	`getScreenCaptureDisabled` (/referenc Determine whether or not screen capture

List (/reference/java/util/List)<UserHandle (/reference/android/os/UserHandle)>	getSecondaryUsers (/reference/androic Called by a device owner to list all second
CharSequence (/reference/java/lang/CharSequence)	getShortSupportMessage (/reference/a Called by a device admin or holder of the
CharSequence (/reference/java/lang/CharSequence)	getStartUserSessionMessage (/refere Returns the user session start message.
boolean	getStorageEncryption (/reference/anc *This method was deprecated in API level 3* (/reference/android/app/admin/DevicePoli (/reference/android/app/admin/DevicePoli
int	getStorageEncryptionStatus (/refere Called by an application that is administer
SystemUpdatePolicy (/reference/android/app/admin/SystemUpdatePolicy)	getSystemUpdatePolicy (/reference/ar Retrieve a local system update policy set p (/reference/android/app/admin/DevicePoli
PersistableBundle (/reference/android/os/PersistableBundle)	getTransferOwnershipBundle (/refere Returns the data passed from the current
List (/reference/java/util/List)<PersistableBundle (/reference/android/os/PersistableBundle)>	getTrustAgentConfiguration (/refere admin, ComponentName (/reference/anc Gets configuration for the given trust ager (/reference/android/app/admin/DevicePoli
List (/reference/java/util/List)<String (/reference/java/lang/String)>	getUserControlDisabledPackages (/ Returns the list of packages over which us (/reference/android/Manifest.permission#

Bundle (/reference/android/os/Bundle)	getUserRestrictions (/reference/andr Called by an admin to get user restrictions (/reference/android/app/admin/DevicePoli
Bundle (/reference/android/os/Bundle)	getUserRestrictionsGlobally (/refe Called by a profile or device owner to get
String (/reference/java/lang/String)	getWifiMacAddress (/reference/androic Called by a device owner or profile owner
WifiSsidPolicy (/reference/android/app/admin/WifiSsidPolicy)	getWifiSsidPolicy (/reference/androic Returns the current Wi-Fi SSID policy.
boolean	grantKeyPairToApp (/reference/androic (/reference/java/lang/String) alias, St Called by a device or profile owner, or dele access to an already-installed (or generat
boolean	grantKeyPairToWifiAuth (/reference/a Called by a device or profile owner, or dele key pair for authentication to Wifi network
boolean	hasCaCertInstalled (/reference/andro Returns whether this certificate is installe
boolean	hasGrantedPolicy (/reference/android/ Returns true if an administrator has been
boolean	hasKeyPair (/reference/android/app/adm This API can be called by the following to • Device owner • Profile owner

- Delegated certificate installer
- Credential management app
- An app that holds the **Manifest.perm**

If called by the credential management ap

---

boolean **hasLockdownAdminConfiguredNetwor**

Called by a device owner or a profile owne

---

boolean **installCaCert** (/reference/android/app/

Installs the given certificate as a user CA.

---

boolean **installExistingPackage** (/reference/a (/reference/java/lang/String) **packageNa**

Install an existing package that has been i (/reference/android/app/admin/DevicePoli

---

boolean **installKeyPair** (/reference/android/ap (/reference/android/content/ComponentN

This API can be called by the following to i

- Device owner
- Profile owner
- Delegated certificate installer
- Credential management app
- An app that holds the **Manifest.perm**

All apps within the profile will be able to ac

---

boolean **installKeyPair** (/reference/android/ap (/reference/android/content/ComponentN **requestAccess**)

This API can be called by the following to i

- Device owner

	• Profile owner • Delegated certificate installer • Credential management app • An app that holds the Manifest.perm All apps within the profile will be able to ac
boolean	installKeyPair (/reference/android/ap (/reference/android/content/ComponentN This API can be called by the following to i • Device owner • Profile owner • Delegated certificate installer • Credential management app • An app that holds the Manifest.perm All apps within the profile will be able to ac
void	installSystemUpdate (/reference/andr (ComponentName (/reference/android/co (/reference/android/app/admin/DevicePoli Called by device owner or profile owner of
boolean	isActivePasswordSufficient (/refere Determines whether the calling user's cur
boolean	isActivePasswordSufficientForDev Called by profile owner of a managed prof
boolean	isAdminActive (/reference/android/app/ Return true if the given administrator com

boolean	isAffiliatedUser (/reference/android/ Returns whether this user is affiliated with
boolean	isAlwaysOnVpnLockdownEnabled (/ref Called by device or profile owner to query
boolean	isApplicationHidden (/reference/andr (/reference/java/lang/String) packageNa Determine if a package is hidden.
boolean	isBackupServiceEnabled (/reference/a Return whether the backup service is enal (/reference/android/app/admin/DevicePoli
boolean	isCallerApplicationRestrictionsM *This method was deprecated in API level 2* (/reference/android/app/admin/DevicePoli
boolean	isCommonCriteriaModeEnabled (/refe Returns whether Common Criteria mode is
boolean	isComplianceAcknowledgementRequi Called by a profile owner of an organizatic
boolean	isDeviceFinanced (/reference/android/ Returns `true` if this device is marked as a
boolean	isDeviceIdAttestationSupported (/ Returns `true` if the device supports attest
boolean	isDeviceOwnerApp (/reference/android/

	Used to determine if a particular package
`boolean`	`isEphemeralUser` (/reference/android/a Checks if the profile owner is running in ar
`boolean`	`isKeyPairGrantedToWifiAuth` (/refere Called by a device or profile owner, or dele KeyChain key pair can be used for authen
`boolean`	`isLockTaskPermitted` (/reference/andr This function lets the caller know whether
`boolean`	`isLogoutEnabled` (/reference/android/a Returns whether logout is enabled by a de
`boolean`	`isManagedProfile` (/reference/android/ Return if this user is a managed profile of
`boolean`	`isMasterVolumeMuted` (/reference/andr Called by profile or device owners to chec
`static boolean`	`isMtePolicyEnforced` (/reference/andr Get the current MTE state of the device.
`boolean`	`isNetworkLoggingEnabled` (/reference Return whether network logging is enable
`boolean`	`isOrganizationOwnedDeviceWithMar Apps can use this method to find out if the
`boolean`	`isOverrideApnEnabled` (/reference/and

	Called by device owner to check if overrid
boolean	isPackageSuspended (/reference/andro (/reference/java/lang/String) packageNa Determine if a package is suspended.
boolean	isPreferentialNetworkServiceEnab Indicates whether preferential network se
boolean	isProfileOwnerApp (/reference/androic Used to determine if a particular package
boolean	isProvisioningAllowed (/reference/ar Returns whether it is possible for the calle
boolean	isResetPasswordTokenActive (/refere Called by a profile, device owner or a hold token is active.
boolean	isSafeOperation (/reference/android/a Checks if it's safe to run operations that c
boolean	isSecurityLoggingEnabled (/referenc Return whether security logging is enable
boolean	isStatusBarDisabled (/reference/andr Returns whether the status bar is disablec
boolean	isUninstallBlocked (/reference/andro (/reference/java/lang/String) packageNa Check whether the user has been blocked

`boolean`	`isUniqueDeviceAttestationSupport` Returns `true` if the StrongBox Keymaster implementations with StrongBox security l
`boolean`	`isUsbDataSignalingEnabled` (/referer Returns whether USB data signaling is cur
`boolean`	`isUsingUnifiedPassword` (/reference/a When called by a profile owner of a manag
`List` (/reference/java/util/List)`<UserHandle` (/reference/android/os/UserHandle)`>`	`listForegroundAffiliatedUsers` (/re Gets the list of `affiliated` (/reference/a
`void`	`lockNow` (/reference/android/app/admin/D Make the device lock immediately, as if the
`void`	`lockNow` (/reference/android/app/admin/D Make the device lock immediately, as if the
`int`	`logoutUser` (/reference/android/app/adm Called by a profile owner of secondary use (/reference/android/app/admin/DevicePoli (/reference/android/app/admin/DevicePoli
`void`	`reboot` (/reference/android/app/admin/De Called by device owner to reboot the devi
`void`	`removeActiveAdmin` (/reference/androic Remove a current administration compone
`boolean`	`removeCrossProfileWidgetProvider` `String` (/reference/java/lang/String) `pac`

	Called by the profile owner of a managed widget providers from a given package to
`boolean`	`removeKeyPair` (/reference/android/app/ This API can be called by the following to r • Device owner • Profile owner • Delegated certificate installer • Credential management app From Android `Build.VERSION_CODES.S`
`boolean`	`removeOverrideApn` (/reference/androic Called by device owner or managed profile
`boolean`	`removeUser` (/reference/android/app/adm (/reference/android/os/UserHandle) `user` Called by a device owner to remove a user
`boolean`	`requestBugreport` (/reference/android/ Called by a device owner to request a bug
`boolean`	`resetPassword` (/reference/android/app/ *This method was deprecated in API level 3* (/reference/android/app/admin/DevicePoli
`boolean`	`resetPasswordWithToken` (/reference/a `String` (/reference/java/lang/String) `pas` Called by device or profile owner to force
`List` (/reference/java/util/List)<`NetworkEvent` (/reference/android/app/admin/NetworkEvent)>	`retrieveNetworkLogs` (/reference/andr Called by device owner, profile owner of a events.

List (/reference/java/util/List)<SecurityLog.SecurityEvent (/reference/android/app/admin/SecurityLog.SecurityEvent)>	retrievePreRebootSecurityLogs (/re Called by device owner or profile owner of
List (/reference/java/util/List)<SecurityLog.SecurityEvent (/reference/android/app/admin/SecurityLog.SecurityEvent)>	retrieveSecurityLogs (/reference/and Called by device owner or profile owner of
boolean	revokeKeyPairFromApp (/reference/and (/reference/java/lang/String) alias, St Called by a device or profile owner, or dele grant to a KeyChain key pair.
boolean	revokeKeyPairFromWifiAuth (/referer Called by a device or profile owner, or dele key pair for authentication to Wifi network
void	setAccountManagementDisabled (/ref admin, String (/reference/java/lang/Str Called by a device owner or profile owner
void	setAffiliationIds (/reference/androic (/reference/java/util/Set)<String (/refere Indicates the entity that controls the devic
void	setAlwaysOnVpnPackage (/reference/ar (/reference/java/lang/String) vpnPackag Called by a device or profile owner to conf
void	setAlwaysOnVpnPackage (/reference/ar (/reference/android/content/ComponentN A version of setAlwaysOnVpnPackage((/reference/android/app/admin/DevicePoli connected.

boolean	setApplicationHidden (/reference/and (/reference/java/lang/String) packageNa Hide or unhide packages.
void	setApplicationRestrictions (/refere admin, String (/reference/java/lang/Str Sets the application restrictions for a give
void	setApplicationRestrictionsManagi (/reference/android/content/ComponentN *This method was deprecated in API level 2* (/reference/android/app/admin/DevicePoli (/reference/android/app/admin/DevicePoli
void	setAutoTimeEnabled (/reference/andro Called by a device owner, a profile owner f
void	setAutoTimeRequired (/reference/andr *This method was deprecated in API level 3* (/reference/android/app/admin/DevicePoli (/reference/android/os/UserManager#DIS
void	setAutoTimeZoneEnabled (/reference/a Called by a device owner, a profile owner f
void	setBackupServiceEnabled (/reference Allows the device owner or profile owner t
void	setBluetoothContactSharingDisabl boolean disabled) Called by a profile owner of a managed pr
void	setCameraDisabled (/reference/androic

	Called by an application that is administer
void	setCertInstallerPackage (/reference (/reference/java/lang/String) installer *This method was deprecated in API level 2* (/reference/android/app/admin/DevicePoli (/reference/android/app/admin/DevicePoli
void	setCommonCriteriaModeEnabled (/ref enabled) Called by device owner or profile owner of
void	setConfiguredNetworksLockdownSta boolean lockdown) Called by a device owner or a profile owne
void	setContentProtectionPolicy (/refere Sets the content protection policy which c
void	setCredentialManagerPolicy (/refere Called by a device owner or profile owner
void	setCrossProfileCalendarPackages admin, Set (/reference/java/util/Set)<St *This method was deprecated in API level 3* (/reference/android/app/admin/DevicePoli
void	setCrossProfileCallerIdDisabled disabled) *This method was deprecated in API level 3* setManagedProfileCallerIdAccessP
void	setCrossProfileContactsSearchDis admin, boolean disabled)

	This method was deprecated in API level 3 (/reference/android/app/admin/DevicePoli
`void`	`setCrossProfilePackages` (/reference (/reference/java/util/Set)`<String` (/refere Sets the set of admin-allowlisted package
`void`	`setDefaultDialerApplication` (/refe Must be called by a device owner or a pro
`void`	`setDefaultSmsApplication` (/referenc (/reference/java/lang/String) **packageNa** Must be called by a device owner or a pro
`void`	`setDelegatedScopes` (/reference/andro `String` (/reference/java/lang/String) **del** Called by a profile owner or device owner
`void`	`setDeviceOwnerLockScreenInfo` (/ref `CharSequence` (/reference/java/lang/Cha Sets the device owner information to be sl
`void`	`setEndUserSessionMessage` (/referenc `CharSequence` (/reference/java/lang/Cha Called by a device owner to specify the us
`void`	`setFactoryResetProtectionPolicy` (/reference/android/content/ComponentN Callable by device owner or profile owner
`int`	`setGlobalPrivateDnsModeOpportuni` Sets the global Private DNS mode to oppo

`int`	`setGlobalPrivateDnsModeSpecifiec` `admin, String` (/reference/java/lang/Str Sets the global Private DNS host to be use
`void`	`setGlobalSetting` (/reference/android/ (/reference/java/lang/String) `setting,` This method is mostly deprecated.
`void`	`setKeepUninstalledPackages` (/refere `List` (/reference/java/util/List)`<String` (/ Set a list of apps to keep around as APKs
`boolean`	`setKeyPairCertificate` (/reference/ar (/reference/android/content/ComponentN This API can be called by the following to a (/reference/android/app/admin/DevicePoli certificate selection prompt: • Device owner • Profile owner • Delegated certificate installer • Credential management app From Android `Build.VERSION_CODES.S`
`boolean`	`setKeyguardDisabled` (/reference/andr Called by a device owner or profile owner
`void`	`setKeyguardDisabledFeatures` (/refe Called by an application that is administer
`void`	`setLocationEnabled` (/reference/andro Called by device owners to set the user's

void	setLockTaskFeatures (/reference/andr Sets which system features are enabled w
void	setLockTaskPackages (/reference/andr (/reference/java/lang/String) packages) Sets which packages may enter lock task r
void	setLogoutEnabled (/reference/android/ Called by a device owner to specify wheth
void	setLongSupportMessage (/reference/ar (/reference/java/lang/CharSequence) me Called by a device admin to set the long su
void	setManagedProfileCallerIdAccessP Called by a profile owner of a managed pr
void	setManagedProfileContactsAccessP Called by a profile owner of a managed pr
void	setManagedProfileMaximumTimeOff (timeoutMillis) Called by a profile owner of an organizatic
void	setManagedSubscriptionsPolicy (/re (/reference/android/app/admin/ManagedS Called by a profile owner of an organizatic Managed subscriptions policy controls hov
void	setMasterVolumeMuted (/reference/and Called by profile or device owners to set th

void	setMaximumFailedPasswordsForWipe Setting this to a value greater than zero er
void	setMaximumTimeToLock (/reference/and Called by an application that is administer
List (/reference/java/util/List)<String (/reference/java/lang/String)>	setMeteredDataDisabledPackages (/ admin, List (/reference/java/util/List)<S Called by a device or profile owner to restr
void	setMinimumRequiredWifiSecurityLe Called by device owner or profile owner of
void	setMtePolicy (/reference/android/app/a Called by a device owner, profile owner of
void	setNearbyAppStreamingPolicy (/refe Called by a device/profile owner to set nea
void	setNearbyNotificationStreamingPo Called by a device/profile owner to set nea
void	setNetworkLoggingEnabled (/referenc Called by a device owner, profile owner of
void	setOrganizationColor (/reference/and *This method was deprecated in API level 3*
void	setOrganizationId (/reference/androic Sets the Enterprise ID for the work profile

void	setOrganizationName (/reference/andr (/reference/java/lang/CharSequence) ti Called by the device owner (since API 26)
void	setOverrideApnsEnabled (/reference/a Called by device owner to set if override A
String[] (/reference/java/lang/String)	setPackagesSuspended (/reference/anc (/reference/java/lang/String) packageNa Called by device or profile owners to susp
void	setPasswordExpirationTimeout (/ref Called by a device admin to set the passw
void	setPasswordHistoryLength (/referenc Called by an application that is administer
void	setPasswordMinimumLength (/referenc *This method was deprecated in API level 3*
void	setPasswordMinimumLetters (/referer *This method was deprecated in API level 3*
void	setPasswordMinimumLowerCase (/refe *This method was deprecated in API level 3*
void	setPasswordMinimumNonLetter (/refe *This method was deprecated in API level 3*
void	setPasswordMinimumNumeric (/referer *This method was deprecated in API level 3*

`void`	`setPasswordMinimumSymbols` (/referer *This method was deprecated in API level 3*
`void`	`setPasswordMinimumUpperCase` (/refe *This method was deprecated in API level 3*
`void`	`setPasswordQuality` (/reference/andro *This method was deprecated in API level 3* *platform, rather than specifying custom pa* *throttling to thwart online and offline brute* `setRequiredPasswordComplexity(ir`
`boolean`	`setPermissionGrantState` (/reference `admin, String` (/reference/java/lang/Str Sets the grant state of a runtime permissio
`void`	`setPermissionPolicy` (/reference/andr Set the default response for future runtim
`boolean`	`setPermittedAccessibilityService` (/reference/android/content/ComponentN Called by a profile or device owner to set t
`boolean`	`setPermittedCrossProfileNotifica` (/reference/android/content/ComponentN Called by a profile owner of a managed pr
`boolean`	`setPermittedInputMethods` (/referenc (/reference/java/util/List)`<String` (/refere Called by a profile or device owner or hold services for this user.
`void`	`setPersonalAppsSuspended` (/referenc

	Called by a profile owner of an organizatic
`void`	setPreferentialNetworkServiceCor <PreferentialNetworkServiceConfi Sets preferential network configurations.
`void`	setPreferentialNetworkServiceEna Sets whether preferential network service
`void`	setProfileEnabled (/reference/androic Sets the enabled state of the profile.
`void`	setProfileName (/reference/android/ap profileName) Sets the name of the profile.
`void`	setRecommendedGlobalProxy (/referer ProxyInfo (/reference/android/net/Proxy Set a network-independent global HTTP p
`void`	setRequiredPasswordComplexity (/re Sets a minimum password complexity requ
`void`	setRequiredStrongAuthTimeout (/ref Called by a device/profile owner to set the
`boolean`	setResetPasswordToken (/reference/ar Called by a profile or device owner to prov String, byte, int) (/reference/andro
`void`	setRestrictionsProvider (/reference ComponentName (/reference/android/cont Designates a specific service component

`void`	`setScreenCaptureDisabled` (/referenc Called by a device/profile owner to set wh
`void`	`setSecureSetting` (/reference/android/ (/reference/java/lang/String) **`setting,`** This method is mostly deprecated.
`void`	`setSecurityLoggingEnabled` (/referer Called by device owner or a profile owner
`void`	`setShortSupportMessage` (/reference/a `CharSequence` (/reference/java/lang/Cha Called by a device admin to set the short s
`void`	`setStartUserSessionMessage` (/refere `CharSequence` (/reference/java/lang/Cha Called by a device owner to specify the us
`boolean`	`setStatusBarDisabled` (/reference/anc Called by device owner or profile owner of
`int`	`setStorageEncryption` (/reference/anc *This method was deprecated in API level 3* *system be encrypted. Does nothing if the* *When multiple device administrators atten* *device encryption while another device ac* *This policy controls encryption of the secu* `Environment.isExternalStorageEmu` (/reference/android/os/Environment#getE *Important Note: On some devices, it is pos* *also require (and check for) a pattern, PIN*

void	setSystemSetting (/reference/android/ (/reference/java/lang/String) setting, Called by a device or profile owner to upd
void	setSystemUpdatePolicy (/reference/ar SystemUpdatePolicy (/reference/andro Called by device owners or profile owners
boolean	setTime (/reference/android/app/admin/D Called by a device owner or a profile owne
boolean	setTimeZone (/reference/android/app/ad Called by a device owner or a profile owne
void	setTrustAgentConfiguration (/refere (/reference/android/content/ComponentN Sets a list of configuration features to ena
void	setUninstallBlocked (/reference/andr (/reference/java/lang/String) packageNa Change whether a user can uninstall a pac
void	setUsbDataSignalingEnabled (/refere Called by a device owner or profile owner
void	setUserControlDisabledPackages (/ admin, List (/reference/java/util/List)<S Called by a device owner or a profile owne
void	setUserIcon (/reference/android/app/ad (/reference/android/graphics/Bitmap) ic Called by profile or device owners to set tl

void	setWifiSsidPolicy (/reference/androic Called by device owner or profile owner of
int	startUserInBackground (/reference/ar (/reference/android/os/UserHandle) user Called by a device owner to start the spec
int	stopUser (/reference/android/app/admin/ userHandle) Called by a device owner to stop the spec
boolean	switchUser (/reference/android/app/adm (/reference/android/os/UserHandle) user Called by a device owner to switch the spe
void	transferOwnership (/reference/androic (/reference/android/content/ComponentN Changes the current administrator to anot
void	uninstallAllUserCaCerts (/reference Uninstalls all custom trusted CA certificate
void	uninstallCaCert (/reference/android/ap Uninstalls the given certificate from truste
boolean	updateOverrideApn (/reference/androic apnId, ApnSetting (/reference/androic Called by device owner or managed profile
void	wipeData (/reference/android/app/admin/ Ask that all user data be wiped.

void	wipeData (/reference/android/app/admin/ See wipeData(int, java.lang.CharS
void	wipeDevice (/reference/android/app/adm Ask that the device be wiped and factory r

Inherited methods

From class java.lang.Object (/reference/java/lang/Object)

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

`final void`	`notifyAll` (/reference/java/lang/Object#notifyAll())`()` Wakes up all threads that are waiting on this object's monitor.
`String` (/reference/java/lang/String)	`toString` (/reference/java/lang/Object#toString())`()` Returns a string representation of the object.
`final void`	`wait` (/reference/java/lang/Object#wait(long,%20int))`(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` (/reference/java/lang/Object#wait(long))`(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` (/reference/java/lang/Object#wait())`()` Causes the current thread to wait until it is awakened, typically by being *notified* or *interrupted*.

Constants

ACTION_ADD_DEVICE_ADMIN

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_ADD_DEVICE_ADMIN
```

Activity action: ask the user to add a new device administrator to the system. The desired policy is the ComponentName of the policy in the `EXTRA_DEVICE_ADMIN` (/reference/android/app/admin/DevicePolicyManager#EXTRA_DEVICE_ADMIN) extra field. This will

invoke a UI to bring the user through adding the device administrator to the system (or allowing them to reject it).

You can optionally include the `EXTRA_ADD_EXPLANATION` (/reference/android/app/admin/DevicePolicyManager#EXTRA_ADD_EXPLANATION) field to provide the user with additional explanation (in addition to your component's description) about what is being added.

If your administrator is already active, this will ordinarily return immediately (without user intervention). However, if your administrator has been updated and is requesting additional uses-policy flags, the user will be presented with the new list. New policies will not be available to the updated administrator until the user has accepted the new list.

Constant Value: "android.app.action.ADD_DEVICE_ADMIN"

ACTION_ADMIN_POLICY_COMPLIANCE

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_ADMIN_POLICY_COMPLIANCE
```

Activity action: Starts the administrator to show policy compliance for the provisioning. This action is used any time that the administrator has an opportunity to show policy compliance before the end of setup wizard. This could happen as part of the admin-integrated provisioning flow (in which case this gets sent after `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE)), or it could happen during provisioning finalization if the administrator supports finalization during setup wizard.

Intents with this action may also be supplied with the `EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE) extra.

See also:

`ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE)

Constant Value: "android.app.action.ADMIN_POLICY_COMPLIANCE"

ACTION_APPLICATION_DELEGATION_SCOPES_CHANGED

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_APPLICATION_DELEGATION_SCO
```

Broadcast Action: Sent after application delegation scopes are changed. The new delegation scopes will be sent in an `ArrayList<String>` extra identified by the `EXTRA_DELEGATION_SCOPES` (/reference/android/app/admin/DevicePolicyManager#EXTRA_DELEGATION_SCOPES) key.

Note: This is a protected intent that can only be sent by the system.

Constant Value: "android.app.action.APPLICATION_DELEGATION_SCOPES_CHANGED"

ACTION_CHECK_POLICY_COMPLIANCE

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_CHECK_POLICY_COMPLIANCE
```

Activity action: launch the DPC to check policy compliance. This intent is launched when the user taps on the notification about personal apps suspension. When handling this intent the DPC must check if personal apps should still be suspended and either unsuspend them or instruct the user on how to resolve the noncompliance causing the suspension.

See also:

`setPersonalAppsSuspended(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setPersonalAppsSuspended(android.content.ComponentName,%20boolean))

Constant Value: "android.app.action.CHECK_POLICY_COMPLIANCE"

ACTION_DEVICE_ADMIN_SERVICE

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_DEVICE_ADMIN_SERVICE
```

Service action: Action for a service that device owner and profile owner can optionally own. If a device owner or a profile owner has such a service, the system tries to keep a bound connection to it, in order to keep their process always running. The service must be protected with the `Manifest.permission.BIND_DEVICE_ADMIN` (/reference/android/Manifest.permission#BIND_DEVICE_ADMIN) permission.

Constant Value: "android.app.action.DEVICE_ADMIN_SERVICE"

ACTION_DEVICE_FINANCING_STATE_CHANGED

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_DEVICE_FINANCING_STATE_CHA
```

Broadcast Action: Broadcast sent to indicate that the device financing state has changed.

This occurs when, for example, a financing kiosk app has been added or removed.

To query the current device financing state see `isDeviceFinanced()` (/reference/android/app/admin/DevicePolicyManager#isDeviceFinanced()).

This will be delivered to the following apps if they include a receiver for this action in their manifest:

- Device owner admins.
- Organization-owned profile owner admins
- The supervision app
- The device management role holder

Constant Value: "android.app.admin.action.DEVICE_FINANCING_STATE_CHANGED"

ACTION_DEVICE_OWNER_CHANGED

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_DEVICE_OWNER_CHANGED
```

Broadcast action: sent when the device owner is set, changed or cleared. This broadcast is sent only to the primary user.

See also:

`ACTION_PROVISION_MANAGED_DEVICE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_DEVICE)

`transferOwnership(ComponentName, ComponentName, PersistableBundle)` (/reference/android/app/admin/DevicePolicyManager#transferOwnership(android.content.ComponentName,%20android.content.ComponentName,%20android.os.PersistableBundle))

Constant Value: "android.app.action.DEVICE_OWNER_CHANGED"

ACTION_DEVICE_POLICY_RESOURCE_UPDATED

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_DEVICE_POLICY_RESOURCE_UPD
```

Broadcast action: notify system apps (e.g. settings, SysUI, etc) that the device management resources with IDs `EXTRA_RESOURCE_IDS` (/reference/android/app/admin/DevicePolicyManager#EXTRA_RESOURCE_IDS) has been updated, the updated resources can be retrieved using `DevicePolicyResourcesManager#getDrawable` (/reference/android/app/admin/DevicePolicyResourcesManager#getDrawable(java.lang.String,%20java.lang.String,%20java.lang.String,%20java.util.function.Supplier<android.graphics.drawable.Drawable>)) and `DevicePolicyResourcesManager#getString` (/reference/android/app/admin/DevicePolicyResourcesManager#getString(java.lang.String,%20java.util.function.Supplier<java.lang.String>))
.

This broadcast is sent to registered receivers only.

`EXTRA_RESOURCE_TYPE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_RESOURCE_TYPE) will be included to identify the type of resource being updated.

Constant Value: "android.app.action.DEVICE_POLICY_RESOURCE_UPDATED"

ACTION_GET_PROVISIONING_MODE

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_GET_PROVISIONING_MODE
```

Activity action: Starts the administrator to get the mode for the provisioning. This intent may contain the following extras:

- `EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE)
- `EXTRA_PROVISIONING_IMEI` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_IMEI)
- `EXTRA_PROVISIONING_SERIAL_NUMBER` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_SERIAL_NUMBER)
- `EXTRA_PROVISIONING_ALLOWED_PROVISIONING_MODES` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_ALLOWED_PROVISIONING_MODES)
- `EXTRA_PROVISIONING_SENSORS_PERMISSION_GRANT_OPT_OUT` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_SENSORS_PERMISSION_GRANT_OPT_OUT)

The target activity should return one of the following values in `EXTRA_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_MODE) as result:

- `PROVISIONING_MODE_FULLY_MANAGED_DEVICE` (/reference/android/app/admin/DevicePolicyManager#PROVISIONING_MODE_FULLY_MANAGED_DEVICE)
- `PROVISIONING_MODE_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#PROVISIONING_MODE_MANAGED_PROFILE)

If performing fully-managed device provisioning and the admin app desires to show its own education screens, the target activity can additionally return `EXTRA_PROVISIONING_SKIP_EDUCATION_SCREENS` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_SKIP_EDUCATION_SCREENS) set to `true`.

The target activity may also return the account that needs to be migrated from primary user to managed profile in case of a profile owner provisioning in `EXTRA_PROVISIONING_ACCOUNT_TO_MIGRATE`

(/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_ACCOUNT_TO_MIGRATE) as result.

The target activity may also include the `EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE) extra in the intent result. The values of this `PersistableBundle` (/reference/android/os/PersistableBundle) will be sent as an intent extra of the same name to the `ACTION_ADMIN_POLICY_COMPLIANCE` (/reference/android/app/admin/DevicePolicyManager#ACTION_ADMIN_POLICY_COMPLIANCE) activity, along with the values of the `EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE) extra that are already supplied to this activity.

Other extras the target activity may include in the intent result:

- `EXTRA_PROVISIONING_DISCLAIMERS` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DISCLAIMERS)
- `EXTRA_PROVISIONING_SKIP_ENCRYPTION` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_SKIP_ENCRYPTION)
- `EXTRA_PROVISIONING_KEEP_SCREEN_ON` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_KEEP_SCREEN_ON)
- `EXTRA_PROVISIONING_KEEP_ACCOUNT_ON_MIGRATION` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_KEEP_ACCOUNT_ON_MIGRATION)
  for work profile provisioning
- `EXTRA_PROVISIONING_LEAVE_ALL_SYSTEM_APPS_ENABLED` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_LEAVE_ALL_SYSTEM_APPS_ENABLED)
  for work profile provisioning
- `EXTRA_PROVISIONING_SENSORS_PERMISSION_GRANT_OPT_OUT` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_SENSORS_PERMISSION_GRANT_OPT_OUT)
  for fully-managed device provisioning
- `EXTRA_PROVISIONING_LOCALE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_LOCALE) for fully-managed device provisioning

- `EXTRA_PROVISIONING_LOCAL_TIME` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_LOCAL_TIME) for fully-managed device provisioning
- `EXTRA_PROVISIONING_TIME_ZONE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_TIME_ZONE) for fully-managed device provisioning

See also:

`ACTION_ADMIN_POLICY_COMPLIANCE` (/reference/android/app/admin/DevicePolicyManager#ACTION_ADMIN_POLICY_COMPLIANCE)

Constant Value: "android.app.action.GET_PROVISIONING_MODE"

ACTION_MANAGED_PROFILE_PROVISIONED

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_MANAGED_PROFILE_PROVISIONE
```

Broadcast Action: This broadcast is sent to indicate that provisioning of a managed profile has completed successfully.

The broadcast is limited to the primary profile, to the app specified in the provisioning intent with action `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE).

This intent will contain the following extras

- `Intent#EXTRA_USER` (/reference/android/content/Intent#EXTRA_USER), corresponds to the `UserHandle` (/reference/android/os/UserHandle) of the managed profile.
- `EXTRA_PROVISIONING_ACCOUNT_TO_MIGRATE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_ACCOUNT_TO_MIGRATE), corresponds to the account requested to be migrated at provisioning time, if any.

Constant Value: "android.app.action.MANAGED_PROFILE_PROVISIONED"

ACTION_PROFILE_OWNER_CHANGED

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_PROFILE_OWNER_CHANGED
```

Broadcast action: sent when the profile owner is set, changed or cleared. This broadcast is sent only to the user managed by the new profile owner.

See also:

`transferOwnership(ComponentName, ComponentName, PersistableBundle)` (/reference/android/app/admin/DevicePolicyManager#transferOwnership(android.content.ComponentName,%20android.content.ComponentName,%20android.os.PersistableBundle))

Constant Value: "android.app.action.PROFILE_OWNER_CHANGED"

ACTION_PROVISIONING_SUCCESSFUL

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_PROVISIONING_SUCCESSFUL
```

Activity action: This activity action is sent to indicate that provisioning of a managed profile or managed device has completed successfully. It'll be sent at the same time as `DeviceAdminReceiver#ACTION_PROFILE_PROVISIONING_COMPLETE` (/reference/android/app/admin/DeviceAdminReceiver#ACTION_PROFILE_PROVISIONING_COMPLETE) broadcast but this will be delivered faster as it's an activity intent.

The intent is only sent to the new device or profile owner.

See also:

`ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE)

`ACTION_PROVISION_MANAGED_DEVICE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_DEVICE)

Constant Value: "android.app.action.PROVISIONING_SUCCESSFUL"

ACTION_PROVISION_MANAGED_DEVICE

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_PROVISION_MANAGED_DEVICE
```

This constant was deprecated in API level 31.
to support `Build.VERSION_CODES.S` (/reference/android/os/Build.VERSION_CODES#S) and later, admin apps must implement activities with intent filters for the `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE) and `ACTION_ADMIN_POLICY_COMPLIANCE` (/reference/android/app/admin/DevicePolicyManager#ACTION_ADMIN_POLICY_COMPLIANCE) intent actions; using `ACTION_PROVISION_MANAGED_DEVICE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_DEVICE) to start provisioning will cause the provisioning to fail; to additionally support pre-`Build.VERSION_CODES.S` (/reference/android/os/Build.VERSION_CODES#S), admin apps must also continue to use this constant.

Activity action: Starts the provisioning flow which sets up a managed device. Must be started with `Activity.startActivityForResult(Intent, int)` (/reference/android/app/Activity#startActivityForResult(android.content.Intent,%20int)).

During device owner provisioning a device admin app is set as the owner of the device. A device owner has full control over the device. The device owner can not be modified by the user.

A typical use case would be a device that is owned by a company, but used by either an employee or client.

An intent with this action can be sent only on an unprovisioned device. It is possible to check if provisioning is allowed or not by querying the method `isProvisioningAllowed(java.lang.String)` (/reference/android/app/admin/DevicePolicyManager#isProvisioningAllowed(java.lang.String)).

The intent contains the following extras:

- `EXTRA_PROVISIONING_DEVICE_ADMIN_COMPONENT_NAME` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_COMPONENT_NAME)

- `EXTRA_PROVISIONING_SKIP_ENCRYPTION` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_SKIP_ENCRYPTION), optional
- `EXTRA_PROVISIONING_LEAVE_ALL_SYSTEM_APPS_ENABLED` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_LEAVE_ALL_SYSTEM_APPS_ENABLED), optional
- `EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE), optional
- `EXTRA_PROVISIONING_LOGO_URI` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_LOGO_URI), optional
- `EXTRA_PROVISIONING_DISCLAIMERS` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DISCLAIMERS), optional
- `EXTRA_PROVISIONING_SKIP_EDUCATION_SCREENS` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_SKIP_EDUCATION_SCREENS), optional

When device owner provisioning has completed, an intent of the type `DeviceAdminReceiver#ACTION_PROFILE_PROVISIONING_COMPLETE` (/reference/android/app/admin/DeviceAdminReceiver#ACTION_PROFILE_PROVISIONING_COMPLETE) is broadcast to the device owner.

From version `Build.VERSION_CODES.O` (/reference/android/os/Build.VERSION_CODES#O), when device owner provisioning has completed, along with the above broadcast, activity intent `ACTION_PROVISIONING_SUCCESSFUL` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISIONING_SUCCESSFUL) will also be sent to the device owner.

If provisioning fails, the device is factory reset.

A result code of `Activity.RESULT_OK` (/reference/android/app/Activity#RESULT_OK) implies that the synchronous part of the provisioning flow was successful, although this doesn't guarantee the full flow will succeed. Conversely a result code of `Activity.RESULT_CANCELED`

(/reference/android/app/Activity#RESULT_CANCELED) implies that the user backed-out of provisioning, or some precondition for provisioning wasn't met.

Constant Value: "android.app.action.PROVISION_MANAGED_DEVICE"

ACTION_PROVISION_MANAGED_PROFILE

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_PROVISION_MANAGED_PROFILE
```

Activity action: Starts the provisioning flow which sets up a managed profile.

A managed profile allows data separation for example for the usage of a device as a personal and corporate device. The user which provisioning is started from and the managed profile share a launcher.

This intent will typically be sent by a mobile device management application (MDM). Provisioning adds a managed profile and sets the MDM as the profile owner who has full control over the profile.

It is possible to check if provisioning is allowed or not by querying the method `isProvisioningAllowed(java.lang.String)` (/reference/android/app/admin/DevicePolicyManager#isProvisioningAllowed(java.lang.String)).

In version `Build.VERSION_CODES.LOLLIPOP` (/reference/android/os/Build.VERSION_CODES#LOLLIPOP), this intent must contain the extra `EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_NAME` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_NAME). As of `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M), it should contain the extra `EXTRA_PROVISIONING_DEVICE_ADMIN_COMPONENT_NAME` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_COMPONENT_NAME) instead, although specifying only `EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_NAME` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_NAME) is still supported.

The intent may also contain the following extras:

- `EXTRA_PROVISIONING_ACCOUNT_TO_MIGRATE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_ACCOUNT_TO_MIGRATE), optional
- `EXTRA_PROVISIONING_SKIP_ENCRYPTION` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_SKIP_ENCRYPTION), optional, supported from `Build.VERSION_CODES.N` (/reference/android/os/Build.VERSION_CODES#N)
- `EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE), optional
- `EXTRA_PROVISIONING_LOGO_URI` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_LOGO_URI), optional
- `EXTRA_PROVISIONING_SKIP_USER_CONSENT` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_SKIP_USER_CONSENT), optional
- `EXTRA_PROVISIONING_KEEP_ACCOUNT_ON_MIGRATION` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_KEEP_ACCOUNT_ON_MIGRATION), optional
- `EXTRA_PROVISIONING_DISCLAIMERS` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DISCLAIMERS), optional

When managed provisioning has completed, broadcasts are sent to the application specified in the provisioning intent. The `DeviceAdminReceiver#ACTION_PROFILE_PROVISIONING_COMPLETE` (/reference/android/app/admin/DeviceAdminReceiver#ACTION_PROFILE_PROVISIONING_COMPLETE) broadcast is sent in the managed profile and the `ACTION_MANAGED_PROFILE_PROVISIONED` (/reference/android/app/admin/DevicePolicyManager#ACTION_MANAGED_PROFILE_PROVISIONED) broadcast is sent in the primary profile.

From version `Build.VERSION_CODES.O` (/reference/android/os/Build.VERSION_CODES#O), when managed provisioning has completed, along with the above broadcast, activity intent `ACTION_PROVISIONING_SUCCESSFUL`

(/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISIONING_SUCCESSFUL) will also be sent to the profile owner.

If provisioning fails, the managedProfile is removed so the device returns to its previous state.

If launched with `Activity.startActivityForResult(Intent, int)` (/reference/android/app/Activity#startActivityForResult(android.content.Intent,%20int)) a result code of `Activity.RESULT_OK` (/reference/android/app/Activity#RESULT_OK) implies that the synchronous part of the provisioning flow was successful, although this doesn't guarantee the full flow will succeed. Conversely a result code of `Activity.RESULT_CANCELED` (/reference/android/app/Activity#RESULT_CANCELED) implies that the user backed-out of provisioning, or some precondition for provisioning wasn't met.

If a device policy management role holder (DPMRH) updater is present on the device, an internet connection attempt must be made prior to launching this intent. If internet connection could not be established, provisioning will fail unless `EXTRA_PROVISIONING_ALLOW_OFFLINE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_ALLOW_OFFLINE) is explicitly set to `true`, in which case provisioning will continue without using the DPMRH. If an internet connection has been established, the DPMRH updater will be launched, which will update the DPMRH if it's not present on the device, or if it's present and not valid.

If a DPMRH is present on the device and valid, the provisioning flow will be deferred to it.

Constant Value: "android.app.action.PROVISION_MANAGED_PROFILE"

ACTION_SET_NEW_PARENT_PROFILE_PASSWORD

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_SET_NEW_PARENT_PROFILE_PAS
```

Activity action: have the user enter a new password for the parent profile. If the intent is launched from within a managed profile, this will trigger entering a new password for the parent of the profile. The caller can optionally set `EXTRA_DEVICE_PASSWORD_REQUIREMENT_ONLY` (/reference/android/app/admin/DevicePolicyManager#EXTRA_DEVICE_PASSWORD_REQUIREMENT_ONLY) to only enforce device-wide password requirement. In all other cases the behaviour is identical to `ACTION_SET_NEW_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PASSWORD).

Constant Value: "android.app.action.SET_NEW_PARENT_PROFILE_PASSWORD"

ACTION_SET_NEW_PASSWORD

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_SET_NEW_PASSWORD
```

Activity action: have the user enter a new password.

For admin apps, this activity should be launched after using `setPasswordQuality(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) , or `setPasswordMinimumLength(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumLength(android.content.ComponentName,%20int)) to have the user enter a new password that meets the current requirements. You can use `isActivePasswordSufficient()` (/reference/android/app/admin/DevicePolicyManager#isActivePasswordSufficient()) to determine whether you need to have the user select a new password in order to meet the current constraints. Upon being resumed from this activity, you can check the new password characteristics to see if they are sufficient.

Non-admin apps can use `getPasswordComplexity()` (/reference/android/app/admin/DevicePolicyManager#getPasswordComplexity()) to check the current screen lock complexity, and use this activity with extra `EXTRA_PASSWORD_COMPLEXITY` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PASSWORD_COMPLEXITY) to suggest to users how complex the app wants the new screen lock to be. Note that both `getPasswordComplexity()` (/reference/android/app/admin/DevicePolicyManager#getPasswordComplexity()) and the extra `EXTRA_PASSWORD_COMPLEXITY` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PASSWORD_COMPLEXITY) require the calling app to have the permission `permission#REQUEST_PASSWORD_COMPLEXITY` (/reference/android/Manifest.permission#REQUEST_PASSWORD_COMPLEXITY).

If the intent is launched from within a managed profile with a profile owner built against `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M) or before, this will trigger entering a new password for the parent of the profile. For all other cases it will trigger entering a new password for the user or profile it is launched from.

See also:

`ACTION_SET_NEW_PARENT_PROFILE_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PARENT_PROFILE_PASSWORD)

Constant Value: "android.app.action.SET_NEW_PASSWORD"

ACTION_START_ENCRYPTION

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_START_ENCRYPTION
```

Activity action: begin the process of encrypting data on the device. This activity should be launched after using `setStorageEncryption(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setStorageEncryption(android.content.ComponentName,%20boolean)) to request encryption be activated. After resuming from this activity, use `getStorageEncryption(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getStorageEncryption(android.content.ComponentName)) to check encryption status. However, on some devices this activity may never return, as it may trigger a reboot and in some cases a complete data wipe of the device.

Constant Value: "android.app.action.START_ENCRYPTION"

ACTION_SYSTEM_UPDATE_POLICY_CHANGED

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) ACTION_SYSTEM_UPDATE_POLICY_CHANG
```

Broadcast action: notify that a new local system update policy has been set by the device owner. The new policy can be retrieved by `getSystemUpdatePolicy()` (/reference/android/app/admin/DevicePolicyManager#getSystemUpdatePolicy()).

Constant Value: "android.app.action.SYSTEM_UPDATE_POLICY_CHANGED"

CONTENT_PROTECTION_DISABLED

Added in Android VanillaIceCream (/preview)

```
public static final int CONTENT_PROTECTION_DISABLED
```

Indicates that content protection is controlled and disabled by a policy (default).

Constant Value: 1 (0x00000001)

CONTENT_PROTECTION_ENABLED

Added in Android VanillaIceCream (/preview)

```
public static final int CONTENT_PROTECTION_ENABLED
```

Indicates that content protection is controlled and enabled by a policy.

Constant Value: 2 (0x00000002)

CONTENT_PROTECTION_NOT_CONTROLLED_BY_POLICY

Added in Android VanillaIceCream (/preview)

```
public static final int CONTENT_PROTECTION_NOT_CONTROLLED_BY_POLICY
```

Indicates that content protection is not controlled by policy, allowing user to choose.

Constant Value: 0 (0x00000000)

DELEGATION_APP_RESTRICTIONS

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) DELEGATION_APP_RESTRICTIONS
```

Delegation of application restrictions management. This scope grants access to the `setApplicationRestrictions(ComponentName, String, Bundle)` (/reference/android/app/admin/DevicePolicyManager#setApplicationRestrictions(android.content.ComponentName,%20java.lang.String,%20android.os.Bundle))

and `getApplicationRestrictions(ComponentName, String)` (/reference/android/app/admin/DevicePolicyManager#getApplicationRestrictions(android.content.ComponentName,%20java.lang.String)) APIs.

Constant Value: "delegation-app-restrictions"

DELEGATION_BLOCK_UNINSTALL

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) DELEGATION_BLOCK_UNINSTALL
```

Delegation of application uninstall block. This scope grants access to the `setUninstallBlocked(ComponentName, String, boolean)` (/reference/android/app/admin/DevicePolicyManager#setUninstallBlocked(android.content.ComponentName,%20java.lang.String,%20boolean)) API.

Constant Value: "delegation-block-uninstall"

DELEGATION_CERT_INSTALL

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) DELEGATION_CERT_INSTALL
```

Delegation of certificate installation and management. This scope grants access to the `getInstalledCaCerts(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getInstalledCaCerts(android.content.ComponentName)), `hasCaCertInstalled(ComponentName, byte)` (/reference/android/app/admin/DevicePolicyManager#hasCaCertInstalled(android.content.ComponentName,%20byte[])), `installCaCert(ComponentName, byte)` (/reference/android/app/admin/DevicePolicyManager#installCaCert(android.content.ComponentName,%20byte[])), `uninstallCaCert(ComponentName, byte)` (/reference/android/app/admin/DevicePolicyManager#uninstallCaCert(android.content.ComponentName,%20byte[]))

, `uninstallAllUserCaCerts(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#uninstallAllUserCaCerts(android.content.ComponentName)) and `installKeyPair(ComponentName, PrivateKey, Certificate, String)` (/reference/android/app/admin/DevicePolicyManager#installKeyPair(android.content.ComponentName,%20java.security.PrivateKey,%20java.security.cert.Certificate,%20java.lang.String)) APIs. This scope also grants the ability to read identifiers that the delegating device owner or profile owner can obtain. See `getEnrollmentSpecificId()` (/reference/android/app/admin/DevicePolicyManager#getEnrollmentSpecificId()).

Constant Value: "delegation-cert-install"

DELEGATION_CERT_SELECTION

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) DELEGATION_CERT_SELECTION
```

Grants access to selection of KeyChain certificates on behalf of requesting apps. Once granted the app will start receiving `DelegatedAdminReceiver#onChoosePrivateKeyAlias` (/reference/android/app/admin/DelegatedAdminReceiver#onChoosePrivateKeyAlias(android.content.Context,%20android.content.Intent,%20int,%20android.net.Uri,%20java.lang.String)). The caller (PO/DO) will no longer receive `DeviceAdminReceiver#onChoosePrivateKeyAlias` (/reference/android/app/admin/DeviceAdminReceiver#onChoosePrivateKeyAlias(android.content.Context,%20android.content.Intent,%20int,%20android.net.Uri,%20java.lang.String)). There can be at most one app that has this delegation. If another app already had delegated certificate selection access, it will lose the delegation when a new app is delegated.

The delegated app can also call `grantKeyPairToApp(ComponentName, String, String)` (/reference/android/app/admin/DevicePolicyManager#grantKeyPairToApp(android.content.ComponentName,%20java.lang.String,%20java.lang.String)) and `revokeKeyPairFromApp(ComponentName, String, String)` (/reference/android/app/admin/DevicePolicyManager#revokeKeyPairFromApp(android.content.ComponentName,%20java.lang.String,%20java.lang.String)) to directly grant KeyChain keys to other apps.

Can be granted by Device Owner or Profile Owner.

Constant Value: "delegation-cert-selection"

DELEGATION_ENABLE_SYSTEM_APP

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) DELEGATION_ENABLE_SYSTEM_APP
```

Delegation for enabling system apps. This scope grants access to the `enableSystemApp(ComponentName, Intent)` (/reference/android/app/admin/DevicePolicyManager#enableSystemApp(android.content.ComponentName,%20android.content.Intent)) API.

Constant Value: "delegation-enable-system-app"

DELEGATION_INSTALL_EXISTING_PACKAGE

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) DELEGATION_INSTALL_EXISTING_PACKA
```

Delegation for installing existing packages. This scope grants access to the `installExistingPackage(ComponentName, String)` (/reference/android/app/admin/DevicePolicyManager#installExistingPackage(android.content.ComponentName,%20java.lang.String)) API.

Constant Value: "delegation-install-existing-package"

DELEGATION_KEEP_UNINSTALLED_PACKAGES

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) DELEGATION_KEEP_UNINSTALLED_PACKA
```

Delegation of management of uninstalled packages. This scope grants access to the `setKeepUninstalledPackages(ComponentName, List)` (/reference/android/app/admin/DevicePolicyManager#setKeepUninstalledPackages(android.content.ComponentName,%20java.util.List<java.lang.String>)) and `getKeepUninstalledPackages(ComponentName)`

(/reference/android/app/admin/DevicePolicyManager#getKeepUninstalledPackages(android.content.ComponentName))
APIs.

Constant Value: "delegation-keep-uninstalled-packages"

DELEGATION_NETWORK_LOGGING

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) DELEGATION_NETWORK_LOGGING
```

Grants access to `setNetworkLoggingEnabled(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setNetworkLoggingEnabled(android.content.ComponentName,%20boolean)), `isNetworkLoggingEnabled(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#isNetworkLoggingEnabled(android.content.ComponentName)) and `retrieveNetworkLogs(ComponentName, long)` (/reference/android/app/admin/DevicePolicyManager#retrieveNetworkLogs(android.content.ComponentName,%20long)). Once granted the delegated app will start receiving DelegatedAdminReceiver.onNetworkLogsAvailable() callback, and Device owner or Profile Owner will no longer receive the DeviceAdminReceiver.onNetworkLogsAvailable() callback. There can be at most one app that has this delegation. If another app already had delegated network logging access, it will lose the delegation when a new app is delegated.

Device Owner can grant this access since Android 10. Profile Owner of a managed profile can grant this access since Android 12.

Constant Value: "delegation-network-logging"

DELEGATION_PACKAGE_ACCESS

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) DELEGATION_PACKAGE_ACCESS
```

Delegation of package access state. This scope grants access to the `isApplicationHidden(ComponentName, String)`

(/reference/android/app/admin/DevicePolicyManager#isApplicationHidden(android.content.ComponentName,%20java.lang.String))
, `setApplicationHidden(ComponentName, String, boolean)`
(/reference/android/app/admin/DevicePolicyManager#setApplicationHidden(android.content.ComponentName,%20java.lang.String,%20boolean))
, `isPackageSuspended(ComponentName, String)`
(/reference/android/app/admin/DevicePolicyManager#isPackageSuspended(android.content.ComponentName,%20java.lang.String))
, and `setPackagesSuspended(ComponentName, String, boolean)`
(/reference/android/app/admin/DevicePolicyManager#setPackagesSuspended(android.content.ComponentName,%20java.lang.String[],%20boolean))
APIs.

Constant Value: "delegation-package-access"

DELEGATION_PERMISSION_GRANT

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) DELEGATION_PERMISSION_GRANT
```

Delegation of permission policy and permission grant state. This scope grants access to the `setPermissionPolicy(ComponentName, int)`
(/reference/android/app/admin/DevicePolicyManager#setPermissionPolicy(android.content.ComponentName,%20int))
, `getPermissionGrantState(ComponentName, String, String)`
(/reference/android/app/admin/DevicePolicyManager#getPermissionGrantState(android.content.ComponentName,%20java.lang.String,%20java.lang.String))
, and `setPermissionGrantState(ComponentName, String, String, int)`
(/reference/android/app/admin/DevicePolicyManager#setPermissionGrantState(android.content.ComponentName,%20java.lang.String,%20java.lang.String,%20int))
APIs.

Constant Value: "delegation-permission-grant"

DELEGATION_SECURITY_LOGGING

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) DELEGATION_SECURITY_LOGGING
```

Grants access to `setSecurityLoggingEnabled(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setSecurityLoggingEnabled(android.content.ComponentName,%20boolean)), `isSecurityLoggingEnabled(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#isSecurityLoggingEnabled(android.content.ComponentName)), `retrieveSecurityLogs(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#retrieveSecurityLogs(android.content.ComponentName)), and `retrievePreRebootSecurityLogs(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#retrievePreRebootSecurityLogs(android.content.ComponentName)). Once granted the delegated app will start receiving `DelegatedAdminReceiver#onSecurityLogsAvailable` (/reference/android/app/admin/DelegatedAdminReceiver#onSecurityLogsAvailable(android.content.Context,%20android.content.Intent)) callback, and Device owner or Profile Owner will no longer receive the `DeviceAdminReceiver#onSecurityLogsAvailable` (/reference/android/app/admin/DeviceAdminReceiver#onSecurityLogsAvailable(android.content.Context,%20android.content.Intent)) callback. There can be at most one app that has this delegation. If another app already had delegated security logging access, it will lose the delegation when a new app is delegated.

Can only be granted by Device Owner or Profile Owner of an organization-owned managed profile.

Constant Value: "delegation-security-logging"

ENCRYPTION_STATUS_ACTIVATING

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int ENCRYPTION_STATUS_ACTIVATING
```

This constant was deprecated in API level 34.
This result code has never actually been used, so there is no reason for apps to check for it.

Result code for `getStorageEncryptionStatus()` (/reference/android/app/admin/DevicePolicyManager#getStorageEncryptionStatus()): indicating that

encryption is not currently active, but is currently being activated.

Constant Value: 2 (0x00000002)

ENCRYPTION_STATUS_ACTIVE

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int ENCRYPTION_STATUS_ACTIVE
```

Result code for `setStorageEncryption(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setStorageEncryption(android.content.ComponentName,%20boolean)) and `getStorageEncryptionStatus()` (/reference/android/app/admin/DevicePolicyManager#getStorageEncryptionStatus()): indicating that encryption is active.

`getStorageEncryptionStatus()` (/reference/android/app/admin/DevicePolicyManager#getStorageEncryptionStatus()) can only return this value for apps targeting API level 23 or lower, or on devices that use Full Disk Encryption. Support for Full Disk Encryption was entirely removed in API level 33, having been replaced by File Based Encryption. The result code `ENCRYPTION_STATUS_ACTIVE_PER_USER` (/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATUS_ACTIVE_PER_USER) is used on devices that use File Based Encryption, except when the app targets API level 23 or lower.

`setStorageEncryption(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setStorageEncryption(android.content.ComponentName,%20boolean)) can still return this value for an unrelated reason, but `setStorageEncryption(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setStorageEncryption(android.content.ComponentName,%20boolean)) is deprecated since it doesn't do anything useful.

Constant Value: 3 (0x00000003)

ENCRYPTION_STATUS_ACTIVE_DEFAULT_KEY

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int ENCRYPTION_STATUS_ACTIVE_DEFAULT_KEY
```

Result code for `getStorageEncryptionStatus()` (/reference/android/app/admin/DevicePolicyManager#getStorageEncryptionStatus()): indicating that encryption is active, but the encryption key is not cryptographically protected by the user's credentials.

This value can only be returned on devices that use Full Disk Encryption. Support for Full Disk Encryption was entirely removed in API level 33, having been replaced by File Based Encryption. With File Based Encryption, each user's credential-encrypted storage is always cryptographically protected by the user's credentials.

Constant Value: 4 (0x00000004)

ENCRYPTION_STATUS_ACTIVE_PER_USER

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int ENCRYPTION_STATUS_ACTIVE_PER_USER
```

Result code for `getStorageEncryptionStatus()` (/reference/android/app/admin/DevicePolicyManager#getStorageEncryptionStatus()): indicating that encryption is active and the encryption key is tied to the user or profile.

This value is only returned to apps targeting API level 24 and above. For apps targeting earlier API levels, `ENCRYPTION_STATUS_ACTIVE` (/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATUS_ACTIVE) is returned, even if the encryption key is specific to the user or profile.

Constant Value: 5 (0x00000005)

ENCRYPTION_STATUS_INACTIVE

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int ENCRYPTION_STATUS_INACTIVE
```

Result code for `setStorageEncryption(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setStorageEncryption(android.content.ComponentName,%20boolean)) and `getStorageEncryptionStatus()` (/reference/android/app/admin/DevicePolicyManager#getStorageEncryptionStatus()): indicating that encryption is supported, but is not currently active.

`getStorageEncryptionStatus()` (/reference/android/app/admin/DevicePolicyManager#getStorageEncryptionStatus()) can only return this value on devices that use Full Disk Encryption. Support for Full Disk Encryption was entirely removed in API level 33, having been replaced by File Based Encryption. Devices that use File Based Encryption always automatically activate their encryption on first boot.

`setStorageEncryption(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setStorageEncryption(android.content.ComponentName,%20boolean)) can still return this value for an unrelated reason, but `setStorageEncryption(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setStorageEncryption(android.content.ComponentName,%20boolean)) is deprecated since it doesn't do anything useful.

Constant Value: 1 (0x00000001)

ENCRYPTION_STATUS_UNSUPPORTED

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int ENCRYPTION_STATUS_UNSUPPORTED
```

Result code for `setStorageEncryption(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setStorageEncryption(android.content.ComponentName,%20boolean)) and `getStorageEncryptionStatus()` (/reference/android/app/admin/DevicePolicyManager#getStorageEncryptionStatus()): indicating that encryption is not supported.

Constant Value: 0 (0x00000000)

EXTRA_ADD_EXPLANATION

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_ADD_EXPLANATION
```

An optional CharSequence providing additional explanation for why the admin is being added.

See also:

`ACTION_ADD_DEVICE_ADMIN` (/reference/android/app/admin/DevicePolicyManager#ACTION_ADD_DEVICE_ADMIN)

Constant Value: "android.app.extra.ADD_EXPLANATION"

EXTRA_DELEGATION_SCOPES

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_DELEGATION_SCOPES
```

An `ArrayList<String>` corresponding to the delegation scopes given to an app in the `ACTION_APPLICATION_DELEGATION_SCOPES_CHANGED` (/reference/android/app/admin/DevicePolicyManager#ACTION_APPLICATION_DELEGATION_SCOPES_CHANGED) broadcast.

Constant Value: "android.app.extra.DELEGATION_SCOPES"

EXTRA_DEVICE_ADMIN

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_DEVICE_ADMIN
```

The ComponentName of the administrator component.

See also:

`ACTION_ADD_DEVICE_ADMIN` (/reference/android/app/admin/DevicePolicyManager#ACTION_ADD_DEVICE_ADMIN)

Constant Value: "android.app.extra.DEVICE_ADMIN"

EXTRA_DEVICE_PASSWORD_REQUIREMENT_ONLY

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_DEVICE_PASSWORD_REQUIREMENT
```

A boolean extra for `ACTION_SET_NEW_PARENT_PROFILE_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PARENT_PROFILE_PASSWORD) requesting that only device password requirement is enforced during the parent profile password enrolment flow.

Normally when enrolling password for the parent profile, both the device-wide password requirement (requirement set via `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) instance) and the profile password requirement are enforced, if the profile currently does not have a separate work challenge. By setting this to `true`, profile password requirement is explicitly disregarded.

See also:

`isActivePasswordSufficientForDeviceRequirement()` (/reference/android/app/admin/DevicePolicyManager#isActivePasswordSufficientForDeviceRequirement())

Constant Value: "android.app.extra.DEVICE_PASSWORD_REQUIREMENT_ONLY"

EXTRA_PASSWORD_COMPLEXITY

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PASSWORD_COMPLEXITY
```

An integer indicating the complexity level of the new password an app would like the user to set when launching the action `ACTION_SET_NEW_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PASSWORD).

Must be one of

- `PASSWORD_COMPLEXITY_HIGH` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPLEXITY_HIGH)
- `PASSWORD_COMPLEXITY_MEDIUM` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPLEXITY_MEDIUM)
- `PASSWORD_COMPLEXITY_LOW` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPLEXITY_LOW)
- `PASSWORD_COMPLEXITY_NONE` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPLEXITY_NONE)

If an invalid value is used, it will be treated as `PASSWORD_COMPLEXITY_NONE` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPLEXITY_NONE). Requires `Manifest.permission.REQUEST_PASSWORD_COMPLEXITY` (/reference/android/Manifest.permission#REQUEST_PASSWORD_COMPLEXITY)

Constant Value: "android.app.extra.PASSWORD_COMPLEXITY"

EXTRA_PROVISIONING_ACCOUNT_TO_MIGRATE

Added in API level 22 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_ACCOUNT_TO_MIG
```

An `Account` (/reference/android/accounts/Account) extra holding the account to migrate during managed profile provisioning. If the account supplied is present in the primary user, it will be copied, along with its credentials to the managed profile and removed from the primary user. Use with `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE), with managed account provisioning, or return as an extra to the intent result from the `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE) activity.

Constant Value: "android.app.extra.PROVISIONING_ACCOUNT_TO_MIGRATE"

EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_ADMIN_EXTRAS_B
```

A `Parcelable` (/reference/android/os/Parcelable) extra of type `PersistableBundle` (/reference/android/os/PersistableBundle) that allows a mobile device management application or NFC programmer application which starts managed provisioning to pass data to the management application instance after provisioning.

If used with `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE) it can be used by the application that sends the intent to pass data to itself on the newly created profile. If used with `ACTION_PROVISION_MANAGED_DEVICE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_DEVICE) it allows passing data to the same instance of the app on the primary user. Starting from `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M), if used with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) as part of NFC managed device provisioning, the NFC message should contain a stringified `Properties` (/reference/java/util/Properties) instance, whose string properties will be converted into a `PersistableBundle` (/reference/android/os/PersistableBundle) and passed to the management application after provisioning.

Admin apps will receive this extra in their `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE) and `ACTION_ADMIN_POLICY_COMPLIANCE` (/reference/android/app/admin/DevicePolicyManager#ACTION_ADMIN_POLICY_COMPLIANCE) intent handlers. Additionally, `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE) may also return this extra which will then be sent over to `ACTION_ADMIN_POLICY_COMPLIANCE` (/reference/android/app/admin/DevicePolicyManager#ACTION_ADMIN_POLICY_COMPLIANCE), alongside the original values that were passed to `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE).

In both cases the application receives the data in `DeviceAdminReceiver#onProfileProvisioningComplete` (/reference/android/app/admin/DeviceAdminReceiver#onProfileProvisioningComplete(android.content.Context,%20android.content.Intent)) via an intent with the action `DeviceAdminReceiver#ACTION_PROFILE_PROVISIONING_COMPLETE`

(/reference/android/app/admin/DeviceAdminReceiver#ACTION_PROFILE_PROVISIONING_COMPLETE). The bundle is not changed during the managed provisioning.

Constant Value: "android.app.extra.PROVISIONING_ADMIN_EXTRAS_BUNDLE"

EXTRA_PROVISIONING_ALLOWED_PROVISIONING_MODES

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_ALLOWED_PROVIS
```

An `ArrayList` (/reference/java/util/ArrayList) of `Integer` (/reference/java/lang/Integer) extra specifying the allowed provisioning modes.

This extra will be passed to the admin app's `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE) activity, whose result intent must contain `EXTRA_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_MODE) set to one of the values in this array.

If the value set to `EXTRA_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_MODE) is not in the array, provisioning will fail.

See also:

`PROVISIONING_MODE_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#PROVISIONING_MODE_MANAGED_PROFILE)

`PROVISIONING_MODE_FULLY_MANAGED_DEVICE` (/reference/android/app/admin/DevicePolicyManager#PROVISIONING_MODE_FULLY_MANAGED_DEVICE)

Constant Value: "android.app.extra.PROVISIONING_ALLOWED_PROVISIONING_MODES"

EXTRA_PROVISIONING_ALLOW_OFFLINE

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_ALLOW_OFFLINE
```

A boolean extra indicating whether offline provisioning is allowed.

For the online provisioning flow, there will be an attempt to download and install the latest version of the device policy management role holder. The platform will then delegate provisioning to the device policy management role holder via role holder-specific provisioning actions.

For the offline provisioning flow, the provisioning flow will always be handled by the platform.

If this extra is set to `false`, the provisioning flow will enforce that an internet connection is established, which will start the online provisioning flow. If an internet connection cannot be established, provisioning will fail.

If this extra is set to `true`, the provisioning flow will still try to connect to the internet, but if it fails it will start the offline provisioning flow.

For T if this extra is set to `true`, the provisioning flow will be forced through the platform and there will be no attempt to download and install the device policy management role holder.

The default value is `false`.

This extra is respected when provided via the provisioning intent actions such as `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE).

Constant Value: "android.app.extra.PROVISIONING_ALLOW_OFFLINE"

EXTRA_PROVISIONING_DEVICE_ADMIN_COMPONENT_NAME

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_DEVICE_ADMIN_C
```

A ComponentName extra indicating the device admin receiver of the mobile device management application that will be set as the profile owner or device owner and active admin.

If an application starts provisioning directly via an intent with action `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE) or `ACTION_PROVISION_MANAGED_DEVICE`

(/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_DEVICE) the package name of this component has to match the package name of the application that started provisioning.

This component is set as device owner and active admin when device owner provisioning is started by an intent with action `ACTION_PROVISION_MANAGED_DEVICE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_DEVICE) or by an NFC message containing an NFC record with MIME type `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC). For the NFC record, the component name must be flattened to a string, via `ComponentName#flattenToShortString()` (/reference/android/content/ComponentName#flattenToShortString()).

See also:

`DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver)

Constant Value: "android.app.extra.PROVISIONING_DEVICE_ADMIN_COMPONENT_NAME"

EXTRA_PROVISIONING_DEVICE_ADMIN_MINIMUM_VERSION_CODE

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_DEVICE_ADMIN_M
```

An int extra holding a minimum required version code for the device admin package. If the device admin is already installed on the device, it will only be re-downloaded from `EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_LOCATION` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_LOCATION) if the version of the installed package is less than this version code.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_DEVICE_ADMIN_MINIMUM_VERSION_CODE"

EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_CHECKSUM

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_DEVICE_ADMIN_P
```

A String extra holding the URL-safe base64 encoded SHA-256 hash of the file at download location specified in `EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_LOCATION` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_LOCATION)

.

Either this extra or `EXTRA_PROVISIONING_DEVICE_ADMIN_SIGNATURE_CHECKSUM` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_SIGNATURE_CHECKSUM)
must be present. The provided checksum must match the checksum of the file at the download location. If the checksum doesn't match an error will be shown to the user and the user will be asked to factory reset the device.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Note: for devices running `Build.VERSION_CODES.LOLLIPOP` (/reference/android/os/Build.VERSION_CODES#LOLLIPOP) and `Build.VERSION_CODES.LOLLIPOP_MR1` (/reference/android/os/Build.VERSION_CODES#LOLLIPOP_MR1) only SHA-1 hash is supported. Starting from `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M), this parameter accepts SHA-256 in addition to SHA-1. From `Build.VERSION_CODES.Q` (/reference/android/os/Build.VERSION_CODES#Q), only SHA-256 hash is supported.

Constant Value: "android.app.extra.PROVISIONING_DEVICE_ADMIN_PACKAGE_CHECKSUM"

EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_COOKIE_HEADER

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_DEVICE_ADMIN_P
```

A String extra holding a http cookie header which should be used in the http request to the url specified in `EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_LOCATION` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_LOCATION)

.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value:
"android.app.extra.PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_COOKIE_HEADER"

EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_LOCATION

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_DEVICE_ADMIN_P
```

A String extra holding a url that specifies the download location of the device admin package. When not provided it is assumed that the device admin package is already installed.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value:
"android.app.extra.PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_LOCATION"

EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_NAME

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_DEVICE_ADMIN_P
```

This constant was deprecated in API level 23.
Use `EXTRA_PROVISIONING_DEVICE_ADMIN_COMPONENT_NAME`

(/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_COMPONENT_NAME)
. This extra is still supported, but only if there is only one device admin receiver in the package that requires the permission **`Manifest.permission.BIND_DEVICE_ADMIN`** (/reference/android/Manifest.permission#BIND_DEVICE_ADMIN).

A String extra holding the package name of the mobile device management application that will be set as the profile owner or device owner.

If an application starts provisioning directly via an intent with action `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE) this package has to match the package name of the application that started provisioning. The package will be set as profile owner in that case.

This package is set as device owner when device owner provisioning is started by an NFC message containing an NFC record with MIME type `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC).

When this extra is set, the application must have exactly one device admin receiver. This receiver will be set as the profile or device owner and active admin.

See also:

`DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver)

Constant Value: "android.app.extra.PROVISIONING_DEVICE_ADMIN_PACKAGE_NAME"

EXTRA_PROVISIONING_DEVICE_ADMIN_SIGNATURE_CHECKSUM

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_DEVICE_ADMIN_S
```

A String extra holding the URL-safe base64 encoded SHA-256 checksum of any signature of the android package archive at the download location specified in `EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_LOCATION`

(/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_LOCATION)

.

The signatures of an android package archive can be obtained using `PackageManager.getPackageArchiveInfo(String, PackageInfoFlags)` (/reference/android/content/pm/PackageManager#getPackageArchiveInfo(java.lang.String,%20android.content.pm.PackageManager.PackageInfoFlags)) with flag `PackageManager.GET_SIGNATURES` (/reference/android/content/pm/PackageManager#GET_SIGNATURES).

Either this extra or `EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_CHECKSUM` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_CHECKSUM) must be present. The provided checksum must match the checksum of any signature of the file at the download location. If the checksum does not match an error will be shown to the user and the user will be asked to factory reset the device.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_DEVICE_ADMIN_SIGNATURE_CHECKSUM"

EXTRA_PROVISIONING_DISCLAIMERS

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_DISCLAIMERS
```

A `Bundle` (/reference/android/os/Bundle)[] extra consisting of list of disclaimer headers and disclaimer contents. Each `Bundle` (/reference/android/os/Bundle) must have both `EXTRA_PROVISIONING_DISCLAIMER_HEADER` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DISCLAIMER_HEADER) as disclaimer header, and `EXTRA_PROVISIONING_DISCLAIMER_CONTENT` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DISCLAIMER_CONTENT) as disclaimer content.

The extra typically contains one disclaimer from the company of mobile device management application (MDM), and one disclaimer from the organization.

Call `Bundle#putParcelableArray(String, Parcelable[])` (/reference/android/os/Bundle#putParcelableArray(java.lang.String,%20android.os.Parcelable[])) to put the `Bundle` (/reference/android/os/Bundle)[]

Maximum 3 key-value pairs can be specified. The rest will be ignored.

Can be used in an intent with action `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE). This extra can also be returned by the admin app when performing the admin-integrated provisioning flow as a result of the `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE) activity.

Constant Value: "android.app.extra.PROVISIONING_DISCLAIMERS"

EXTRA_PROVISIONING_DISCLAIMER_CONTENT

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_DISCLAIMER_CON
```

A `Uri` (/reference/android/net/Uri) extra pointing to disclaimer content.

The following URI schemes are accepted:

- content (`ContentResolver.SCHEME_CONTENT` (/reference/android/content/ContentResolver#SCHEME_CONTENT))
- android.resource (`ContentResolver.SCHEME_ANDROID_RESOURCE` (/reference/android/content/ContentResolver#SCHEME_ANDROID_RESOURCE))

Styled text is supported in the disclaimer content. The content is parsed by `Html.fromHtml(String)` (/reference/android/text/Html#fromHtml(java.lang.String)) and displayed in a `TextView` (/reference/android/widget/TextView).

If a `content:` URI is passed, URI is passed, the intent should have the flag `Intent#FLAG_GRANT_READ_URI_PERMISSION` (/reference/android/content/Intent#FLAG_GRANT_READ_URI_PERMISSION) and the uri should be added to the `ClipData` (/reference/android/content/ClipData) of the intent too.

Use in Bundle `EXTRA_PROVISIONING_DISCLAIMERS` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DISCLAIMERS)

System app, i.e. application with `ApplicationInfo#FLAG_SYSTEM` (/reference/android/content/pm/ApplicationInfo#FLAG_SYSTEM), can also insert a disclaimer by declaring an application-level meta-data in `AndroidManifest.xml`. Must use it with `EXTRA_PROVISIONING_DISCLAIMER_HEADER` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DISCLAIMER_HEADER). Here is the example:

```
 <meta-data
     android:name="android.app.extra.PROVISIONING_DISCLAIMER_CONTENT"
     android:resource="@string/disclaimer_content"
/>
```

Constant Value: "android.app.extra.PROVISIONING_DISCLAIMER_CONTENT"

EXTRA_PROVISIONING_DISCLAIMER_HEADER

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_DISCLAIMER_HEA
```

A String extra of localized disclaimer header.

The extra is typically the company name of mobile device management application (MDM) or the organization name.

Use in Bundle `EXTRA_PROVISIONING_DISCLAIMERS` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DISCLAIMERS)

System app, i.e. application with `ApplicationInfo#FLAG_SYSTEM` (/reference/android/content/pm/ApplicationInfo#FLAG_SYSTEM), can also insert a disclaimer by declaring an application-level meta-data in `AndroidManifest.xml`. Must use it with `EXTRA_PROVISIONING_DISCLAIMER_CONTENT` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DISCLAIMER_CONTENT). Here is the example:

```
 <meta-data
     android:name="android.app.extra.PROVISIONING_DISCLAIMER_HEADER"
     android:resource="@string/disclaimer_header"
/>
```

Constant Value: "android.app.extra.PROVISIONING_DISCLAIMER_HEADER"

EXTRA_PROVISIONING_EMAIL_ADDRESS

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_EMAIL_ADDRESS
```

This constant was deprecated in API level 26.
From **`Build.VERSION_CODES.O`** (/reference/android/os/Build.VERSION_CODES#O), never used while provisioning the device.

Constant Value: "android.app.extra.PROVISIONING_EMAIL_ADDRESS"

EXTRA_PROVISIONING_IMEI

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_IMEI
```

A string extra holding the IMEI (International Mobile Equipment Identity) of the device.

Constant Value: "android.app.extra.PROVISIONING_IMEI"

EXTRA_PROVISIONING_KEEP_ACCOUNT_ON_MIGRATION

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_KEEP_ACCOUNT_O
```

Boolean extra to indicate that the migrated account should be kept. This is used in conjunction with `EXTRA_PROVISIONING_ACCOUNT_TO_MIGRATE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_ACCOUNT_TO_MIGRATE). If it's set to `true`, the account will not be removed from the primary user after it is migrated to the newly created user or profile.

Defaults to `false`

Use with `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE) or set as an extra to the intent result of the `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE) activity.

See also:

`EXTRA_PROVISIONING_ACCOUNT_TO_MIGRATE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_ACCOUNT_TO_MIGRATE)

Constant Value: "android.app.extra.PROVISIONING_KEEP_ACCOUNT_ON_MIGRATION"

EXTRA_PROVISIONING_KEEP_SCREEN_ON

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_KEEP_SCREEN_ON
```

This constant was deprecated in API level 34.
from `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), the flag wouldn't be functional. The screen is kept on throughout the provisioning flow.

A `boolean` flag that indicates whether the screen should be on throughout the provisioning flow.

This extra can either be passed as an extra to the `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE) intent, or it can be returned by the admin app when performing the admin-integrated provisioning flow

as a result of the `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE) activity.

Constant Value: "android.app.extra.PROVISIONING_KEEP_SCREEN_ON"

EXTRA_PROVISIONING_LEAVE_ALL_SYSTEM_APPS_ENABLED

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_LEAVE_ALL_SYST
```

A Boolean extra that can be used by the mobile device management application to skip the disabling of system apps during provisioning when set to `true`.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC), an intent with action `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE) that starts profile owner provisioning or set as an extra to the intent result of the `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE) activity.

Constant Value: "android.app.extra.PROVISIONING_LEAVE_ALL_SYSTEM_APPS_ENABLED"

EXTRA_PROVISIONING_LOCALE

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_LOCALE
```

A String extra holding the `Locale` (/reference/java/util/Locale) that the device will be set to. Format: xx_yy, where xx is the language code, and yy the country code.

Use only for device owner provisioning. This extra can be returned by the admin app when performing the admin-integrated provisioning flow as a result of the `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE) activity.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_LOCALE"

EXTRA_PROVISIONING_LOCAL_TIME

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_LOCAL_TIME
```

A Long extra holding the wall clock time (in milliseconds) to be set on the device's `AlarmManager` (/reference/android/app/AlarmManager).

Use only for device owner provisioning. This extra can be returned by the admin app when performing the admin-integrated provisioning flow as a result of the `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE) activity.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_LOCAL_TIME"

EXTRA_PROVISIONING_LOGO_URI

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_LOGO_URI
```

This constant was deprecated in API level 33.
Logo customization is no longer supported in the provisioning flow.

A `Uri` (/reference/android/net/Uri) extra pointing to a logo image. This image will be shown during the provisioning. If this extra is not passed, a default image will be shown.

The following URI schemes are accepted:

- content (`ContentResolver.SCHEME_CONTENT` (/reference/android/content/ContentResolver#SCHEME_CONTENT))
- android.resource (`ContentResolver.SCHEME_ANDROID_RESOURCE` (/reference/android/content/ContentResolver#SCHEME_ANDROID_RESOURCE))

It is the responsibility of the caller to provide an image with a reasonable pixel density for the device.

If a content: URI is passed, the intent should have the flag `Intent#FLAG_GRANT_READ_URI_PERMISSION` (/reference/android/content/Intent#FLAG_GRANT_READ_URI_PERMISSION) and the uri should be added to the `ClipData` (/reference/android/content/ClipData) of the intent too.

Use in an intent with action `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE) or `ACTION_PROVISION_MANAGED_DEVICE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_DEVICE)

Constant Value: "android.app.extra.PROVISIONING_LOGO_URI"

EXTRA_PROVISIONING_MAIN_COLOR

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_MAIN_COLOR
```

This constant was deprecated in API level 31.
Color customization is no longer supported in the provisioning flow.

A integer extra indicating the predominant color to show during the provisioning. Refer to `Color` (/reference/android/graphics/Color) for how the color is represented.

Use with `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE) or `ACTION_PROVISION_MANAGED_DEVICE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_DEVICE).

Constant Value: "android.app.extra.PROVISIONING_MAIN_COLOR"

EXTRA_PROVISIONING_MODE

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_MODE
```

An intent extra holding the provisioning mode returned by the administrator. The value of this extra must be one of the values provided in `EXTRA_PROVISIONING_ALLOWED_PROVISIONING_MODES` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_ALLOWED_PROVISIONING_MODES), which is provided as an intent extra to the admin app's `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE) activity.

See also:

`PROVISIONING_MODE_FULLY_MANAGED_DEVICE` (/reference/android/app/admin/DevicePolicyManager#PROVISIONING_MODE_FULLY_MANAGED_DEVICE)

`PROVISIONING_MODE_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#PROVISIONING_MODE_MANAGED_PROFILE)

Constant Value: "android.app.extra.PROVISIONING_MODE"

EXTRA_PROVISIONING_SENSORS_PERMISSION_GRANT_OPT_OUT

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_SENSORS_PERMIS
```

A boolean extra indicating the admin of a fully-managed device opts out of controlling permission grants for sensor-related permissions, see `setPermissionGrantState(android.content.ComponentName, java.lang.String, java.lang.String, int)` (/reference/android/app/admin/DevicePolicyManager#setPermissionGrantState(android.content.ComponentName,%20java.lang.String,%20java.lang.String,%20int)). The default for this extra is `false` - by default, the admin of a fully-managed device has the ability to grant sensors-related permissions.

Use only for device owner provisioning. This extra can be returned by the admin app when performing the admin-integrated provisioning flow as a result of the `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE) activity.

This extra may also be provided to the admin app via an intent extra for `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE).

See also:

`ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE)

Constant Value: "android.app.extra.PROVISIONING_SENSORS_PERMISSION_GRANT_OPT_OUT"

EXTRA_PROVISIONING_SERIAL_NUMBER

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_SERIAL_NUMBER
```

A string extra holding the serial number of the device.

Constant Value: "android.app.extra.PROVISIONING_SERIAL_NUMBER"

EXTRA_PROVISIONING_SHOULD_LAUNCH_RESULT_INTENT

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_SHOULD_LAUNCH_
```

A boolean extra that determines whether the provisioning flow should launch the resulting launch intent, if one is supplied by the device policy management role holder via `EXTRA_RESULT_LAUNCH_INTENT` (/reference/android/app/admin/DevicePolicyManager#EXTRA_RESULT_LAUNCH_INTENT). Default value is `false`.

If `true`, the resulting intent will be launched by the provisioning flow, if one is supplied by the device policy management role holder.

If `false`, the resulting intent will be returned as `EXTRA_RESULT_LAUNCH_INTENT` (/reference/android/app/admin/DevicePolicyManager#EXTRA_RESULT_LAUNCH_INTENT) to the provisioning initiator, if one is supplied by the device manager role holder. It will be the responsibility of the provisioning initiator to launch this `Intent` (/reference/android/content/Intent) after provisioning completes.

This extra is respected when provided via the provisioning intent actions such as `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE).

Constant Value: "android.app.extra.PROVISIONING_SHOULD_LAUNCH_RESULT_INTENT"

EXTRA_PROVISIONING_SKIP_EDUCATION_SCREENS

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_SKIP_EDUCATION
```

A boolean extra indicating if the education screens from the provisioning flow should be skipped. If unspecified, defaults to `false`.

This extra can be set in the following ways:

- By the admin app when performing the admin-integrated provisioning flow as a result of the `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE) activity
- For managed account enrollment

If the education screens are skipped, it is the admin application's responsibility to display its own user education screens.

Constant Value: "android.app.extra.PROVISIONING_SKIP_EDUCATION_SCREENS"

EXTRA_PROVISIONING_SKIP_ENCRYPTION

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_SKIP_ENCRYPTIO
```

A boolean extra indicating whether device encryption can be skipped as part of device owner or managed profile provisioning.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) or an intent with action `ACTION_PROVISION_MANAGED_DEVICE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_DEVICE) that starts device owner provisioning.

From `Build.VERSION_CODES.N` (/reference/android/os/Build.VERSION_CODES#N) onwards, this is also supported for an intent with action `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE).

This extra can also be returned by the admin app when performing the admin-integrated provisioning flow as a result of the `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE) activity.

Constant Value: "android.app.extra.PROVISIONING_SKIP_ENCRYPTION"

EXTRA_PROVISIONING_SKIP_USER_CONSENT

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_SKIP_USER_CONS
```

This constant was deprecated in API level 31.
this extra is no longer relevant as device owners cannot create managed profiles

A boolean extra indicating if the user consent steps from the provisioning flow should be skipped. If unspecified, defaults to `false`. It can only be used by an existing device owner trying to create a managed profile via `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE). Otherwise it is ignored.

Constant Value: "android.app.extra.PROVISIONING_SKIP_USER_CONSENT"

EXTRA_PROVISIONING_TIME_ZONE

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_TIME_ZONE
```

A String extra holding the time zone `AlarmManager` (/reference/android/app/AlarmManager) that the device will be set to.

Use only for device owner provisioning. This extra can be returned by the admin app when performing the admin-integrated provisioning flow as a result of the `ACTION_GET_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#ACTION_GET_PROVISIONING_MODE) activity.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_TIME_ZONE"

EXTRA_PROVISIONING_USE_MOBILE_DATA

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_USE_MOBILE_DAT
```

A boolean extra indicating if mobile data should be used during the provisioning flow for downloading the admin app. If `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID) is also specified, wifi network will be used instead.

Default value is `false`.

If this extra is set to `true` and `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID) is not specified, this extra has different behaviour depending on the way provisioning is triggered:

- For provisioning started via a QR code or an NFC tag, mobile data is always used for downloading the admin app.
- For all other provisioning methods, a mobile data connection check is made at the start of provisioning. If mobile data is connected at that point, the admin app download will

happen using mobile data. If mobile data is not connected at that point, the end-user will be asked to pick a wifi network and the admin app download will proceed over wifi.

Constant Value: "android.app.extra.PROVISIONING_USE_MOBILE_DATA"

EXTRA_PROVISIONING_WIFI_ANONYMOUS_IDENTITY

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_WIFI_ANONYMOUS
```

The anonymous identity of the wifi network in `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID). This is only used if the `EXTRA_PROVISIONING_WIFI_SECURITY_TYPE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SECURITY_TYPE) is `EAP`.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_WIFI_ANONYMOUS_IDENTITY"

EXTRA_PROVISIONING_WIFI_CA_CERTIFICATE

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_WIFI_CA_CERTIF
```

The CA certificate of the wifi network in `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID). This should be an X.509 certificate Base64 encoded DER format, ie. PEM representation of a certificate without header, footer and line breaks. More information (https://tools.ietf.org/html/rfc7468) This is only used if the `EXTRA_PROVISIONING_WIFI_SECURITY_TYPE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SECURITY_TYPE) is `EAP`.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts

device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_WIFI_CA_CERTIFICATE"

EXTRA_PROVISIONING_WIFI_DOMAIN

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_WIFI_DOMAIN
```

The domain of the wifi network in `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID). This is only used if the `EXTRA_PROVISIONING_WIFI_SECURITY_TYPE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SECURITY_TYPE) is `EAP`.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_WIFI_DOMAIN"

EXTRA_PROVISIONING_WIFI_EAP_METHOD

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_WIFI_EAP_METHO
```

The EAP method of the wifi network in `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID) and could be one of `PEAP`, `TLS`, `TTLS`, `PWD`, `SIM`, `AKA` or `AKA_PRIME`. This is only used if the `EXTRA_PROVISIONING_WIFI_SECURITY_TYPE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SECURITY_TYPE) is `EAP`.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_WIFI_EAP_METHOD"

EXTRA_PROVISIONING_WIFI_HIDDEN

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_WIFI_HIDDEN
```

A boolean extra indicating whether the wifi network in `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID) is hidden or not.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_WIFI_HIDDEN"

EXTRA_PROVISIONING_WIFI_IDENTITY

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_WIFI_IDENTITY
```

The identity of the wifi network in `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID). This is only used if the `EXTRA_PROVISIONING_WIFI_SECURITY_TYPE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SECURITY_TYPE) is `EAP`.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_WIFI_IDENTITY"

EXTRA_PROVISIONING_WIFI_PAC_URL

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_WIFI_PAC_URL
```

A String extra holding the proxy auto-config (PAC) URL for the wifi network in `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID).

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_WIFI_PAC_URL"

EXTRA_PROVISIONING_WIFI_PASSWORD

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_WIFI_PASSWORD
```

A String extra holding the password of the wifi network in `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID).

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_WIFI_PASSWORD"

EXTRA_PROVISIONING_WIFI_PHASE2_AUTH

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_WIFI_PHASE2_AU
```

The phase 2 authentication of the wifi network in `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID) and could be one of `NONE`, `PAP`, `MSCHAP`, `MSCHAPV2`, `GTC`, `SIM`, `AKA` or `AKA_PRIME`. This is only used if the `EXTRA_PROVISIONING_WIFI_SECURITY_TYPE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SECURITY_TYPE) is `EAP`.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_WIFI_PHASE2_AUTH"

EXTRA_PROVISIONING_WIFI_PROXY_BYPASS

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_WIFI_PROXY_BYP
```

A String extra holding the proxy bypass for the wifi network in `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID).

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_WIFI_PROXY_BYPASS"

EXTRA_PROVISIONING_WIFI_PROXY_HOST

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_WIFI_PROXY_HOS
```

A String extra holding the proxy host for the wifi network in `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID).

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_WIFI_PROXY_HOST"

EXTRA_PROVISIONING_WIFI_PROXY_PORT

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_WIFI_PROXY_POR
```

An int extra holding the proxy port for the wifi network in `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID).

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_WIFI_PROXY_PORT"

EXTRA_PROVISIONING_WIFI_SECURITY_TYPE

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_WIFI_SECURITY_
```

A String extra indicating the security type of the wifi network in `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID) and could be one of `NONE`, `WPA`, `WEP` or `EAP`.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_WIFI_SECURITY_TYPE"

EXTRA_PROVISIONING_WIFI_SSID

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_WIFI_SSID
```

A String extra holding the ssid of the wifi network that should be used during nfc device owner provisioning for downloading the mobile device management application.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_WIFI_SSID"

EXTRA_PROVISIONING_WIFI_USER_CERTIFICATE

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_PROVISIONING_WIFI_USER_CERT
```

The user certificate of the wifi network in `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID). This should be an X.509 certificate and private key Base64 encoded DER format, ie. PEM representation of a certificate and key without header, footer and line breaks. More information (https://tools.ietf.org/html/rfc7468) This is only used if the `EXTRA_PROVISIONING_WIFI_SECURITY_TYPE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SECURITY_TYPE) is `EAP`.

Use in an NFC record with `MIME_TYPE_PROVISIONING_NFC` (/reference/android/app/admin/DevicePolicyManager#MIME_TYPE_PROVISIONING_NFC) that starts device owner provisioning via an NFC bump. It can also be used for QR code provisioning.

Constant Value: "android.app.extra.PROVISIONING_WIFI_USER_CERTIFICATE"

EXTRA_RESOURCE_IDS

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_RESOURCE_IDS
```

An integer array extra for `ACTION_DEVICE_POLICY_RESOURCE_UPDATED` (/reference/android/app/admin/DevicePolicyManager#ACTION_DEVICE_POLICY_RESOURCE_UPDATED) to indicate which resource IDs (i.e. strings and drawables) have been updated.

Constant Value: "android.app.extra.RESOURCE_IDS"

EXTRA_RESOURCE_TYPE

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_RESOURCE_TYPE
```

An `int` extra for `ACTION_DEVICE_POLICY_RESOURCE_UPDATED` (/reference/android/app/admin/DevicePolicyManager#ACTION_DEVICE_POLICY_RESOURCE_UPDATED) to indicate the type of the resource being updated, the type can be `EXTRA_RESOURCE_TYPE_DRAWABLE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_RESOURCE_TYPE_DRAWABLE) or `EXTRA_RESOURCE_TYPE_STRING` (/reference/android/app/admin/DevicePolicyManager#EXTRA_RESOURCE_TYPE_STRING)

Constant Value: "android.app.extra.RESOURCE_TYPE"

EXTRA_RESOURCE_TYPE_DRAWABLE

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int EXTRA_RESOURCE_TYPE_DRAWABLE
```

A `int` value for `EXTRA_RESOURCE_TYPE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_RESOURCE_TYPE) to indicate that a resource of type `Drawable` (/reference/android/graphics/drawable/Drawable) is being updated.

Constant Value: 1 (0x00000001)

EXTRA_RESOURCE_TYPE_STRING

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int EXTRA_RESOURCE_TYPE_STRING
```

A `int` value for `EXTRA_RESOURCE_TYPE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_RESOURCE_TYPE) to indicate that a resource of type `String` (/reference/java/lang/String) is being updated.

Constant Value: 2 (0x00000002)

EXTRA_RESULT_LAUNCH_INTENT

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) EXTRA_RESULT_LAUNCH_INTENT
```

An `Intent` (/reference/android/content/Intent) result extra specifying the `Intent` (/reference/android/content/Intent) to be launched after provisioning is finalized.

If `EXTRA_PROVISIONING_SHOULD_LAUNCH_RESULT_INTENT` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_SHOULD_LAUNCH_RESULT_INTENT) is set to `false`, this result will be supplied as part of the result `Intent` (/reference/android/content/Intent) for provisioning actions such as `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE). This result will also be supplied as part of the result `Intent` (/reference/android/content/Intent) for the device policy management role holder provisioning actions.

Constant Value: "android.app.extra.RESULT_LAUNCH_INTENT"

FLAG_EVICT_CREDENTIAL_ENCRYPTION_KEY

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int FLAG_EVICT_CREDENTIAL_ENCRYPTION_KEY
```

Flag for `lockNow(int)` (/reference/android/app/admin/DevicePolicyManager#lockNow(int)): also evict the user's credential encryption key from the keyring. The user's credential will need to be entered again in order to derive the credential encryption key that will be stored back in the keyring for future use.

This flag can only be used by a profile owner when locking a managed profile when `getStorageEncryptionStatus()` (/reference/android/app/admin/DevicePolicyManager#getStorageEncryptionStatus()) returns `ENCRYPTION_STATUS_ACTIVE_PER_USER` (/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATUS_ACTIVE_PER_USER).

In order to secure user data, the user will be stopped and restarted so apps should wait until they are next run to perform further actions.

Constant Value: 1 (0x00000001)

FLAG_MANAGED_CAN_ACCESS_PARENT

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int FLAG_MANAGED_CAN_ACCESS_PARENT
```

Flag used by `addCrossProfileIntentFilter(ComponentName, IntentFilter, int)` (/reference/android/app/admin/DevicePolicyManager#addCrossProfileIntentFilter(android.content.ComponentName,%20android.content.IntentFilter,%20int)) to allow activities in the managed profile to access intents sent from the parent profile. That is, when an app in the parent profile calls `Activity#startActivity(Intent)` (/reference/android/app/Activity#startActivity(android.content.Intent)), the intent can be resolved by a matching activity in the managed profile.

Constant Value: 2 (0x00000002)

FLAG_PARENT_CAN_ACCESS_MANAGED

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int FLAG_PARENT_CAN_ACCESS_MANAGED
```

Flag used by `addCrossProfileIntentFilter(ComponentName, IntentFilter, int)` (/reference/android/app/admin/DevicePolicyManager#addCrossProfileIntentFilter(android.content.ComponentName,%20android.content.IntentFilter,%20int)) to allow activities in the parent profile to access intents sent from the managed profile. That is, when an app in the managed profile calls `Activity#startActivity(Intent)` (/reference/android/app/Activity#startActivity(android.content.Intent)), the intent can be resolved by a matching activity in the parent profile.

Constant Value: 1 (0x00000001)

ID_TYPE_BASE_INFO

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int ID_TYPE_BASE_INFO
```

Specifies that the device should attest its manufacturer details. For use with `generateKeyPair(ComponentName, String, KeyGenParameterSpec, int)` (/reference/android/app/admin/DevicePolicyManager#generateKeyPair(android.content.ComponentName,%20java.lang.String,%20android.security.keystore.KeyGenParameterSpec,%20int)).

See also:

`generateKeyPair(ComponentName, String, KeyGenParameterSpec, int)` (/reference/android/app/admin/DevicePolicyManager#generateKeyPair(android.content.ComponentName,%20java.lang.String,%20android.security.keystore.KeyGenParameterSpec,%20int))

Constant Value: 1 (0x00000001)

ID_TYPE_IMEI

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int ID_TYPE_IMEI
```

Specifies that the device should attest its IMEI. For use with `generateKeyPair(ComponentName, String, KeyGenParameterSpec, int)` (/reference/android/app/admin/DevicePolicyManager#generateKeyPair(android.content.ComponentName,%20java.lang.String,%20android.security.keystore.KeyGenParameterSpec,%20int)).

See also:

`generateKeyPair(ComponentName, String, KeyGenParameterSpec, int)` (/reference/android/app/admin/DevicePolicyManager#generateKeyPair(android.content.ComponentName,%20java.lang.String,%20android.security.keystore.KeyGenParameterSpec,%20int))

Constant Value: 4 (0x00000004)

ID_TYPE_INDIVIDUAL_ATTESTATION

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int ID_TYPE_INDIVIDUAL_ATTESTATION
```

Specifies that the device should attest using an individual attestation certificate. For use with `generateKeyPair(ComponentName, String, KeyGenParameterSpec, int)` (/reference/android/app/admin/DevicePolicyManager#generateKeyPair(android.content.ComponentName,%20java.lang.String,%20android.security.keystore.KeyGenParameterSpec,%20int)).

See also:

`generateKeyPair(ComponentName, String, KeyGenParameterSpec, int)` (/reference/android/app/admin/DevicePolicyManager#generateKeyPair(android.content.ComponentName,%20java.lang.String,%20android.security.keystore.KeyGenParameterSpec,%20int))

Constant Value: 16 (0x00000010)

ID_TYPE_MEID

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int ID_TYPE_MEID
```

Specifies that the device should attest its MEID. For use with `generateKeyPair(ComponentName, String, KeyGenParameterSpec, int)` (/reference/android/app/admin/DevicePolicyManager#generateKeyPair(android.content.ComponentName,%20java.lang.String,%20android.security.keystore.KeyGenParameterSpec,%20int)).

See also:

`generateKeyPair(ComponentName, String, KeyGenParameterSpec, int)` (/reference/android/app/admin/DevicePolicyManager#generateKeyPair(android.content.ComponentName,%20java.lang.String,%20android.security.keystore.KeyGenParameterSpec,%20int))

Constant Value: 8 (0x00000008)

ID_TYPE_SERIAL

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int ID_TYPE_SERIAL
```

Specifies that the device should attest its serial number. For use with `generateKeyPair(ComponentName, String, KeyGenParameterSpec, int)` (/reference/android/app/admin/DevicePolicyManager#generateKeyPair(android.content.ComponentName,%20java.lang.String,%20android.security.keystore.KeyGenParameterSpec,%20int)).

See also:

`generateKeyPair(ComponentName, String, KeyGenParameterSpec, int)` (/reference/android/app/admin/DevicePolicyManager#generateKeyPair(android.content.ComponentName,%20java.lang.String,%20android.security.keystore.KeyGenParameterSpec,%20int))

Constant Value: 2 (0x00000002)

INSTALLKEY_REQUEST_CREDENTIALS_ACCESS

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int INSTALLKEY_REQUEST_CREDENTIALS_ACCESS
```

Specifies that the calling app should be granted access to the installed credentials immediately. Otherwise, access to the credentials will be gated by user approval. For use with `installKeyPair(android.content.ComponentName, java.security.PrivateKey, java.security.cert.Certificate[], java.lang.String, int)` (/reference/android/app/admin/DevicePolicyManager#installKeyPair(android.content.ComponentName,%20java.security.PrivateKey,%20java.security.cert.Certificate[],%20java.lang.String,%20int)).

See also:

`installKeyPair(ComponentName, PrivateKey, Certificate[], String, int)` (/reference/android/app/admin/DevicePolicyManager#installKeyPair(android.content.ComponentName,%20java.security.PrivateKey,%20java.security.cert.Certificate[],%20java.lang.String,%20int))

Constant Value: 1 (0x00000001)

INSTALLKEY_SET_USER_SELECTABLE

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int INSTALLKEY_SET_USER_SELECTABLE
```

Specifies that a user can select the key via the Certificate Selection prompt. If this flag is not set when calling `installKeyPair(ComponentName, PrivateKey, Certificate, String)` (/reference/android/app/admin/DevicePolicyManager#installKeyPair(android.content.ComponentName,%20java.security.PrivateKey,%20java.security.cert.Certificate,%20java.lang.String)), the key can only be granted access by implementing `DeviceAdminReceiver.onChoosePrivateKeyAlias(Context, Intent, int, Uri, String)` (/reference/android/app/admin/DeviceAdminReceiver#onChoosePrivateKeyAlias(android.content.Context,%20android.content.Intent,%20int,%20android.net.Uri,%20java.lang.String)). For use with `installKeyPair(android.content.ComponentName, java.security.PrivateKey, java.security.cert.Certificate[], java.lang.String, int)` (/reference/android/app/admin/DevicePolicyManager#installKeyPair(android.content.ComponentName,%20java.security.PrivateKey,%20java.security.cert.Certificate[],%20java.lang.String,%20int))

See also:

`installKeyPair(ComponentName, PrivateKey, Certificate[], String, int)` (/reference/android/app/admin/DevicePolicyManager#installKeyPair(android.content.ComponentName,%20java.security.PrivateKey,%20java.security.cert.Certificate[],%20java.lang.String,%20int))

Constant Value: 2 (0x00000002)

KEYGUARD_DISABLE_BIOMETRICS

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int KEYGUARD_DISABLE_BIOMETRICS
```

Disable all biometric authentication on keyguard secure screens (e.g. PIN/Pattern/Password).

Constant Value: 416 (0x000001a0)

KEYGUARD_DISABLE_FACE

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int KEYGUARD_DISABLE_FACE
```

Disable face authentication on keyguard secure screens (e.g. PIN/Pattern/Password).

Constant Value: 128 (0x00000080)

KEYGUARD_DISABLE_FEATURES_ALL

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int KEYGUARD_DISABLE_FEATURES_ALL
```

Disable all current and future keyguard customizations.

Constant Value: 2147483647 (0x7fffffff)

KEYGUARD_DISABLE_FEATURES_NONE

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int KEYGUARD_DISABLE_FEATURES_NONE
```

Widgets are enabled in keyguard

Constant Value: 0 (0x00000000)

KEYGUARD_DISABLE_FINGERPRINT

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int KEYGUARD_DISABLE_FINGERPRINT
```

Disable fingerprint authentication on keyguard secure screens (e.g. PIN/Pattern/Password).

Constant Value: 32 (0x00000020)

KEYGUARD_DISABLE_IRIS

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int KEYGUARD_DISABLE_IRIS
```

Disable iris authentication on keyguard secure screens (e.g. PIN/Pattern/Password).

Constant Value: 256 (0x00000100)

KEYGUARD_DISABLE_REMOTE_INPUT

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int KEYGUARD_DISABLE_REMOTE_INPUT
```

This constant was deprecated in API level 33.
This flag was added in version **`Build.VERSION_CODES.N`** (/reference/android/os/Build.VERSION_CODES#N), but it never had any effect.

Disable text entry into notifications on secure keyguard screens (e.g. PIN/Pattern/Password).

Constant Value: 64 (0x00000040)

KEYGUARD_DISABLE_SECURE_CAMERA

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int KEYGUARD_DISABLE_SECURE_CAMERA
```

Disable the camera on secure keyguard screens (e.g. PIN/Pattern/Password)

Constant Value: 2 (0x00000002)

KEYGUARD_DISABLE_SECURE_NOTIFICATIONS

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int KEYGUARD_DISABLE_SECURE_NOTIFICATIONS
```

Disable showing all notifications on secure keyguard screens (e.g. PIN/Pattern/Password)

Constant Value: 4 (0x00000004)

KEYGUARD_DISABLE_SHORTCUTS_ALL

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int KEYGUARD_DISABLE_SHORTCUTS_ALL
```

Disable all keyguard shortcuts.

Constant Value: 512 (0x00000200)

KEYGUARD_DISABLE_TRUST_AGENTS

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int KEYGUARD_DISABLE_TRUST_AGENTS
```

Disable trust agents on secure keyguard screens (e.g. PIN/Pattern/Password). By setting this flag alone, all trust agents are disabled. If the admin then wants to allowlist specific features of some trust agent, `setTrustAgentConfiguration(ComponentName, ComponentName, PersistableBundle)` (/reference/android/app/admin/DevicePolicyManager#setTrustAgentConfiguration(android.content.ComponentName,%20android.content.ComponentName,%20android.os.PersistableBundle)) can be used in conjuction to set trust-agent-specific configurations.

Constant Value: 16 (0x00000010)

KEYGUARD_DISABLE_UNREDACTED_NOTIFICATIONS

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int KEYGUARD_DISABLE_UNREDACTED_NOTIFICATIONS
```

Only allow redacted notifications on secure keyguard screens (e.g. PIN/Pattern/Password)

Constant Value: 8 (0x00000008)

KEYGUARD_DISABLE_WIDGETS_ALL

Added in API level 17 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int KEYGUARD_DISABLE_WIDGETS_ALL
```

Disable all keyguard widgets. Has no effect starting from `Build.VERSION_CODES.LOLLIPOP` (/reference/android/os/Build.VERSION_CODES#LOLLIPOP) since keyguard widget is only supported on Android versions lower than 5.0.

Constant Value: 1 (0x00000001)

LEAVE_ALL_SYSTEM_APPS_ENABLED

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int LEAVE_ALL_SYSTEM_APPS_ENABLED
```

Flag used by `createAndManageUser(ComponentName, String, ComponentName, PersistableBundle, int)` (/reference/android/app/admin/DevicePolicyManager#createAndManageUser(android.content.ComponentName,%20java.lang.String,%20android.content.ComponentName,%20android.os.PersistableBundle,%20int)) to specify that the newly created user should skip the disabling of system apps during provisioning.

Constant Value: 16 (0x00000010)

LOCK_TASK_FEATURE_BLOCK_ACTIVITY_START_IN_TASK

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int LOCK_TASK_FEATURE_BLOCK_ACTIVITY_START_IN_TASK
```

Enable blocking of non-allowlisted activities from being started into a locked task.

See also:

`setLockTaskFeatures(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setLockTaskFeatures(android.content.ComponentName,%20int))

Constant Value: 64 (0x00000040)

LOCK_TASK_FEATURE_GLOBAL_ACTIONS

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int LOCK_TASK_FEATURE_GLOBAL_ACTIONS
```

Enable the global actions dialog during LockTask mode. This is the dialog that shows up when the user long-presses the power button, for example. Note that the user may not be able to power off the device if this flag is not set.

This flag is enabled by default until `setLockTaskFeatures(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setLockTaskFeatures(android.content.ComponentName,%20int)) is called for the first time.

See also:

`setLockTaskFeatures(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setLockTaskFeatures(android.content.ComponentName,%20int))

Constant Value: 16 (0x00000010)

LOCK_TASK_FEATURE_HOME

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int LOCK_TASK_FEATURE_HOME
```

Enable the Home button during LockTask mode. Note that if a custom launcher is used, it has to be registered as the default launcher with `addPersistentPreferredActivity(android.content.ComponentName, android.content.IntentFilter, android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#addPersistentPreferredActivity(android.content.ComponentName,%20android.content.IntentFilter,%20android.content.ComponentName)), and its package needs to be allowlisted for LockTask with `setLockTaskPackages(android.content.ComponentName, java.lang.String[])` (/reference/android/app/admin/DevicePolicyManager#setLockTaskPackages(android.content.ComponentName,%20java.lang.String[])).

See also:

`setLockTaskFeatures(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setLockTaskFeatures(android.content.ComponentName,%20int))

Constant Value: 4 (0x00000004)

LOCK_TASK_FEATURE_KEYGUARD

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int LOCK_TASK_FEATURE_KEYGUARD
```

Enable the keyguard during LockTask mode. Note that if the keyguard is already disabled with `setKeyguardDisabled(android.content.ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setKeyguardDisabled(android.content.ComponentName,%20boolean)), setting this flag will have no effect. If this flag is not set, the keyguard will not be shown even if the user has a lock screen credential.

See also:

`setLockTaskFeatures(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setLockTaskFeatures(android.content.ComponentName,%20int))

Constant Value: 32 (0x00000020)

LOCK_TASK_FEATURE_NONE

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int LOCK_TASK_FEATURE_NONE
```

Disable all configurable SystemUI features during LockTask mode. This includes,

- system info area in the status bar (connectivity icons, clock, etc.)
- notifications (including alerts, icons, and the notification shade)
- Home button
- Recents button and UI
- global actions menu (i.e. power button menu)
- keyguard

See also:

`setLockTaskFeatures(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setLockTaskFeatures(android.content.ComponentName,%20int))

Constant Value: 0 (0x00000000)

LOCK_TASK_FEATURE_NOTIFICATIONS

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int LOCK_TASK_FEATURE_NOTIFICATIONS
```

Enable notifications during LockTask mode. This includes notification icons on the status bar, heads-up notifications, and the expandable notification shade. Note that the Quick Settings panel remains disabled. This feature flag can only be used in combination with `LOCK_TASK_FEATURE_HOME` (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FEATURE_HOME). `setLockTaskFeatures(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setLockTaskFeatures(android.content.ComponentName,%20int)) throws an `IllegalArgumentException` (/reference/java/lang/IllegalArgumentException) if this feature flag is defined without `LOCK_TASK_FEATURE_HOME` (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FEATURE_HOME).

See also:

`setLockTaskFeatures(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setLockTaskFeatures(android.content.ComponentName,%20int))

Constant Value: 2 (0x00000002)

LOCK_TASK_FEATURE_OVERVIEW

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int LOCK_TASK_FEATURE_OVERVIEW
```

Enable the Overview button and the Overview screen during LockTask mode. This feature flag can only be used in combination with `LOCK_TASK_FEATURE_HOME` (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FEATURE_HOME), and `setLockTaskFeatures(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setLockTaskFeatures(android.content.ComponentName,%20int)) will throw an `IllegalArgumentException` (/reference/java/lang/IllegalArgumentException) if this feature flag is defined without `LOCK_TASK_FEATURE_HOME` (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FEATURE_HOME).

See also:

`setLockTaskFeatures(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setLockTaskFeatures(android.content.ComponentName,%20int))

Constant Value: 8 (0x00000008)

LOCK_TASK_FEATURE_SYSTEM_INFO

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int LOCK_TASK_FEATURE_SYSTEM_INFO
```

Enable the system info area in the status bar during LockTask mode. The system info area usually occupies the right side of the status bar (although this can differ across OEMs). It includes all system information indicators, such as date and time, connectivity, battery, vibration mode, etc.

See also:

`setLockTaskFeatures(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setLockTaskFeatures(android.content.ComponentName,%20int))

Constant Value: 1 (0x00000001)

MAKE_USER_EPHEMERAL

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int MAKE_USER_EPHEMERAL
```

Flag used by `createAndManageUser(ComponentName, String, ComponentName, PersistableBundle, int)` (/reference/android/app/admin/DevicePolicyManager#createAndManageUser(android.content.ComponentName,%20java.lang.String,%20android.content.ComponentName,%20android.os.PersistableBundle,%20int)) to specify that the user should be created ephemeral. Ephemeral users will be removed after switching to another user or rebooting the device.

Constant Value: 2 (0x00000002)

MIME_TYPE_PROVISIONING_NFC

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) MIME_TYPE_PROVISIONING_NFC
```

This MIME type is used for starting the device owner provisioning.

During device owner provisioning a device admin app is set as the owner of the device. A device owner has full control over the device. The device owner can not be modified by the user and the only way of resetting the device is if the device owner app calls a factory reset.

A typical use case would be a device that is owned by a company, but used by either an employee or client.

The NFC message must be sent to an unprovisioned device.

The NFC record must contain a serialized `Properties` (/reference/java/util/Properties) object which contains the following properties:

- `EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_NAME` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_NAME)
- `EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_LOCATION` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_LOCATION), optional

- `EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_COOKIE_HEADER` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_DOWNLOAD_COOKIE_HEADER), optional
- `EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_CHECKSUM` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_CHECKSUM), optional
- `EXTRA_PROVISIONING_LOCAL_TIME` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_LOCAL_TIME) (convert to String), optional
- `EXTRA_PROVISIONING_TIME_ZONE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_TIME_ZONE), optional
- `EXTRA_PROVISIONING_LOCALE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_LOCALE), optional
- `EXTRA_PROVISIONING_WIFI_SSID` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SSID), optional
- `EXTRA_PROVISIONING_WIFI_HIDDEN` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_HIDDEN) (convert to String), optional
- `EXTRA_PROVISIONING_WIFI_SECURITY_TYPE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_SECURITY_TYPE), optional
- `EXTRA_PROVISIONING_WIFI_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_PASSWORD), optional
- `EXTRA_PROVISIONING_WIFI_PROXY_HOST` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_PROXY_HOST), optional
- `EXTRA_PROVISIONING_WIFI_PROXY_PORT` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_PROXY_PORT) (convert to String), optional

- `EXTRA_PROVISIONING_WIFI_PROXY_BYPASS` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_PROXY_BYPASS), optional
- `EXTRA_PROVISIONING_WIFI_PAC_URL` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_PAC_URL), optional
- `EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_ADMIN_EXTRAS_BUNDLE), optional, supported from `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M)
- `EXTRA_PROVISIONING_WIFI_EAP_METHOD` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_EAP_METHOD), optional, supported from `Build.VERSION_CODES.Q` (/reference/android/os/Build.VERSION_CODES#Q)
- `EXTRA_PROVISIONING_WIFI_PHASE2_AUTH` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_PHASE2_AUTH), optional, supported from `Build.VERSION_CODES.Q` (/reference/android/os/Build.VERSION_CODES#Q)
- `EXTRA_PROVISIONING_WIFI_CA_CERTIFICATE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_CA_CERTIFICATE), optional, supported from `Build.VERSION_CODES.Q` (/reference/android/os/Build.VERSION_CODES#Q)
- `EXTRA_PROVISIONING_WIFI_USER_CERTIFICATE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_USER_CERTIFICATE), optional, supported from `Build.VERSION_CODES.Q` (/reference/android/os/Build.VERSION_CODES#Q)
- `EXTRA_PROVISIONING_WIFI_IDENTITY` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_IDENTITY), optional, supported from `Build.VERSION_CODES.Q` (/reference/android/os/Build.VERSION_CODES#Q)
- `EXTRA_PROVISIONING_WIFI_ANONYMOUS_IDENTITY` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_ANONYMOUS_IDENTITY)

, optional, supported from `Build.VERSION_CODES.Q` (/reference/android/os/Build.VERSION_CODES#Q)

- `EXTRA_PROVISIONING_WIFI_DOMAIN` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_WIFI_DOMAIN), optional, supported from `Build.VERSION_CODES.Q` (/reference/android/os/Build.VERSION_CODES#Q)

As of `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M), the properties should contain `EXTRA_PROVISIONING_DEVICE_ADMIN_COMPONENT_NAME` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_COMPONENT_NAME) instead of `EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_NAME` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_NAME), (although specifying only `EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_NAME` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_DEVICE_ADMIN_PACKAGE_NAME) is still supported).

Constant Value: "application/com.android.managedprovisioning"

MTE_DISABLED

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int MTE_DISABLED
```

Require that MTE be disabled on the device. Can be set by a device owner.

Constant Value: 2 (0x00000002)

MTE_ENABLED

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int MTE_ENABLED
```

Require that MTE be enabled on the device, if supported. Can be set by a device owner or a profile owner of an organization-owned managed profile.

Constant Value: 1 (0x00000001)

MTE_NOT_CONTROLLED_BY_POLICY

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int MTE_NOT_CONTROLLED_BY_POLICY
```

Allow the user to choose whether to enable MTE on the device.

Constant Value: 0 (0x00000000)

NEARBY_STREAMING_DISABLED

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int NEARBY_STREAMING_DISABLED
```

Indicates that nearby streaming is disabled.

Constant Value: 1 (0x00000001)

NEARBY_STREAMING_ENABLED

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int NEARBY_STREAMING_ENABLED
```

Indicates that nearby streaming is enabled.

Constant Value: 2 (0x00000002)

NEARBY_STREAMING_NOT_CONTROLLED_BY_POLICY

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int NEARBY_STREAMING_NOT_CONTROLLED_BY_POLICY
```

Indicates that nearby streaming is not controlled by policy, which means nearby streaming is allowed.

Constant Value: 0 (0x00000000)

NEARBY_STREAMING_SAME_MANAGED_ACCOUNT_ONLY

Added in API level 37 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int NEARBY_STREAMING_SAME_MANAGED_ACCOUNT_ONLY
```

Indicates that nearby streaming is enabled only to devices offering a comparable level of security, with the same authenticated managed account.

Constant Value: 3 (0x00000003)

OPERATION_SAFETY_REASON_DRIVING_DISTRACTION

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int OPERATION_SAFETY_REASON_DRIVING_DISTRACTION
```

Indicates that a `UnsafeStateException` (/reference/android/app/admin/UnsafeStateException) was thrown because the operation would distract the driver of the vehicle.

Constant Value: 1 (0x00000001)

PASSWORD_COMPLEXITY_HIGH

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PASSWORD_COMPLEXITY_HIGH
```

Constant for `getPasswordComplexity()` (/reference/android/app/admin/DevicePolicyManager#getPasswordComplexity()) and `setRequiredPasswordComplexity(int)` (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswordComplexity(int)). Define the high password complexity band as:

- PIN with **no** repeating (4444) or ordered (1234, 4321, 2468) sequences, length at least 8
- alphabetic, length at least 6
- alphanumeric, length at least 6

When returned from `getPasswordComplexity()` (/reference/android/app/admin/DevicePolicyManager#getPasswordComplexity()), the constant represents the exact complexity band the password is in. When passed to {@link #setRequiredPasswordComplexity(int), it sets the minimum complexity band which the password must meet.

See also:

`PASSWORD_QUALITY_NUMERIC_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_NUMERIC_COMPLEX)

`PASSWORD_QUALITY_ALPHABETIC` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_ALPHABETIC)

`PASSWORD_QUALITY_ALPHANUMERIC` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_ALPHANUMERIC)

Constant Value: 327680 (0x00050000)

PASSWORD_COMPLEXITY_LOW

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PASSWORD_COMPLEXITY_LOW
```

Constant for `getPasswordComplexity()` (/reference/android/app/admin/DevicePolicyManager#getPasswordComplexity()) and `setRequiredPasswordComplexity(int)` (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswordComplexity(int)). Define the low password complexity band as:

- pattern
- PIN with repeating (4444) or ordered (1234, 4321, 2468) sequences

When returned from `getPasswordComplexity()` (/reference/android/app/admin/DevicePolicyManager#getPasswordComplexity()), the constant

represents the exact complexity band the password is in. When passed to {@link #setRequiredPasswordComplexity(int), it sets the minimum complexity band which the password must meet.

See also:

`PASSWORD_QUALITY_SOMETHING` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_SOMETHING)

`PASSWORD_QUALITY_NUMERIC` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_NUMERIC)

Constant Value: 65536 (0x00010000)

PASSWORD_COMPLEXITY_MEDIUM

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PASSWORD_COMPLEXITY_MEDIUM
```

Constant for `getPasswordComplexity()` (/reference/android/app/admin/DevicePolicyManager#getPasswordComplexity()) and `setRequiredPasswordComplexity(int)` (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswordComplexity(int)). Define the medium password complexity band as:

- PIN with **no** repeating (4444) or ordered (1234, 4321, 2468) sequences, length at least 4
- alphabetic, length at least 4
- alphanumeric, length at least 4

When returned from `getPasswordComplexity()` (/reference/android/app/admin/DevicePolicyManager#getPasswordComplexity()), the constant represents the exact complexity band the password is in. When passed to {@link #setRequiredPasswordComplexity(int), it sets the minimum complexity band which the password must meet.

See also:

`PASSWORD_QUALITY_NUMERIC_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_NUMERIC_COMPLEX)

`PASSWORD_QUALITY_ALPHABETIC` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_ALPHABETIC)

`PASSWORD_QUALITY_ALPHANUMERIC` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_ALPHANUMERIC)

Constant Value: 196608 (0x00030000)

PASSWORD_COMPLEXITY_NONE

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PASSWORD_COMPLEXITY_NONE
```

Constant for `getPasswordComplexity()` (/reference/android/app/admin/DevicePolicyManager#getPasswordComplexity()) and `setRequiredPasswordComplexity(int)` (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswordComplexity(int)): no password.

When returned from `getPasswordComplexity()` (/reference/android/app/admin/DevicePolicyManager#getPasswordComplexity()), the constant represents the exact complexity band the password is in. When passed to {@link #setRequiredPasswordComplexity(int), it sets the minimum complexity band which the password must meet.

Constant Value: 0 (0x00000000)

PASSWORD_QUALITY_ALPHABETIC

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PASSWORD_QUALITY_ALPHABETIC
```

Constant for `setPasswordQuality(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) : the user must have entered a password containing at least alphabetic (or other symbol) characters. Note that quality constants are ordered so that higher values are more restrictive.

Constant Value: 262144 (0x00040000)

PASSWORD_QUALITY_ALPHANUMERIC

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PASSWORD_QUALITY_ALPHANUMERIC
```

Constant for `setPasswordQuality(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)): the user must have entered a password containing at least *both*> numeric *and* alphabetic (or other symbol) characters. Note that quality constants are ordered so that higher values are more restrictive.

Constant Value: 327680 (0x00050000)

PASSWORD_QUALITY_BIOMETRIC_WEAK

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PASSWORD_QUALITY_BIOMETRIC_WEAK
```

Constant for `setPasswordQuality(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)): the policy allows for low-security biometric recognition technology. This implies technologies that can recognize the identity of an individual to about a 3 digit PIN (false detection is less than 1 in 1,000). Note that quality constants are ordered so that higher values are more restrictive.

Constant Value: 32768 (0x00008000)

PASSWORD_QUALITY_COMPLEX

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PASSWORD_QUALITY_COMPLEX
```

Constant for `setPasswordQuality(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) : allows the admin to set precisely how many characters of various types the password should contain to satisfy the policy. The admin should set these requirements via `setPasswordMinimumLetters(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumLetters(android.content.ComponentName,%20int)) , `setPasswordMinimumNumeric(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumNumeric(android.content.ComponentName,%20int)) , `setPasswordMinimumSymbols(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumSymbols(android.content.ComponentName,%20int)) , `setPasswordMinimumUpperCase(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumUpperCase(android.content.ComponentName,%20int)) , `setPasswordMinimumLowerCase(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumLowerCase(android.content.ComponentName,%20int)) , `setPasswordMinimumNonLetter(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumNonLetter(android.content.ComponentName,%20int)) , and `setPasswordMinimumLength(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumLength(android.content.ComponentName,%20int)) . Note that quality constants are ordered so that higher values are more restrictive.

Constant Value: 393216 (0x00060000)

PASSWORD_QUALITY_NUMERIC

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PASSWORD_QUALITY_NUMERIC
```

Constant for `setPasswordQuality(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) : the user must have entered a password containing at least numeric characters. Note that quality constants are ordered so that higher values are more restrictive.

Constant Value: 131072 (0x00020000)

PASSWORD_QUALITY_NUMERIC_COMPLEX

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PASSWORD_QUALITY_NUMERIC_COMPLEX
```

Constant for `setPasswordQuality(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) : the user must have entered a password containing at least numeric characters with no repeating (4444) or ordered (1234, 4321, 2468) sequences. Note that quality constants are ordered so that higher values are more restrictive.

Constant Value: 196608 (0x00030000)

PASSWORD_QUALITY_SOMETHING

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PASSWORD_QUALITY_SOMETHING
```

Constant for `setPasswordQuality(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) : the policy requires some kind of password or pattern, but doesn't care what it is. Note that quality constants are ordered so that higher values are more restrictive.

Constant Value: 65536 (0x00010000)

PASSWORD_QUALITY_UNSPECIFIED

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PASSWORD_QUALITY_UNSPECIFIED
```

Constant for `setPasswordQuality(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) : the policy has no requirements for the password. Note that quality constants are ordered so that higher values are more restrictive.

Constant Value: 0 (0x00000000)

PERMISSION_GRANT_STATE_DEFAULT

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PERMISSION_GRANT_STATE_DEFAULT
```

Runtime permission state: The user can manage the permission through the UI.

Constant Value: 0 (0x00000000)

PERMISSION_GRANT_STATE_DENIED

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PERMISSION_GRANT_STATE_DENIED
```

Runtime permission state: The permission is denied to the app and the user cannot manage the permission through the UI.

Constant Value: 2 (0x00000002)

PERMISSION_GRANT_STATE_GRANTED

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PERMISSION_GRANT_STATE_GRANTED
```

Runtime permission state: The permission is granted to the app and the user cannot manage the permission through the UI.

Constant Value: 1 (0x00000001)

PERMISSION_POLICY_AUTO_DENY

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PERMISSION_POLICY_AUTO_DENY
```

Permission policy to always deny new permission requests for runtime permissions. Already granted or denied permissions are not affected by this.

Constant Value: 2 (0x00000002)

PERMISSION_POLICY_AUTO_GRANT

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PERMISSION_POLICY_AUTO_GRANT
```

Permission policy to always grant new permission requests for runtime permissions. Already granted or denied permissions are not affected by this.

Constant Value: 1 (0x00000001)

PERMISSION_POLICY_PROMPT

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PERMISSION_POLICY_PROMPT
```

Permission policy to prompt user for new permission requests for runtime permissions. Already granted or denied permissions are not affected by this.

Constant Value: 0 (0x00000000)

PERSONAL_APPS_NOT_SUSPENDED

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PERSONAL_APPS_NOT_SUSPENDED
```

Return value for `getPersonalAppsSuspendedReasons(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPersonalAppsSuspendedReasons(android.content.ComponentName)) when personal apps are not suspended.

Constant Value: 0 (0x00000000)

PERSONAL_APPS_SUSPENDED_EXPLICITLY

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PERSONAL_APPS_SUSPENDED_EXPLICITLY
```

Flag for `getPersonalAppsSuspendedReasons(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPersonalAppsSuspendedReasons(android.content.ComponentName)) return value. Set when personal apps are suspended by an admin explicitly via `setPersonalAppsSuspended(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setPersonalAppsSuspended(android.content.ComponentName,%20boolean)) .

Constant Value: 1 (0x00000001)

PERSONAL_APPS_SUSPENDED_PROFILE_TIMEOUT

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PERSONAL_APPS_SUSPENDED_PROFILE_TIMEOUT
```

Flag for `getPersonalAppsSuspendedReasons(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPersonalAppsSuspendedReasons(android.content.ComponentName)) return value. Set when personal apps are suspended by framework because managed profile was off for longer than allowed by policy.

See also:

`setManagedProfileMaximumTimeOff(ComponentName, long)` (/reference/android/app/admin/DevicePolicyManager#setManagedProfileMaximumTimeOff(android.content.ComponentName,%20long))

Constant Value: 2 (0x00000002)

POLICY_DISABLE_CAMERA

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) POLICY_DISABLE_CAMERA
```

Constant to indicate the feature of disabling the camera. Used as argument to `createAdminSupportIntent(java.lang.String)` (/reference/android/app/admin/DevicePolicyManager#createAdminSupportIntent(java.lang.String)).

See also:

`setCameraDisabled(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setCameraDisabled(android.content.ComponentName,%20boolean))

Constant Value: "policy_disable_camera"

POLICY_DISABLE_SCREEN_CAPTURE

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final String (/reference/java/lang/String) POLICY_DISABLE_SCREEN_CAPTURE
```

Constant to indicate the feature of disabling screen captures. Used as argument to `createAdminSupportIntent(java.lang.String)` (/reference/android/app/admin/DevicePolicyManager#createAdminSupportIntent(java.lang.String)).

See also:

`setScreenCaptureDisabled(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setScreenCaptureDisabled(android.content.ComponentName,%20boolean))

Constant Value: "policy_disable_screen_capture"

PRIVATE_DNS_MODE_OFF

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PRIVATE_DNS_MODE_OFF
```

Specifies that Private DNS was turned off completely.

Constant Value: 1 (0x00000001)

PRIVATE_DNS_MODE_OPPORTUNISTIC

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PRIVATE_DNS_MODE_OPPORTUNISTIC
```

Specifies that the device owner requested opportunistic DNS over TLS

Constant Value: 2 (0x00000002)

PRIVATE_DNS_MODE_PROVIDER_HOSTNAME

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PRIVATE_DNS_MODE_PROVIDER_HOSTNAME
```

Specifies that the device owner configured a specific host to use for Private DNS.

Constant Value: 3 (0x00000003)

PRIVATE_DNS_MODE_UNKNOWN

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PRIVATE_DNS_MODE_UNKNOWN
```

Specifies that the Private DNS setting is in an unknown state.

Constant Value: 0 (0x00000000)

PRIVATE_DNS_SET_ERROR_FAILURE_SETTING

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PRIVATE_DNS_SET_ERROR_FAILURE_SETTING
```

General failure to set the Private DNS mode, not due to one of the reasons listed above.

Constant Value: 2 (0x00000002)

PRIVATE_DNS_SET_ERROR_HOST_NOT_SERVING

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PRIVATE_DNS_SET_ERROR_HOST_NOT_SERVING
```

If the `privateDnsHost` provided was of a valid hostname but that host was found to not support DNS-over-TLS.

Constant Value: 1 (0x00000001)

PRIVATE_DNS_SET_NO_ERROR

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PRIVATE_DNS_SET_NO_ERROR
```

The selected mode has been set successfully. If the mode is `PRIVATE_DNS_MODE_PROVIDER_HOSTNAME` then it implies the supplied host is valid and reachable.

Constant Value: 0 (0x00000000)

PROVISIONING_MODE_FULLY_MANAGED_DEVICE

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PROVISIONING_MODE_FULLY_MANAGED_DEVICE
```

The provisioning mode for fully managed device.

Constant Value: 1 (0x00000001)

PROVISIONING_MODE_MANAGED_PROFILE

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PROVISIONING_MODE_MANAGED_PROFILE
```

The provisioning mode for managed profile.

Constant Value: 2 (0x00000002)

PROVISIONING_MODE_MANAGED_PROFILE_ON_PERSONAL_DEVICE

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int PROVISIONING_MODE_MANAGED_PROFILE_ON_PERSONAL_DEVICE
```

The provisioning mode for a managed profile on a personal device.

This mode is only available when the provisioning initiator has explicitly instructed the provisioning flow to support managed profile on a personal device provisioning. In that case, `PROVISIONING_MODE_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#PROVISIONING_MODE_MANAGED_PROFILE) corresponds to an organization-owned managed profile, whereas this constant corresponds to a personally-owned managed profile.

See also:

`EXTRA_PROVISIONING_MODE` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_MODE)

Constant Value: 3 (0x00000003)

RESET_PASSWORD_DO_NOT_ASK_CREDENTIALS_ON_BOOT

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int RESET_PASSWORD_DO_NOT_ASK_CREDENTIALS_ON_BOOT
```

Flag for `resetPasswordWithToken(ComponentName, String, byte, int)` (/reference/android/app/admin/DevicePolicyManager#resetPasswordWithToken(android.content.ComponentName,%20java.lang.String,%20byte[],%20int)) and `resetPassword(String, int)` (/reference/android/app/admin/DevicePolicyManager#resetPassword(java.lang.String,%20int)): don't ask for user credentials on device boot. If the flag is set, the device can be booted without asking for user password. The absence of this flag does not change the current boot requirements. This flag can be set by the device owner only. If the app is not the device owner, the flag is ignored. Once the flag is set, it cannot be reverted back without resetting the device to factory defaults.

Constant Value: 2 (0x00000002)

RESET_PASSWORD_REQUIRE_ENTRY

Added in API level 18 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int RESET_PASSWORD_REQUIRE_ENTRY
```

Flag for `resetPasswordWithToken(ComponentName, String, byte, int)` (/reference/android/app/admin/DevicePolicyManager#resetPasswordWithToken(android.content.ComponentName,%20java.lang.String,%20byte[],%20int)) and `resetPassword(String, int)` (/reference/android/app/admin/DevicePolicyManager#resetPassword(java.lang.String,%20int)): don't allow other admins to change the password again until the user has entered it.

Constant Value: 1 (0x00000001)

SKIP_SETUP_WIZARD

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int SKIP_SETUP_WIZARD
```

Flag used by `createAndManageUser(ComponentName, String, ComponentName, PersistableBundle, int)` (/reference/android/app/admin/DevicePolicyManager#createAndManageUser(android.content.ComponentName,%20java.lang.String,%20android.content.ComponentName,%20android.os.PersistableBundle,%20int)) to skip setup wizard after creating a new user.

Constant Value: 1 (0x00000001)

WIFI_SECURITY_ENTERPRISE_192

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int WIFI_SECURITY_ENTERPRISE_192
```

Constant for `getMinimumRequiredWifiSecurityLevel()` (/reference/android/app/admin/DevicePolicyManager#getMinimumRequiredWifiSecurityLevel()) and `setMinimumRequiredWifiSecurityLevel(int)` (/reference/android/app/admin/DevicePolicyManager#setMinimumRequiredWifiSecurityLevel(int)): enterprise 192 bit network.

When returned from `getMinimumRequiredWifiSecurityLevel()` (/reference/android/app/admin/DevicePolicyManager#getMinimumRequiredWifiSecurityLevel()), the constant represents the current minimum security level required. When passed to `setMinimumRequiredWifiSecurityLevel(int)` (/reference/android/app/admin/DevicePolicyManager#setMinimumRequiredWifiSecurityLevel(int)), it sets the minimum security level a Wi-Fi network must meet.

See also:

`WIFI_SECURITY_OPEN` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_OPEN)

`WIFI_SECURITY_PERSONAL` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_PERSONAL)

`WIFI_SECURITY_ENTERPRISE_EAP` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_ENTERPRISE_EAP)

Constant Value: 3 (0x00000003)

WIFI_SECURITY_ENTERPRISE_EAP

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int WIFI_SECURITY_ENTERPRISE_EAP
```

Constant for `getMinimumRequiredWifiSecurityLevel()` (/reference/android/app/admin/DevicePolicyManager#getMinimumRequiredWifiSecurityLevel()) and `setMinimumRequiredWifiSecurityLevel(int)` (/reference/android/app/admin/DevicePolicyManager#setMinimumRequiredWifiSecurityLevel(int)): enterprise EAP network.

When returned from `getMinimumRequiredWifiSecurityLevel()` (/reference/android/app/admin/DevicePolicyManager#getMinimumRequiredWifiSecurityLevel()), the constant represents the current minimum security level required. When passed to `setMinimumRequiredWifiSecurityLevel(int)` (/reference/android/app/admin/DevicePolicyManager#setMinimumRequiredWifiSecurityLevel(int)), it sets the minimum security level a Wi-Fi network must meet.

See also:

`WIFI_SECURITY_OPEN` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_OPEN)

`WIFI_SECURITY_PERSONAL` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_PERSONAL)

`WIFI_SECURITY_ENTERPRISE_192` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_ENTERPRISE_192)

Constant Value: 2 (0x00000002)

WIFI_SECURITY_OPEN Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int WIFI_SECURITY_OPEN
```

Constant for `getMinimumRequiredWifiSecurityLevel()` (/reference/android/app/admin/DevicePolicyManager#getMinimumRequiredWifiSecurityLevel()) and `setMinimumRequiredWifiSecurityLevel(int)` (/reference/android/app/admin/DevicePolicyManager#setMinimumRequiredWifiSecurityLevel(int)): no minimum security level.

When returned from `getMinimumRequiredWifiSecurityLevel()` (/reference/android/app/admin/DevicePolicyManager#getMinimumRequiredWifiSecurityLevel()), the constant represents the current minimum security level required. When passed to `setMinimumRequiredWifiSecurityLevel(int)`

(/reference/android/app/admin/DevicePolicyManager#setMinimumRequiredWifiSecurityLevel(int)), it sets the minimum security level a Wi-Fi network must meet.

See also:

`WIFI_SECURITY_PERSONAL` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_PERSONAL)

`WIFI_SECURITY_ENTERPRISE_EAP` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_ENTERPRISE_EAP)

`WIFI_SECURITY_ENTERPRISE_192` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_ENTERPRISE_192)

Constant Value: 0 (0x00000000)

WIFI_SECURITY_PERSONAL

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int WIFI_SECURITY_PERSONAL
```

Constant for `getMinimumRequiredWifiSecurityLevel()` (/reference/android/app/admin/DevicePolicyManager#getMinimumRequiredWifiSecurityLevel()) and `setMinimumRequiredWifiSecurityLevel(int)` (/reference/android/app/admin/DevicePolicyManager#setMinimumRequiredWifiSecurityLevel(int)): personal network such as WEP, WPA2-PSK.

When returned from `getMinimumRequiredWifiSecurityLevel()` (/reference/android/app/admin/DevicePolicyManager#getMinimumRequiredWifiSecurityLevel()), the constant represents the current minimum security level required. When passed to `setMinimumRequiredWifiSecurityLevel(int)` (/reference/android/app/admin/DevicePolicyManager#setMinimumRequiredWifiSecurityLevel(int)), it sets the minimum security level a Wi-Fi network must meet.

See also:

`WIFI_SECURITY_OPEN` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_OPEN)

`WIFI_SECURITY_ENTERPRISE_EAP` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_ENTERPRISE_EAP)

`WIFI_SECURITY_ENTERPRISE_192`
(/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_ENTERPRISE_192)

Constant Value: 1 (0x00000001)

WIPE_EUICC

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int WIPE_EUICC
```

Flag for `wipeData(int)` (/reference/android/app/admin/DevicePolicyManager#wipeData(int)): also erase the device's eUICC data.

Constant Value: 4 (0x00000004)

WIPE_EXTERNAL_STORAGE

Added in API level 9 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int WIPE_EXTERNAL_STORAGE
```

Flag for `wipeData(int)` (/reference/android/app/admin/DevicePolicyManager#wipeData(int)): also erase the device's adopted external storage (such as adopted SD cards).

See also:

Adoptable Storage Devices
(/reference/android/app/admin/{@docRoot}about/versions/marshmallow/android-6.0#adoptable-storage)

Constant Value: 1 (0x00000001)

WIPE_RESET_PROTECTION_DATA

Added in API level 22 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int WIPE_RESET_PROTECTION_DATA
```

Flag for `wipeData(int)` (/reference/android/app/admin/DevicePolicyManager#wipeData(int)): also erase the factory reset protection data.

This flag may only be set by device owner admins; if it is set by other admins a `SecurityException` (/reference/java/lang/SecurityException) will be thrown.

Constant Value: 2 (0x00000002)

WIPE_SILENTLY

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int WIPE_SILENTLY
```

Flag for `wipeData(int)` (/reference/android/app/admin/DevicePolicyManager#wipeData(int)): won't show reason for wiping to the user.

Constant Value: 8 (0x00000008)

Public methods

acknowledgeDeviceCompliant

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void acknowledgeDeviceCompliant ()
```

Called by a profile owner of an organization-owned managed profile to acknowledge that the device is compliant and the user can turn the profile off if needed according to the maximum time off policy. This method should be called when the device is deemed compliant after getting `DeviceAdminReceiver#onComplianceAcknowledgementRequired(Context, Intent)` (/reference/android/app/admin/DeviceAdminReceiver#onComplianceAcknowledgementRequired(android.content.Context,%20android.content.Intent)) callback in case it is overridden. Before this method is called the user is still free to turn the profile off, but the timer won't be reset, so personal apps will be suspended sooner. DPCs only need acknowledging device compliance if they override `DeviceAdminReceiver#onComplianceAcknowledgementRequired(Context, Intent)`

(/reference/android/app/admin/DeviceAdminReceiver#onComplianceAcknowledgementRequired(android.content.Context,%20android.content.Intent))
, otherwise compliance is acknowledged automatically.

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if the user isn't unlocked

See also:

`isComplianceAcknowledgementRequired()` (/reference/android/app/admin/DevicePolicyManager#isComplianceAcknowledgementRequired())

`setManagedProfileMaximumTimeOff(ComponentName, long)` (/reference/android/app/admin/DevicePolicyManager#setManagedProfileMaximumTimeOff(android.content.ComponentName,%20long))

`DeviceAdminReceiver.onComplianceAcknowledgementRequired(Context, Intent)` (/reference/android/app/admin/DeviceAdminReceiver#onComplianceAcknowledgementRequired(android.content.Context,%20android.content.Intent))

addCrossProfileIntentFilter

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void addCrossProfileIntentFilter (ComponentName (/reference/android/content/Compor
                IntentFilter (/reference/android/content/IntentFilter) filter,
                int flags)
```

Called by the profile owner of a managed profile so that some intents sent in the managed profile can also be resolved in the parent, or vice versa. Only activity intents are supported.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is assoc

	may be **null**.
filter	**IntentFilter**: The **IntentFilter** (/reference/android/content/IntentFilt match to be also resolved in the other profile
flags	**int**: **DevicePolicyManager#FLAG_MANAGED_CAN_ACCESS_PARENT** (/reference/android/app/admin/DevicePolicyManager#FLAG_MANAGED_C and **DevicePolicyManager#FLAG_PARENT_CAN_ACCESS_MANAGED** (/reference/android/app/admin/DevicePolicyManager#FLAG_PARENT_CAN are supported.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device or profile owner.

addCrossProfileWidgetProvider

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean addCrossProfileWidgetProvider (ComponentName (/reference/android/content/C
                String (/reference/java/lang/String) packageName)
```

Called by the profile owner of a managed profile or a holder of the permission **Manifest.permission.MANAGE_DEVICE_POLICY_PROFILE_INTERACTION** (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_PROFILE_INTERACTION) to enable widget providers from a given package to be available in the parent profile. As a result the user will be able to add widgets from the allowlisted package running under the profile to a widget host which runs under the parent profile, for example the home screen. Note that a package may have zero or more provider components, where each component provides a different widget type.

Note: By default no widget provider package is allowlisted.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.
`packageName`	`String`: The package from which widget providers are allowlisted.

Returns

`boolean`	Whether the package was added.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a profile owner and not a holder of the permission `MANAGE_DEVICE_POLICY_PROFILE_INTERACTION` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC.

See also:

`removeCrossProfileWidgetProvider(android.content.ComponentName, String)` (/reference/android/app/admin/DevicePolicyManager#removeCrossProfileWidgetProvider(android.content.ComponentName,%20java.lang.String))

`getCrossProfileWidgetProviders(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getCrossProfileWidgetProviders(android.content.ComponentName))

addOverrideApn

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int addOverrideApn (ComponentName (/reference/android/content/ComponentName) admin
                ApnSetting (/reference/android/telephony/data/ApnSetting) apnSetting)
```

Called by device owner or managed profile owner to add an override APN.

This method may returns `-1` if `apnSetting` conflicts with an existing override APN. Update the existing conflicted APN with `updateOverrideApn(android.content.ComponentName, int, android.telephony.data.ApnSetting)` (/reference/android/app/admin/DevicePolicyManager#updateOverrideApn(android.content.ComponentName,%20int,%20android.telephony.data.ApnSetting)) instead of adding a new entry.

Two override APNs are considered to conflict when all the following APIs return the same values on both override APNs:

- `ApnSetting#getOperatorNumeric()` (/reference/android/telephony/data/ApnSetting#getOperatorNumeric())
- `ApnSetting#getApnName()` (/reference/android/telephony/data/ApnSetting#getApnName())
- `ApnSetting#getProxyAddressAsString()` (/reference/android/telephony/data/ApnSetting#getProxyAddressAsString())
- `ApnSetting#getProxyPort()` (/reference/android/telephony/data/ApnSetting#getProxyPort())
- `ApnSetting#getMmsProxyAddressAsString()` (/reference/android/telephony/data/ApnSetting#getMmsProxyAddressAsString())
- `ApnSetting#getMmsProxyPort()` (/reference/android/telephony/data/ApnSetting#getMmsProxyPort())
- `ApnSetting#getMmsc()` (/reference/android/telephony/data/ApnSetting#getMmsc())
- `ApnSetting#isEnabled()` (/reference/android/telephony/data/ApnSetting#isEnabled())
- `ApnSetting#getMvnoType()` (/reference/android/telephony/data/ApnSetting#getMvnoType())
- `ApnSetting#getProtocol()` (/reference/android/telephony/data/ApnSetting#getProtocol())
- `ApnSetting#getRoamingProtocol()` (/reference/android/telephony/data/ApnSetting#getRoamingProtocol())

Before Android version `Build.VERSION_CODES.TIRAMISU` (/reference/android/os/Build.VERSION_CODES#TIRAMISU): Only device owners can add APNs.

Starting from Android version `Build.VERSION_CODES.TIRAMISU` (/reference/android/os/Build.VERSION_CODES#TIRAMISU): Both device owners and managed profile owners can add enterprise APNs (`ApnSetting#TYPE_ENTERPRISE` (/reference/android/telephony/data/ApnSetting#TYPE_ENTERPRISE)), while only device owners can add other type of APNs. Enterprise APNs are specific to the managed profile and do not override any user-configured VPNs. They are prerequisites for enabling preferential network service on the managed profile on 4G networks (`setPreferentialNetworkServiceConfigs(List)` (/reference/android/app/admin/DevicePolicyManager#setPreferentialNetworkServiceConfigs(java.util.List<android.app.admin.PreferentialNetworkServiceConfig>))).

Parameters

`admin`	`ComponentName`: which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with This value cannot be `null`.
`apnSetting`	`ApnSetting`: the override APN to insert This value cannot be `null`.

Returns

`int`	The `id` of inserted override APN. Or `-1` when failed to insert into the database.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	If request is for enterprise APN `admin` is either device owner or profile owner and in all other types of APN if `admin` is not a device owner.

See also:

`setOverrideApnsEnabled(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setOverrideApnsEnabled(android.content.ComponentName,%20boolean))

addPersistentPreferredActivity

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void addPersistentPreferredActivity (ComponentName (/reference/android/content/Com
                IntentFilter (/reference/android/content/IntentFilter) filter,
                ComponentName (/reference/android/content/ComponentName) activity)
```

Called by a profile owner or device owner or holder of the permission `Manifest.permission.MANAGE_DEVICE_POLICY_LOCK_TASK` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_LOCK_TASK). to set a default activity that the system selects to handle intents that match the given `IntentFilter` (/reference/android/content/IntentFilter) instead of showing the default disambiguation mechanism. This activity will remain the default intent handler even if the set of potential event handlers for the intent filter changes and if the intent preferences are reset.

Note that the target application should still declare the activity in the manifest, the API just sets the activity to be the default one to handle the given intent filter.

The default disambiguation mechanism takes over if the activity is not installed (anymore). When the activity is (re)installed, it is automatically reset as default intent handler for the filter.

Note that calling this API to set a default intent handler, only allow to avoid the default disambiguation mechanism. Implicit intents that do not trigger this mechanism (like invoking the browser) cannot be configured as they are controlled by other configurations.

The calling device admin must be a profile owner or device owner. If it is not, a security exception will be thrown.

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), after the persistent preferred activity policy has been set, `PolicyUpdateReceiver#onPolicySetResult(Context, String, Bundle, TargetUser, PolicyUpdateResult)`

(/reference/android/app/admin/PolicyUpdateReceiver#onPolicySetResult(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult))
will notify the admin on whether the policy was successfully set or not. This callback will contain:

- The policy identifier `DevicePolicyIdentifiers#PERSISTENT_PREFERRED_ACTIVITY_POLICY` (/reference/android/app/admin/DevicePolicyIdentifiers#PERSISTENT_PREFERRED_ACTIVITY_POLICY)
- The additional policy params bundle, which contains `PolicyUpdateReceiver#EXTRA_INTENT_FILTER` (/reference/android/app/admin/PolicyUpdateReceiver#EXTRA_INTENT_FILTER) the intent filter the policy applies to
- The `TargetUser` (/reference/android/app/admin/TargetUser) that this policy relates to
- The `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult), which will be `PolicyUpdateResult#RESULT_POLICY_SET` (/reference/android/app/admin/PolicyUpdateResult#RESULT_POLICY_SET) if the policy was successfully set or the reason the policy failed to be set (e.g. `PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY` (/reference/android/app/admin/PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY))

If there has been a change to the policy, `PolicyUpdateReceiver#onPolicyChanged(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicyChanged(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult))
will notify the admin of this change. This callback will contain the same parameters as PolicyUpdateReceiver#onPolicySetResult and the `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult) will contain the reason why the policy changed.

NOTE: Performs disk I/O and shouldn't be called on the main thread.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is

	associated with. Null if the caller is not a device admin. This value may be `null`.
`filter`	`IntentFilter`: The IntentFilter for which a default handler is added.
`activity`	`ComponentName`: The Activity that is added as default intent handler. This value cannot be `null`.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner or holder of the permission `permission.MANAGE_DEVICE_POLICY_LOCK_TASK` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC.

addUserRestriction

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void addUserRestriction (ComponentName (/reference/android/content/ComponentName)
                String (/reference/java/lang/String) key)
```

Called by a profile owner, device owner or a holder of any permission that is associated with a user restriction to set a user restriction specified by the key.

The calling device admin must be a profile owner, device owner or holder of any permission that is associated with a user restriction; if it is not, a security exception will be thrown.

The profile owner of an organization-owned managed profile may invoke this method on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance it obtained from `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) , for enforcing device-wide restrictions.

See the constants in `UserManager` (/reference/android/os/UserManager) for the list of restrictions that can be enforced device-wide. These constants will also state in their documentation which permission is required to manage the restriction using this API.

For callers targeting Android `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE) or above, calling this API will result in applying the restriction locally on the calling user, or locally on the parent profile if called from the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance obtained from `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) . To set a restriction globally, call `addUserRestrictionGlobally(String)` (/reference/android/app/admin/DevicePolicyManager#addUserRestrictionGlobally(java.lang.String)) instead.

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), after the user restriction policy has been set, `PolicyUpdateReceiver#onPolicySetResult(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicySetResult(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin on whether the policy was successfully set or not. This callback will contain:

- The policy identifier returned from `DevicePolicyIdentifiers#getIdentifierForUserRestriction(String)` (/reference/android/app/admin/DevicePolicyIdentifiers#getIdentifierForUserRestriction(java.lang.String))
- The `TargetUser` (/reference/android/app/admin/TargetUser) that this policy relates to
- The `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult), which will be `PolicyUpdateResult#RESULT_POLICY_SET` (/reference/android/app/admin/PolicyUpdateResult#RESULT_POLICY_SET) if the policy was successfully set or the reason the policy failed to be set (e.g. `PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY` (/reference/android/app/admin/PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY))

)

If there has been a change to the policy, `PolicyUpdateReceiver#onPolicyChanged(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicyChanged(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin of this change. This callback will contain the same parameters as PolicyUpdateReceiver#onPolicySetResult and the `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult) will contain the reason why the policy changed.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is assoc value cannot be `null`.
`key`	`String`: The key of the restriction. Value is `UserManager.DISALLOW_MOD` (/reference/android/os/UserManager#DISALLOW_MODIFY_ACCOUNTS), `U` `DISALLOW_CONFIG_WIFI` (/reference/android/os/UserManager#DISALLOV `UserManager.DISALLOW_CONFIG_LOCALE` (/reference/android/os/UserManager#DISALLOW_CONFIG_LOCALE), `User` `DISALLOW_INSTALL_APPS` (/reference/android/os/UserManager#DISALLC `UserManager.DISALLOW_UNINSTALL_APPS` (/reference/android/os/UserManager#DISALLOW_UNINSTALL_APPS), `Use` `DISALLOW_SHARE_LOCATION` (/reference/android/os/UserManager#DISALLOW_SHARE_LOCATION), `Use` `DISALLOW_AIRPLANE_MODE` (/reference/android/os/UserManager#DISALLOW_AIRPLANE_MODE), `User` `DISALLOW_CONFIG_BRIGHTNESS` (/reference/android/os/UserManager#DISALLOW_CONFIG_BRIGHTNESS), `DISALLOW_AMBIENT_DISPLAY` (/reference/android/os/UserManager#DISALLOW_AMBIENT_DISPLAY), `Use` `DISALLOW_CONFIG_SCREEN_TIMEOUT` (/reference/android/os/UserManager#DISALLOW_CONFIG_SCREEN_TIMEC `UserManager.DISALLOW_INSTALL_UNKNOWN_SOURCES` (/reference/android/os/UserManager#DISALLOW_INSTALL_UNKNOWN_SC `UserManager.DISALLOW_INSTALL_UNKNOWN_SOURCES_GLOBALLY` (/reference/android/os/UserManager#DISALLOW_INSTALL_UNKNOWN_SC , `UserManager.DISALLOW_CONFIG_BLUETOOTH` (/reference/android/os/UserManager#DISALLOW_CONFIG_BLUETOOTH), `DISALLOW_BLUETOOTH` (/reference/android/os/UserManager#DISALLOW_ `UserManager.DISALLOW_BLUETOOTH_SHARING`

(/reference/android/os/UserManager#DISALLOW_BLUETOOTH_SHARING),
DISALLOW_USB_FILE_TRANSFER
(/reference/android/os/UserManager#DISALLOW_USB_FILE_TRANSFER), U
DISALLOW_CONFIG_CREDENTIALS
(/reference/android/os/UserManager#DISALLOW_CONFIG_CREDENTIALS),
DISALLOW_REMOVE_USER (/reference/android/os/UserManager#DISALLOV
UserManager.DISALLOW_REMOVE_MANAGED_PROFILE
(/reference/android/os/UserManager#DISALLOW_REMOVE_MANAGED_PR
UserManager.DISALLOW_DEBUGGING_FEATURES
(/reference/android/os/UserManager#DISALLOW_DEBUGGING_FEATURES
DISALLOW_CONFIG_VPN (/reference/android/os/UserManager#DISALLOW_
UserManager.DISALLOW_CONFIG_LOCATION
(/reference/android/os/UserManager#DISALLOW_CONFIG_LOCATION), Us
DISALLOW_CONFIG_DATE_TIME
(/reference/android/os/UserManager#DISALLOW_CONFIG_DATE_TIME), Us
DISALLOW_CONFIG_TETHERING
(/reference/android/os/UserManager#DISALLOW_CONFIG_TETHERING), Us
DISALLOW_NETWORK_RESET
(/reference/android/os/UserManager#DISALLOW_NETWORK_RESET), Use
DISALLOW_FACTORY_RESET
(/reference/android/os/UserManager#DISALLOW_FACTORY_RESET), User
DISALLOW_ADD_USER (/reference/android/os/UserManager#DISALLOW_AI
UserManager.DISALLOW_ADD_MANAGED_PROFILE
(/reference/android/os/UserManager#DISALLOW_ADD_MANAGED_PROFIL
android.os.UserManager.DISALLOW_ADD_CLONE_PROFILE,
android.os.UserManager.DISALLOW_ADD_PRIVATE_PROFILE, UserManage
VERIFY_APPS (/reference/android/os/UserManager#ENSURE_VERIFY_APPS
DISALLOW_CONFIG_CELL_BROADCASTS
(/reference/android/os/UserManager#DISALLOW_CONFIG_CELL_BROADC
UserManager.DISALLOW_CONFIG_MOBILE_NETWORKS
(/reference/android/os/UserManager#DISALLOW_CONFIG_MOBILE_NETW
UserManager.DISALLOW_APPS_CONTROL
(/reference/android/os/UserManager#DISALLOW_APPS_CONTROL), UserI
DISALLOW_MOUNT_PHYSICAL_MEDIA
(/reference/android/os/UserManager#DISALLOW_MOUNT_PHYSICAL_MED
UserManager.DISALLOW_UNMUTE_MICROPHONE
(/reference/android/os/UserManager#DISALLOW_UNMUTE_MICROPHONE
DISALLOW_ADJUST_VOLUME
(/reference/android/os/UserManager#DISALLOW_ADJUST_VOLUME), User
DISALLOW_OUTGOING_CALLS
(/reference/android/os/UserManager#DISALLOW_OUTGOING_CALLS), Use
DISALLOW_SMS (/reference/android/os/UserManager#DISALLOW_SMS), Us
DISALLOW_FUN (/reference/android/os/UserManager#DISALLOW_FUN), Us
DISALLOW_CREATE_WINDOWS
(/reference/android/os/UserManager#DISALLOW_CREATE_WINDOWS), Us
DISALLOW_SYSTEM_ERROR_DIALOGS

(/reference/android/os/UserManager#DISALLOW_SYSTEM_ERROR_DIALO
UserManager.DISALLOW_CROSS_PROFILE_COPY_PASTE
(/reference/android/os/UserManager#DISALLOW_CROSS_PROFILE_COPY_
UserManager.DISALLOW_OUTGOING_BEAM
(/reference/android/os/UserManager#DISALLOW_OUTGOING_BEAM),
android.os.UserManager.DISALLOW_WALLPAPER, UserManager.DISALL
WALLPAPER (/reference/android/os/UserManager#DISALLOW_SET_WALLPA
UserManager.DISALLOW_SAFE_BOOT
(/reference/android/os/UserManager#DISALLOW_SAFE_BOOT),
android.os.UserManager.DISALLOW_RECORD_AUDIO,
android.os.UserManager.DISALLOW_RUN_IN_BACKGROUND,
android.os.UserManager.DISALLOW_CAMERA,
android.os.UserManager.DISALLOW_UNMUTE_DEVICE, UserManager.DIS
ROAMING (/reference/android/os/UserManager#DISALLOW_DATA_ROAMIN
DISALLOW_SET_USER_ICON
(/reference/android/os/UserManager#DISALLOW_SET_USER_ICON),
android.os.UserManager.DISALLOW_OEM_UNLOCK, UserManager.DISAL
PASSWORD (/reference/android/os/UserManager#DISALLOW_UNIFIED_PAS
UserManager.ALLOW_PARENT_PROFILE_APP_LINKING
(/reference/android/os/UserManager#ALLOW_PARENT_PROFILE_APP_LINK
UserManager.DISALLOW_AUTOFILL
(/reference/android/os/UserManager#DISALLOW_AUTOFILL), UserManag
CONTENT_CAPTURE (/reference/android/os/UserManager#DISALLOW_CON
UserManager.DISALLOW_CONTENT_SUGGESTIONS
(/reference/android/os/UserManager#DISALLOW_CONTENT_SUGGESTION
DISALLOW_USER_SWITCH (/reference/android/os/UserManager#DISALLOW
UserManager.DISALLOW_SHARE_INTO_MANAGED_PROFILE
(/reference/android/os/UserManager#DISALLOW_SHARE_INTO_MANAGED
UserManager.DISALLOW_PRINTING
(/reference/android/os/UserManager#DISALLOW_PRINTING), UserManage
CONFIG_PRIVATE_DNS
(/reference/android/os/UserManager#DISALLOW_CONFIG_PRIVATE_DNS),
DISALLOW_MICROPHONE_TOGGLE
(/reference/android/os/UserManager#DISALLOW_MICROPHONE_TOGGLE)
DISALLOW_CAMERA_TOGGLE
(/reference/android/os/UserManager#DISALLOW_CAMERA_TOGGLE), Use
RESTRICTIONS_PENDING
(/reference/android/os/UserManager#KEY_RESTRICTIONS_PENDING),
android.os.UserManager.DISALLOW_BIOMETRIC, UserManager.DISALLO
STATE (/reference/android/os/UserManager#DISALLOW_CHANGE_WIFI_ST
UserManager.DISALLOW_WIFI_TETHERING
(/reference/android/os/UserManager#DISALLOW_WIFI_TETHERING), User
DISALLOW_SHARING_ADMIN_CONFIGURED_WIFI
(/reference/android/os/UserManager#DISALLOW_SHARING_ADMIN_CONF
UserManager.DISALLOW_WIFI_DIRECT
(/reference/android/os/UserManager#DISALLOW_WIFI_DIRECT), UserMan

ADD_WIFI_CONFIG (/reference/android/os/UserManager#DISALLOW_ADD
UserManager.DISALLOW_CELLULAR_2G
(/reference/android/os/UserManager#DISALLOW_CELLULAR_2G), UserMa
ULTRA_WIDEBAND_RADIO
(/reference/android/os/UserManager#DISALLOW_ULTRA_WIDEBAND_RAD
DISALLOW_GRANT_ADMIN (/reference/android/os/UserManager#DISALLOV
UserManager.DISALLOW_NEAR_FIELD_COMMUNICATION_RADIO
(/reference/android/os/UserManager#DISALLOW_NEAR_FIELD_COMMUNI
android.os.UserManager.DISALLOW_THREAD_NETWORK,
android.os.UserManager.DISALLOW_SIM_GLOBALLY, or
android.os.UserManager.DISALLOW_ASSIST_CONTENT

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device or profile owner and if the caller has not been granted the permission to set the given user restriction.

addUserRestrictionGlobally

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void addUserRestrictionGlobally (String (/reference/java/lang/String) key)
```

Called by a profile owner, device owner or a holder of any permission that is associated with a user restriction to set a user restriction specified by the provided `key` globally on all users. To clear the restriction use `clearUserRestriction(ComponentName, String)` (/reference/android/app/admin/DevicePolicyManager#clearUserRestriction(android.content.ComponentName,%20java.lang.String))
.

For a given user, a restriction will be set if it was applied globally or locally by any admin.

The calling device admin must be a profile owner, device owner or or a holder of any permission that is associated with a user restriction; if it is not, a security exception will be thrown.

See the constants in `UserManager` (/reference/android/os/UserManager) for the list of restrictions that can be enforced device-wide. These constants will also state in their documentation

which permission is required to manage the restriction using this API.

After the user restriction policy has been set, `PolicyUpdateReceiver#onPolicySetResult(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicySetResult(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin on whether the policy was successfully set or not. This callback will contain:

- The policy identifier returned from `DevicePolicyIdentifiers#getIdentifierForUserRestriction(String)` (/reference/android/app/admin/DevicePolicyIdentifiers#getIdentifierForUserRestriction(java.lang.String))
- The `TargetUser` (/reference/android/app/admin/TargetUser) that this policy relates to
- The `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult), which will be `PolicyUpdateResult#RESULT_POLICY_SET` (/reference/android/app/admin/PolicyUpdateResult#RESULT_POLICY_SET) if the policy was successfully set or the reason the policy failed to be set (e.g. `PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY` (/reference/android/app/admin/PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY))

If there has been a change to the policy, `PolicyUpdateReceiver#onPolicyChanged(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicyChanged(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin of this change. This callback will contain the same parameters as PolicyUpdateReceiver#onPolicySetResult and the `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult) will contain the reason why the policy changed.

Parameters

`key`	`String`: The key of the restriction. This value cannot be `null`. Value is `UseDISALLOW_MODIFY_ACCOUNTS` (/reference/android/os/UserManager#DISALLOW_MODIFY_ACCOUNTS), `U`

DISALLOW_CONFIG_WIFI (/reference/android/os/UserManager#DISALLOV
UserManager.DISALLOW_CONFIG_LOCALE
(/reference/android/os/UserManager#DISALLOW_CONFIG_LOCALE), User
DISALLOW_INSTALL_APPS (/reference/android/os/UserManager#DISALLC
UserManager.DISALLOW_UNINSTALL_APPS
(/reference/android/os/UserManager#DISALLOW_UNINSTALL_APPS), Use
DISALLOW_SHARE_LOCATION
(/reference/android/os/UserManager#DISALLOW_SHARE_LOCATION), Use
DISALLOW_AIRPLANE_MODE
(/reference/android/os/UserManager#DISALLOW_AIRPLANE_MODE), User
DISALLOW_CONFIG_BRIGHTNESS
(/reference/android/os/UserManager#DISALLOW_CONFIG_BRIGHTNESS),
DISALLOW_AMBIENT_DISPLAY
(/reference/android/os/UserManager#DISALLOW_AMBIENT_DISPLAY), Use
DISALLOW_CONFIG_SCREEN_TIMEOUT
(/reference/android/os/UserManager#DISALLOW_CONFIG_SCREEN_TIMEC
UserManager.DISALLOW_INSTALL_UNKNOWN_SOURCES
(/reference/android/os/UserManager#DISALLOW_INSTALL_UNKNOWN_SC
UserManager.DISALLOW_INSTALL_UNKNOWN_SOURCES_GLOBALLY
(/reference/android/os/UserManager#DISALLOW_INSTALL_UNKNOWN_SC
, UserManager.DISALLOW_CONFIG_BLUETOOTH
(/reference/android/os/UserManager#DISALLOW_CONFIG_BLUETOOTH), U
DISALLOW_BLUETOOTH (/reference/android/os/UserManager#DISALLOW_I
UserManager.DISALLOW_BLUETOOTH_SHARING
(/reference/android/os/UserManager#DISALLOW_BLUETOOTH_SHARING),
DISALLOW_USB_FILE_TRANSFER
(/reference/android/os/UserManager#DISALLOW_USB_FILE_TRANSFER), U
DISALLOW_CONFIG_CREDENTIALS
(/reference/android/os/UserManager#DISALLOW_CONFIG_CREDENTIALS),
DISALLOW_REMOVE_USER (/reference/android/os/UserManager#DISALLOV
UserManager.DISALLOW_REMOVE_MANAGED_PROFILE
(/reference/android/os/UserManager#DISALLOW_REMOVE_MANAGED_PR
UserManager.DISALLOW_DEBUGGING_FEATURES
(/reference/android/os/UserManager#DISALLOW_DEBUGGING_FEATURES
DISALLOW_CONFIG_VPN (/reference/android/os/UserManager#DISALLOW_
UserManager.DISALLOW_CONFIG_LOCATION
(/reference/android/os/UserManager#DISALLOW_CONFIG_LOCATION), Us
DISALLOW_CONFIG_DATE_TIME
(/reference/android/os/UserManager#DISALLOW_CONFIG_DATE_TIME), Us
DISALLOW_CONFIG_TETHERING
(/reference/android/os/UserManager#DISALLOW_CONFIG_TETHERING), Us
DISALLOW_NETWORK_RESET
(/reference/android/os/UserManager#DISALLOW_NETWORK_RESET), Use
DISALLOW_FACTORY_RESET
(/reference/android/os/UserManager#DISALLOW_FACTORY_RESET), User
DISALLOW_ADD_USER (/reference/android/os/UserManager#DISALLOW_AI

UserManager.DISALLOW_ADD_MANAGED_PROFILE
(/reference/android/os/UserManager#DISALLOW_ADD_MANAGED_PROFIL
android.os.UserManager.DISALLOW_ADD_CLONE_PROFILE,
android.os.UserManager.DISALLOW_ADD_PRIVATE_PROFILE, UserManage
VERIFY_APPS (/reference/android/os/UserManager#ENSURE_VERIFY_APPS
DISALLOW_CONFIG_CELL_BROADCASTS
(/reference/android/os/UserManager#DISALLOW_CONFIG_CELL_BROADC
UserManager.DISALLOW_CONFIG_MOBILE_NETWORKS
(/reference/android/os/UserManager#DISALLOW_CONFIG_MOBILE_NETW
UserManager.DISALLOW_APPS_CONTROL
(/reference/android/os/UserManager#DISALLOW_APPS_CONTROL), UserI
DISALLOW_MOUNT_PHYSICAL_MEDIA
(/reference/android/os/UserManager#DISALLOW_MOUNT_PHYSICAL_MED
UserManager.DISALLOW_UNMUTE_MICROPHONE
(/reference/android/os/UserManager#DISALLOW_UNMUTE_MICROPHONE
DISALLOW_ADJUST_VOLUME
(/reference/android/os/UserManager#DISALLOW_ADJUST_VOLUME), User
DISALLOW_OUTGOING_CALLS
(/reference/android/os/UserManager#DISALLOW_OUTGOING_CALLS), Use
DISALLOW_SMS (/reference/android/os/UserManager#DISALLOW_SMS), Us
DISALLOW_FUN (/reference/android/os/UserManager#DISALLOW_FUN), Us
DISALLOW_CREATE_WINDOWS
(/reference/android/os/UserManager#DISALLOW_CREATE_WINDOWS), Us
DISALLOW_SYSTEM_ERROR_DIALOGS
(/reference/android/os/UserManager#DISALLOW_SYSTEM_ERROR_DIALOG
UserManager.DISALLOW_CROSS_PROFILE_COPY_PASTE
(/reference/android/os/UserManager#DISALLOW_CROSS_PROFILE_COPY_
UserManager.DISALLOW_OUTGOING_BEAM
(/reference/android/os/UserManager#DISALLOW_OUTGOING_BEAM),
android.os.UserManager.DISALLOW_WALLPAPER, UserManager.DISALLO
WALLPAPER (/reference/android/os/UserManager#DISALLOW_SET_WALLPA
UserManager.DISALLOW_SAFE_BOOT
(/reference/android/os/UserManager#DISALLOW_SAFE_BOOT),
android.os.UserManager.DISALLOW_RECORD_AUDIO,
android.os.UserManager.DISALLOW_RUN_IN_BACKGROUND,
android.os.UserManager.DISALLOW_CAMERA,
android.os.UserManager.DISALLOW_UNMUTE_DEVICE, UserManager.DIS
ROAMING (/reference/android/os/UserManager#DISALLOW_DATA_ROAMING
DISALLOW_SET_USER_ICON
(/reference/android/os/UserManager#DISALLOW_SET_USER_ICON),
android.os.UserManager.DISALLOW_OEM_UNLOCK, UserManager.DISALL
PASSWORD (/reference/android/os/UserManager#DISALLOW_UNIFIED_PASS
UserManager.ALLOW_PARENT_PROFILE_APP_LINKING
(/reference/android/os/UserManager#ALLOW_PARENT_PROFILE_APP_LINK
UserManager.DISALLOW_AUTOFILL
(/reference/android/os/UserManager#DISALLOW_AUTOFILL), UserManag

CONTENT_CAPTURE (/reference/android/os/UserManager#DISALLOW_CON
UserManager.DISALLOW_CONTENT_SUGGESTIONS
(/reference/android/os/UserManager#DISALLOW_CONTENT_SUGGESTION
DISALLOW_USER_SWITCH (/reference/android/os/UserManager#DISALLOV
UserManager.DISALLOW_SHARE_INTO_MANAGED_PROFILE
(/reference/android/os/UserManager#DISALLOW_SHARE_INTO_MANAGED
UserManager.DISALLOW_PRINTING
(/reference/android/os/UserManager#DISALLOW_PRINTING), UserManage
CONFIG_PRIVATE_DNS
(/reference/android/os/UserManager#DISALLOW_CONFIG_PRIVATE_DNS),
DISALLOW_MICROPHONE_TOGGLE
(/reference/android/os/UserManager#DISALLOW_MICROPHONE_TOGGLE)
DISALLOW_CAMERA_TOGGLE
(/reference/android/os/UserManager#DISALLOW_CAMERA_TOGGLE), Use
RESTRICTIONS_PENDING
(/reference/android/os/UserManager#KEY_RESTRICTIONS_PENDING),
android.os.UserManager.DISALLOW_BIOMETRIC, UserManager.DISALLO
STATE (/reference/android/os/UserManager#DISALLOW_CHANGE_WIFI_ST
UserManager.DISALLOW_WIFI_TETHERING
(/reference/android/os/UserManager#DISALLOW_WIFI_TETHERING), Userl
DISALLOW_SHARING_ADMIN_CONFIGURED_WIFI
(/reference/android/os/UserManager#DISALLOW_SHARING_ADMIN_CONF
UserManager.DISALLOW_WIFI_DIRECT
(/reference/android/os/UserManager#DISALLOW_WIFI_DIRECT), UserMana
ADD_WIFI_CONFIG (/reference/android/os/UserManager#DISALLOW_ADD
UserManager.DISALLOW_CELLULAR_2G
(/reference/android/os/UserManager#DISALLOW_CELLULAR_2G), UserMa
ULTRA_WIDEBAND_RADIO
(/reference/android/os/UserManager#DISALLOW_ULTRA_WIDEBAND_RAD
DISALLOW_GRANT_ADMIN (/reference/android/os/UserManager#DISALLOV
UserManager.DISALLOW_NEAR_FIELD_COMMUNICATION_RADIO
(/reference/android/os/UserManager#DISALLOW_NEAR_FIELD_COMMUNI
android.os.UserManager.DISALLOW_THREAD_NETWORK,
android.os.UserManager.DISALLOW_SIM_GLOBALLY, or
android.os.UserManager.DISALLOW_ASSIST_CONTENT

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner and if the caller has not granted the permission to set the given user restriction.

IllegalStateException (/reference/java/lang/IllegalStateException)	if caller is not targeting Android Build.VERSION_CODES.UPSIDE_DOWN_CAKE (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_C or above.

bindDeviceAdminServiceAsUser

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean bindDeviceAdminServiceAsUser (ComponentName (/reference/android/content/Co
                Intent (/reference/android/content/Intent) serviceIntent,
                ServiceConnection (/reference/android/content/ServiceConnection) conn,
                int flags,
                UserHandle (/reference/android/os/UserHandle) targetUser)
```

Called by a device owner to bind to a service from a secondary managed user or vice versa. See `getBindDeviceAdminTargetUsers(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getBindDeviceAdminTargetUsers(android.content.ComponentName)) for the pre-requirements of a device owner to bind to services of another managed user.

The service must be protected by `Manifest.permission.BIND_DEVICE_ADMIN` (/reference/android/Manifest.permission#BIND_DEVICE_ADMIN). Note that the `Context` (/reference/android/content/Context) used to obtain this `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance via `Context#getSystemService(Class)` (/reference/android/content/Context#getSystemService(java.lang.Class<T>)) will be used to bind to the `Service` (/reference/android/app/Service).

Note: This method used to be available for communication between device owner and profile owner. However, since Android 11, this combination is not possible. This method is now only useful for communication between device owner and managed secondary users.

Parameters

admin	ComponentName: Which DeviceAdminReceiver (/reference/android/app associated with. This value cannot be null.

`serviceIntent`	`Intent`: Identifies the service to connect to. The Intent must specify either to match an `IntentFilter` (/reference/android/content/IntentFilter) publis
`conn`	`ServiceConnection`: Receives information as the service is started and s `ServiceConnection` (/reference/android/content/ServiceConnection) obj
`flags`	`int`: Operation options for the binding operation. See `Context#bindServ` (/reference/android/content/Context#bindService(android.content.Intent,% Value is either `0` or a combination of `Context.BIND_AUTO_CREATE` (/reference/android/content/Context#BIND_AUTO_CREATE), `Context.BIN` (/reference/android/content/Context#BIND_DEBUG_UNBIND), `Context.B` (/reference/android/content/Context#BIND_NOT_FOREGROUND), `Context` (/reference/android/content/Context#BIND_ABOVE_CLIENT), `Context.BI` (/reference/android/content/Context#BIND_ALLOW_OOM_MANAGEMENT) (/reference/android/content/Context#BIND_WAIVE_PRIORITY), `Context.B` (/reference/android/content/Context#BIND_IMPORTANT), `Context.BIND_` (/reference/android/content/Context#BIND_ADJUST_WITH_ACTIVITY), `Con` (/reference/android/content/Context#BIND_NOT_PERCEPTIBLE), `Context` (/reference/android/content/Context#BIND_ALLOW_ACTIVITY_STARTS), `Cc` (/reference/android/content/Context#BIND_INCLUDE_CAPABILITIES), `Cont` (/reference/android/content/Context#BIND_SHARED_ISOLATED_PROCESS android.content.Context.BIND_PACKAGE_ISOLATED_PROCESS, and `Conte` (/reference/android/content/Context#BIND_EXTERNAL_SERVICE)
`targetUser`	`UserHandle`: Which user to bind to. Must be one of the users returned by `ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getBindDeviceAdmi , otherwise a `SecurityException` (/reference/java/lang/SecurityExceptio

Returns

`boolean`	If you have successfully bound to the service, `true` is returned; `false` is returned if the connection is not made and you will not receive the service object.

See also:

`Context.bindService(Intent, ServiceConnection, int)` (/reference/android/content/Context#bindService(android.content.Intent,%20android.content.ServiceConnection,%20int))

`getBindDeviceAdminTargetUsers(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getBindDeviceAdminTargetUsers(android.content.ComponentName))

bindDeviceAdminServiceAsUser

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean bindDeviceAdminServiceAsUser (ComponentName (/reference/android/content/Co
                Intent (/reference/android/content/Intent) serviceIntent,
                ServiceConnection (/reference/android/content/ServiceConnection) conn,
                Context.BindServiceFlags (/reference/android/content/Context.BindServiceFlags)
                UserHandle (/reference/android/os/UserHandle) targetUser)
```

See `bindDeviceAdminServiceAsUser(android.content.ComponentName, android.content.Intent, android.content.ServiceConnection, int, android.os.UserHandle)` (/reference/android/app/admin/DevicePolicyManager#bindDeviceAdminServiceAsUser(android.content.ComponentName,%20android.content.Intent,%20android.content.ServiceConnection,%20int,%20android.os.UserHandle)). Call `Context.BindServiceFlags#of(long)` (/reference/android/content/Context.BindServiceFlags#of(long)) to obtain a BindServiceFlags object.

Parameters

`admin`	`ComponentName`: This value cannot be `null`.
`serviceIntent`	`Intent`: This value cannot be `null`.
`conn`	`ServiceConnection`: This value cannot be `null`.

`flags`	`Context.BindServiceFlags`: This value cannot be `null`.
`targetUser`	`UserHandle`: This value cannot be `null`.

Returns

`boolean`

canAdminGrantSensorsPermissions

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean canAdminGrantSensorsPermissions ()
```

Returns true if the caller is running on a device where an admin can grant permissions related to device sensors. This is a signal that the device is a fully-managed device where personal usage is discouraged. The list of permissions is listed in `setPermissionGrantState(android.content.ComponentName, java.lang.String, java.lang.String, int)` (/reference/android/app/admin/DevicePolicyManager#setPermissionGrantState(android.content.ComponentName,%20java.lang.String,%20java.lang.String,%20int)). May be called by any app.

Returns

`boolean`	true if an admin can grant device sensors-related permissions, false otherwise.

canUsbDataSignalingBeDisabled

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean canUsbDataSignalingBeDisabled ()
```

Returns whether enabling or disabling USB data signaling is supported on the device.

Returns

`boolean`	`true` if the device supports enabling and disabling USB data signaling.

clearApplicationUserData

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void clearApplicationUserData (ComponentName (/reference/android/content/Component
                String (/reference/java/lang/String) packageName,
                Executor (/reference/java/util/concurrent/Executor) executor,
                DevicePolicyManager.OnClearApplicationUserDataListener (/reference/and
```

Called by the device owner or profile owner to clear application user data of a given package. The behaviour of this is equivalent to the target application calling `ActivityManager.clearApplicationUserData()` (/reference/android/app/ActivityManager#clearApplicationUserData()).

Note: an application can store data outside of its application data, e.g. external storage or user dictionary. This data will not be wiped by calling this API.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`packageName`	`String`: The name of the package which will have its user data wiped. This value cannot be `null`.

`executor`	`Executor`: The executor through which the listener should be invoked. This value cannot be `null`. Callback and listener events are dispatched through this `Executor` (/reference/java/util/concurrent/Executor), providing an easy way to control which thread is used. To dispatch events through the main thread of your application, you can use `Context.getMainExecutor()` (/reference/android/content/Context#getMainExecutor()). Otherwise, provide an `Executor` (/reference/java/util/concurrent/Executor) that dispatches to an appropriate thread.
`listener`	`DevicePolicyManager.OnClearApplicationUserDataListener`: A callback object that will inform the caller when the clearing is done. This value cannot be `null`.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not the device owner/profile owner.

clearCrossProfileIntentFilters

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void clearCrossProfileIntentFilters (ComponentName (/reference/android/content/Com
```

Called by a profile owner of a managed profile to remove the cross-profile intent filters that go from the managed profile to the parent, or from the parent to the managed profile. Only removes those that have been set by the profile owner.

Note: A list of default cross profile intent filters are set up by the system when the profile is created, some of them ensure the proper functioning of the profile, while others enable sharing of data from the parent to the managed profile for user convenience. These default intent filters are not cleared when this API is called. If the default cross profile data sharing is not desired, they can be disabled with `UserManager#DISALLOW_SHARE_INTO_MANAGED_PROFILE` (/reference/android/os/UserManager#DISALLOW_SHARE_INTO_MANAGED_PROFILE).

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value may be `null`.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a profile owner.

clearDeviceOwnerApp

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void clearDeviceOwnerApp (String (/reference/java/lang/String) packageName)
```

This method was deprecated in API level 26.
This method is expected to be used for testing purposes only. The device owner will lose control of the device and its data after calling it. In order to protect any sensitive data that remains on the device, it is advised that the device owner factory resets the device instead of calling this method. See `wipeData(int)` (/reference/android/app/admin/DevicePolicyManager#wipeData(int)).

Clears the current device owner. The caller must be the device owner. This function should be used cautiously as once it is called it cannot be undone. The device owner can only be set as a part of device setup, before it completes.

While some policies previously set by the device owner will be cleared by this method, it is a best-effort process and some other policies will still remain in place after the device owner is cleared.

Parameters

`packageName`	`String`: The package name of the device owner.

Throws

SecurityException (/reference/java/lang/SecurityException)	if the caller is not in `packageName` or `packageName` does not own the current device owner component.

clearPackagePersistentPreferredActivities

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void clearPackagePersistentPreferredActivities (ComponentName (/reference/andro
                String (/reference/java/lang/String) packageName)
```

Called by a profile owner or device owner or holder of the permission `Manifest.permission.MANAGE_DEVICE_POLICY_LOCK_TASK` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_LOCK_TASK) to remove all persistent intent handler preferences associated with the given package that were set by `addPersistentPreferredActivity(ComponentName, IntentFilter, ComponentName)` (/reference/android/app/admin/DevicePolicyManager#addPersistentPreferredActivity(android.content.ComponentName,%20android.content.IntentFilter,%20android.content.ComponentName)) .

The calling device admin must be a profile owner. If it is not, a security exception will be thrown.

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), after the persistent preferred activity policy has been cleared, `PolicyUpdateReceiver#onPolicySetResult(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicySetResult(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult))

will notify the admin on whether the policy was successfully cleared or not. This callback will contain:

- The policy identifier `DevicePolicyIdentifiers#PERSISTENT_PREFERRED_ACTIVITY_POLICY` (/reference/android/app/admin/DevicePolicyIdentifiers#PERSISTENT_PREFERRED_ACTIVITY_POLICY)
- The additional policy params bundle, which contains `PolicyUpdateReceiver#EXTRA_INTENT_FILTER` (/reference/android/app/admin/PolicyUpdateReceiver#EXTRA_INTENT_FILTER) the intent filter the policy applies to
- The `TargetUser` (/reference/android/app/admin/TargetUser) that this policy relates to
- The `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult), which will be `PolicyUpdateResult#RESULT_POLICY_SET` (/reference/android/app/admin/PolicyUpdateResult#RESULT_POLICY_SET) if the policy was successfully cleared or the reason the policy failed to be cleared (e.g. `PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY` (/reference/android/app/admin/PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY))
)

If there has been a change to the policy, `PolicyUpdateReceiver#onPolicyChanged(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicyChanged(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin of this change. This callback will contain the same parameters as PolicyUpdateReceiver#onPolicySetResult and the `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult) will contain the reason why the policy changed.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.

`packageName`	`String`: The name of the package for which preferences are removed.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner or holder of the permission `permission.MANAGE_DEVICE_POLICY_LOCK_TASK` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC .

clearProfileOwner

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void clearProfileOwner (ComponentName (/reference/android/content/ComponentName) a
```

This method was deprecated in API level 26.
This method is expected to be used for testing purposes only. The profile owner will lose control of the user and its data after calling it. In order to protect any sensitive data that remains on this user, it is advised that the profile owner deletes it instead of calling this method. See `wipeData(int)` (/reference/android/app/admin/DevicePolicyManager#wipeData(int)).

Clears the active profile owner. The caller must be the profile owner of this user, otherwise a SecurityException will be thrown. This method is not available to managed profile owners.

While some policies previously set by the profile owner will be cleared by this method, it is a best-effort process and some other policies will still remain in place after the profile owner is cleared.

Parameters

`admin`	`ComponentName`: The component to remove as the profile owner. This value cannot be `null`.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not an active profile owner, or the method is being called from a managed profile.

clearResetPasswordToken

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean clearResetPasswordToken (ComponentName (/reference/android/content/Compone
```

Called by a profile, device owner or holder of the permission `Manifest.permission.MANAGE_DEVICE_POLICY_RESET_PASSWORD` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_RESET_PASSWORD) to revoke the current password reset token.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, this method has no effect - the reset token should not have been set in the first place - and false is returned.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.

Returns

`boolean`	true if the operation is successful, false otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if admin is not a device or profile owner and if the caller does not th `Manifest.permission.MANAGE_DEVICE_POLICY_RESET_PASS` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC .

clearUserRestriction

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void clearUserRestriction (ComponentName (/reference/android/content/ComponentName
                String (/reference/java/lang/String) key)
```

Called by a profile owner, device owner or a holder of any permission that is associated with a user restriction to clear a user restriction specified by the key.

The calling device admin must be a profile or device owner; if it is not, a security exception will be thrown.

The profile owner of an organization-owned managed profile may invoke this method on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance it obtained from `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) , for clearing device-wide restrictions.

See the constants in `UserManager` (/reference/android/os/UserManager) for the list of restrictions. These constants state in their documentation which permission is required to manage the restriction using this API.

For callers targeting Android `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE) or above, calling this API will result in clearing any local and global restriction with the specified key that was previously set by the caller.

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), after the user restriction policy has been cleared, `PolicyUpdateReceiver#onPolicySetResult(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicySetResult(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin on whether the policy was successfully cleared or not. This callback will contain:

- The policy identifier returned from `DevicePolicyIdentifiers#getIdentifierForUserRestriction(String)` (/reference/android/app/admin/DevicePolicyIdentifiers#getIdentifierForUserRestriction(java.lang.String))
- The `TargetUser` (/reference/android/app/admin/TargetUser) that this policy relates to
- The `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult), which will be `PolicyUpdateResult#RESULT_POLICY_SET` (/reference/android/app/admin/PolicyUpdateResult#RESULT_POLICY_SET) if the policy was successfully cleared or the reason the policy failed to be cleared (e.g. `PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY` (/reference/android/app/admin/PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY))

If there has been a change to the policy, `PolicyUpdateReceiver#onPolicyChanged(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicyChanged(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin of this change. This callback will contain the same parameters as PolicyUpdateReceiver#onPolicySetResult and the `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult) will contain the reason why the policy changed.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is assoc value cannot be `null`.
`key`	`String`: The key of the restriction. Value is `UserManager.DISALLOW_MOD` (/reference/android/os/UserManager#DISALLOW_MODIFY_ACCOUNTS), `U` `DISALLOW_CONFIG_WIFI` (/reference/android/os/UserManager#DISALLOV `UserManager.DISALLOW_CONFIG_LOCALE` (/reference/android/os/UserManager#DISALLOW_CONFIG_LOCALE), `User` `DISALLOW_INSTALL_APPS` (/reference/android/os/UserManager#DISALLC `UserManager.DISALLOW_UNINSTALL_APPS` (/reference/android/os/UserManager#DISALLOW_UNINSTALL_APPS), `Use` `DISALLOW_SHARE_LOCATION` (/reference/android/os/UserManager#DISALLOW_SHARE_LOCATION), `Use` `DISALLOW_AIRPLANE_MODE` (/reference/android/os/UserManager#DISALLOW_AIRPLANE_MODE), `User` `DISALLOW_CONFIG_BRIGHTNESS` (/reference/android/os/UserManager#DISALLOW_CONFIG_BRIGHTNESS), `DISALLOW_AMBIENT_DISPLAY` (/reference/android/os/UserManager#DISALLOW_AMBIENT_DISPLAY), `Use` `DISALLOW_CONFIG_SCREEN_TIMEOUT` (/reference/android/os/UserManager#DISALLOW_CONFIG_SCREEN_TIMEC `UserManager.DISALLOW_INSTALL_UNKNOWN_SOURCES` (/reference/android/os/UserManager#DISALLOW_INSTALL_UNKNOWN_SC `UserManager.DISALLOW_INSTALL_UNKNOWN_SOURCES_GLOBALLY` (/reference/android/os/UserManager#DISALLOW_INSTALL_UNKNOWN_SC , `UserManager.DISALLOW_CONFIG_BLUETOOTH` (/reference/android/os/UserManager#DISALLOW_CONFIG_BLUETOOTH), `U` `DISALLOW_BLUETOOTH` (/reference/android/os/UserManager#DISALLOW_ `UserManager.DISALLOW_BLUETOOTH_SHARING` (/reference/android/os/UserManager#DISALLOW_BLUETOOTH_SHARING), `DISALLOW_USB_FILE_TRANSFER` (/reference/android/os/UserManager#DISALLOW_USB_FILE_TRANSFER), `U` `DISALLOW_CONFIG_CREDENTIALS` (/reference/android/os/UserManager#DISALLOW_CONFIG_CREDENTIALS), `DISALLOW_REMOVE_USER` (/reference/android/os/UserManager#DISALLOV `UserManager.DISALLOW_REMOVE_MANAGED_PROFILE` (/reference/android/os/UserManager#DISALLOW_REMOVE_MANAGED_PR `UserManager.DISALLOW_DEBUGGING_FEATURES` (/reference/android/os/UserManager#DISALLOW_DEBUGGING_FEATURES `DISALLOW_CONFIG_VPN` (/reference/android/os/UserManager#DISALLOW_

UserManager.DISALLOW_CONFIG_LOCATION
(/reference/android/os/UserManager#DISALLOW_CONFIG_LOCATION), Us
DISALLOW_CONFIG_DATE_TIME
(/reference/android/os/UserManager#DISALLOW_CONFIG_DATE_TIME), Us
DISALLOW_CONFIG_TETHERING
(/reference/android/os/UserManager#DISALLOW_CONFIG_TETHERING), Us
DISALLOW_NETWORK_RESET
(/reference/android/os/UserManager#DISALLOW_NETWORK_RESET), Use
DISALLOW_FACTORY_RESET
(/reference/android/os/UserManager#DISALLOW_FACTORY_RESET), User
DISALLOW_ADD_USER (/reference/android/os/UserManager#DISALLOW_AI
UserManager.DISALLOW_ADD_MANAGED_PROFILE
(/reference/android/os/UserManager#DISALLOW_ADD_MANAGED_PROFIL
android.os.UserManager.DISALLOW_ADD_CLONE_PROFILE,
android.os.UserManager.DISALLOW_ADD_PRIVATE_PROFILE, UserManage
VERIFY_APPS (/reference/android/os/UserManager#ENSURE_VERIFY_APPS
DISALLOW_CONFIG_CELL_BROADCASTS
(/reference/android/os/UserManager#DISALLOW_CONFIG_CELL_BROADC
UserManager.DISALLOW_CONFIG_MOBILE_NETWORKS
(/reference/android/os/UserManager#DISALLOW_CONFIG_MOBILE_NETW
UserManager.DISALLOW_APPS_CONTROL
(/reference/android/os/UserManager#DISALLOW_APPS_CONTROL), Userl
DISALLOW_MOUNT_PHYSICAL_MEDIA
(/reference/android/os/UserManager#DISALLOW_MOUNT_PHYSICAL_MED
UserManager.DISALLOW_UNMUTE_MICROPHONE
(/reference/android/os/UserManager#DISALLOW_UNMUTE_MICROPHONE
DISALLOW_ADJUST_VOLUME
(/reference/android/os/UserManager#DISALLOW_ADJUST_VOLUME), Use
DISALLOW_OUTGOING_CALLS
(/reference/android/os/UserManager#DISALLOW_OUTGOING_CALLS), Use
DISALLOW_SMS (/reference/android/os/UserManager#DISALLOW_SMS), Us
DISALLOW_FUN (/reference/android/os/UserManager#DISALLOW_FUN), Us
DISALLOW_CREATE_WINDOWS
(/reference/android/os/UserManager#DISALLOW_CREATE_WINDOWS), Us
DISALLOW_SYSTEM_ERROR_DIALOGS
(/reference/android/os/UserManager#DISALLOW_SYSTEM_ERROR_DIALO
UserManager.DISALLOW_CROSS_PROFILE_COPY_PASTE
(/reference/android/os/UserManager#DISALLOW_CROSS_PROFILE_COPY_
UserManager.DISALLOW_OUTGOING_BEAM
(/reference/android/os/UserManager#DISALLOW_OUTGOING_BEAM),
android.os.UserManager.DISALLOW_WALLPAPER, UserManager.DISALLO
WALLPAPER (/reference/android/os/UserManager#DISALLOW_SET_WALLPA
UserManager.DISALLOW_SAFE_BOOT
(/reference/android/os/UserManager#DISALLOW_SAFE_BOOT),
android.os.UserManager.DISALLOW_RECORD_AUDIO,
android.os.UserManager.DISALLOW_RUN_IN_BACKGROUND,

android.os.UserManager.DISALLOW_CAMERA,
android.os.UserManager.DISALLOW_UNMUTE_DEVICE, UserManager.DIS
ROAMING (/reference/android/os/UserManager#DISALLOW_DATA_ROAMIN
DISALLOW_SET_USER_ICON
(/reference/android/os/UserManager#DISALLOW_SET_USER_ICON),
android.os.UserManager.DISALLOW_OEM_UNLOCK, UserManager.DISAL
PASSWORD (/reference/android/os/UserManager#DISALLOW_UNIFIED_PASS
UserManager.ALLOW_PARENT_PROFILE_APP_LINKING
(/reference/android/os/UserManager#ALLOW_PARENT_PROFILE_APP_LINK
UserManager.DISALLOW_AUTOFILL
(/reference/android/os/UserManager#DISALLOW_AUTOFILL), UserManag
CONTENT_CAPTURE (/reference/android/os/UserManager#DISALLOW_CON
UserManager.DISALLOW_CONTENT_SUGGESTIONS
(/reference/android/os/UserManager#DISALLOW_CONTENT_SUGGESTION
DISALLOW_USER_SWITCH (/reference/android/os/UserManager#DISALLOV
UserManager.DISALLOW_SHARE_INTO_MANAGED_PROFILE
(/reference/android/os/UserManager#DISALLOW_SHARE_INTO_MANAGED
UserManager.DISALLOW_PRINTING
(/reference/android/os/UserManager#DISALLOW_PRINTING), UserManage
CONFIG_PRIVATE_DNS
(/reference/android/os/UserManager#DISALLOW_CONFIG_PRIVATE_DNS),
DISALLOW_MICROPHONE_TOGGLE
(/reference/android/os/UserManager#DISALLOW_MICROPHONE_TOGGLE)
DISALLOW_CAMERA_TOGGLE
(/reference/android/os/UserManager#DISALLOW_CAMERA_TOGGLE), Use
RESTRICTIONS_PENDING
(/reference/android/os/UserManager#KEY_RESTRICTIONS_PENDING),
android.os.UserManager.DISALLOW_BIOMETRIC, UserManager.DISALLO
STATE (/reference/android/os/UserManager#DISALLOW_CHANGE_WIFI_ST
UserManager.DISALLOW_WIFI_TETHERING
(/reference/android/os/UserManager#DISALLOW_WIFI_TETHERING), User
DISALLOW_SHARING_ADMIN_CONFIGURED_WIFI
(/reference/android/os/UserManager#DISALLOW_SHARING_ADMIN_CONF
UserManager.DISALLOW_WIFI_DIRECT
(/reference/android/os/UserManager#DISALLOW_WIFI_DIRECT), UserMan
ADD_WIFI_CONFIG (/reference/android/os/UserManager#DISALLOW_ADD
UserManager.DISALLOW_CELLULAR_2G
(/reference/android/os/UserManager#DISALLOW_CELLULAR_2G), UserMa
ULTRA_WIDEBAND_RADIO
(/reference/android/os/UserManager#DISALLOW_ULTRA_WIDEBAND_RAD
DISALLOW_GRANT_ADMIN (/reference/android/os/UserManager#DISALLOV
UserManager.DISALLOW_NEAR_FIELD_COMMUNICATION_RADIO
(/reference/android/os/UserManager#DISALLOW_NEAR_FIELD_COMMUNI
android.os.UserManager.DISALLOW_THREAD_NETWORK,
android.os.UserManager.DISALLOW_SIM_GLOBALLY, or
android.os.UserManager.DISALLOW_ASSIST_CONTENT

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device or profile owner and if the caller has not been granted the permission to set the given user restriction.

createAdminSupportIntent

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public Intent (/reference/android/content/Intent) createAdminSupportIntent (String (/reference/
```

Called by any app to display a support dialog when a feature was disabled by an admin. This returns an intent that can be used with `Context#startActivity(Intent)` (/reference/android/content/Context#startActivity(android.content.Intent)) to display the dialog. It will tell the user that the feature indicated by `restriction` was disabled by an admin, and include a link for more information. The default content of the dialog can be changed by the restricting admin via `setShortSupportMessage(android.content.ComponentName, java.lang.CharSequence)` (/reference/android/app/admin/DevicePolicyManager#setShortSupportMessage(android.content.ComponentName,%20java.lang.CharSequence)) . If the restriction is not set (i.e. the feature is available), then the return value will be `null`.

Parameters

`restriction`	**String**: Indicates for which feature the dialog should be displayed. Can be from **UserManager** (/reference/android/os/UserManager), e.g. **UserManag ADJUST_VOLUME** (/reference/android/os/UserManager#DISALLOW_ADJUS one of the constants **POLICY_DISABLE_CAMERA** (/reference/android/app/admin/DevicePolicyManager#POLICY_DISABLE_C **POLICY_DISABLE_SCREEN_CAPTURE** (/reference/android/app/admin/DevicePolicyManager#POLICY_DISABLE_S . This value cannot be **null**.

Returns

Intent (/reference/android/content/Intent)	Intent An intent to be used to start the dialog-activity if the restriction is set by an admin, or null if the restriction does not exist or no admin set it.

createAndManageUser Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public UserHandle (/reference/android/os/UserHandle) createAndManageUser (ComponentName (/re
                String (/reference/java/lang/String) name,
                ComponentName (/reference/android/content/ComponentName) profileOwner,
                PersistableBundle (/reference/android/os/PersistableBundle) adminExtras,
                int flags)
```

Called by a device owner to create a user with the specified name and a given component of the calling package as profile owner. The UserHandle returned by this method should not be persisted as user handles are recycled as users are removed and created. If you need to persist an identifier for this user, use `UserManager#getSerialNumberForUser` (/reference/android/os/UserManager#getSerialNumberForUser(android.os.UserHandle)). The new user will not be started in the background.

admin is the `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) which is the device owner. profileOwner is also a DeviceAdminReceiver in the same package as admin, and will become the profile owner and will be registered as an active admin on the new user. The profile owner package will be installed on the new user.

If the adminExtras are not null, they will be stored on the device until the user is started for the first time. Then the extras will be passed to the admin when onEnable is called.

From `Build.VERSION_CODES.P` (/reference/android/os/Build.VERSION_CODES#P) onwards, if targeting `Build.VERSION_CODES.P` (/reference/android/os/Build.VERSION_CODES#P), throws `UserOperationException` (/reference/android/os/UserManager.UserOperationException) instead of returning `null` on failure.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is assoc value cannot be `null`.
`name`	`String`: The user's name. This value cannot be `null`.
`profileOwner`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) will be profile owner same package as admin, otherwise no user is created and an IllegalArgume thrown. This value cannot be `null`.
`adminExtras`	`PersistableBundle`: Extras that will be passed to onEnable of the admin new user. This value may be `null`.
`flags`	`int`: `SKIP_SETUP_WIZARD` (/reference/android/app/admin/DevicePolicyManager#SKIP_SETUP_WIZARI `EPHEMERAL` (/reference/android/app/admin/DevicePolicyManager#MAKE_U and `LEAVE_ALL_SYSTEM_APPS_ENABLED` (/reference/android/app/admin/DevicePolicyManager#LEAVE_ALL_SYSTEM are supported. Value is either `0` or a combination of `SKIP_SETUP_WIZARD` (/reference/android/app/admin/DevicePolicyManager#SKIP_SETUP_WIZARI `EPHEMERAL` (/reference/android/app/admin/DevicePolicyManager#MAKE_U , android.app.admin.DevicePolicyManager.MAKE_USER_DEMO, and `LEAVE_ APPS_ENABLED` (/reference/android/app/admin/DevicePolicyManager#LEAVE_ALL_SYSTEM

Returns

`UserHandle`	the `UserHandle` (/reference/android/os/UserHandle) object for the (/reference/android/os/UserHandle)created user, or `null` if the user could not be created.

Throws

SecurityException (/reference/java/lang/SecurityException)	if headless device is in ERROR(DeviceAdminInfo#HEADLESS_DEVICE_OWNER_MODE_SINGLE_USER/android.app.admin.DeviceAdminInfo#HEADLESS_DEVICE_OWNER_MODE_SINGLE_USER DeviceAdminInfo#HEADLESS_DEVICE_OWNER_MODE_SINGLE_USER) (/) mode.
SecurityException (/reference/java/lang/SecurityException)	if admin is not a device owner
UserManager.UserOperationException (/reference/android/os/UserManager.UserOperationException)	if the user could not be created and the callin app is targeting Build.VERSION_CODES.P (/reference/android/os/Build.VERSION_CODES and running on Build.VERSION_CODES.P (/reference/android/os/Build.VERSION_CODES .

See also:

UserHandle (/reference/android/os/UserHandle)

enableSystemApp

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int enableSystemApp (ComponentName (/reference/android/content/ComponentName) admi
                Intent (/reference/android/content/Intent) intent)
```

Re-enable system apps by intent that were disabled by default when the user was initialized. This function can be called by a device owner, profile owner, or by a delegate given the DELEGATION_ENABLE_SYSTEM_APP (/reference/android/app/admin/DevicePolicyManager#DELEGATION_ENABLE_SYSTEM_APP) scope via setDelegatedScopes(ComponentName, String, List)

(/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if the caller is an enable system app delegate.
`intent`	`Intent`: An intent matching the app(s) to be installed. All apps that resolve for this intent will be re-enabled in the calling profile.

Returns

`int`	int The number of activities that matched the intent and were installed.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner.

See also:

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`DELEGATION_PACKAGE_ACCESS` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PACKAGE_ACCESS)

enableSystemApp

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void enableSystemApp (ComponentName (/reference/android/content/ComponentName) admin,
                String (/reference/java/lang/String) packageName)
```

Re-enable a system app that was disabled by default when the user was initialized. This function can be called by a device owner, profile owner, or by a delegate given the `DELEGATION_ENABLE_SYSTEM_APP` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_ENABLE_SYSTEM_APP) scope via `setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>)) .

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if the caller is an enable system app delegate.
`packageName`	`String`: The package to be re-enabled in the calling profile.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner.

See also:

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`DELEGATION_PACKAGE_ACCESS` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PACKAGE_ACCESS)

generateKeyPair

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public AttestedKeyPair (/reference/android/security/AttestedKeyPair) generateKeyPair (Compone
                String (/reference/java/lang/String) algorithm,
                KeyGenParameterSpec (/reference/android/security/keystore/KeyGenParameterSpec
                int idAttestationFlags)
```

This API can be called by the following to generate a new private/public key pair:

- Device owner
- Profile owner
- Delegated certificate installer
- Credential management app
- An app that holds the `Manifest.permission.MANAGE_DEVICE_POLICY_CERTIFICATES` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_CERTIFICATES) permission

If the device supports key generation via secure hardware, this method is useful for creating a key in KeyChain that never left the secure hardware. Access to the key is controlled the same way as in `installKeyPair(ComponentName, PrivateKey, Certificate, String)` (/reference/android/app/admin/DevicePolicyManager#installKeyPair(android.content.ComponentName,%20java.security.PrivateKey,%20java.security.cert.Certificate,%20java.lang.String)) .

From Android `Build.VERSION_CODES.S` (/reference/android/os/Build.VERSION_CODES#S), the credential management app can call this API. If called by the credential management app, the componentName must be `null`. Note, there can only be a credential management app on an unmanaged device.

Because this method might take several seconds to complete, it should only be called from a worker thread. This method returns `null` when called from the main thread.

This method is not thread-safe, calling it from multiple threads at the same time will result in undefined behavior. If the calling thread is interrupted while the invocation is in-flight, it will

eventually terminate and return `null`.

Note: If the provided `alias` is of an existing alias, all former grants that apps have been given to access the key and certificates associated with this alias will be revoked.

Attestation: to enable attestation, set an attestation challenge in `keySpec` via `KeyGenParameterSpec.Builder#setAttestationChallenge` (/reference/android/security/keystore/KeyGenParameterSpec.Builder#setAttestationChallenge(byte[])). By specifying flags to the `idAttestationFlags` parameter, it is possible to request the device's unique identity to be included in the attestation record.

Specific identifiers can be included in the attestation record, and an individual attestation certificate can be used to sign the attestation record. To find out if the device supports these features, refer to `isDeviceIdAttestationSupported()` (/reference/android/app/admin/DevicePolicyManager#isDeviceIdAttestationSupported()) and `isUniqueDeviceAttestationSupported()` (/reference/android/app/admin/DevicePolicyManager#isUniqueDeviceAttestationSupported()).

Device owner, profile owner, their delegated certificate installer and the credential management app can use `ID_TYPE_BASE_INFO` (/reference/android/app/admin/DevicePolicyManager#ID_TYPE_BASE_INFO) to request inclusion of the general device information including manufacturer, model, brand, device and product in the attestation record. Only device owner, profile owner on an organization-owned device or affiliated user, and their delegated certificate installers can use `ID_TYPE_SERIAL` (/reference/android/app/admin/DevicePolicyManager#ID_TYPE_SERIAL), `ID_TYPE_IMEI` (/reference/android/app/admin/DevicePolicyManager#ID_TYPE_IMEI) and `ID_TYPE_MEID` (/reference/android/app/admin/DevicePolicyManager#ID_TYPE_MEID) to request unique device identifiers to be attested (the serial number, IMEI and MEID correspondingly), if supported by the device (see `isDeviceIdAttestationSupported()` (/reference/android/app/admin/DevicePolicyManager#isDeviceIdAttestationSupported())). Additionally, device owner, profile owner on an organization-owned device and their delegated certificate installers can also request the attestation record to be signed using an individual attestation certificate by specifying the `ID_TYPE_INDIVIDUAL_ATTESTATION` (/reference/android/app/admin/DevicePolicyManager#ID_TYPE_INDIVIDUAL_ATTESTATION) flag (if supported by the device, see `isUniqueDeviceAttestationSupported()` (/reference/android/app/admin/DevicePolicyManager#isUniqueDeviceAttestationSupported())).

If any of `ID_TYPE_SERIAL` (/reference/android/app/admin/DevicePolicyManager#ID_TYPE_SERIAL), `ID_TYPE_IMEI` (/reference/android/app/admin/DevicePolicyManager#ID_TYPE_IMEI) and `ID_TYPE_MEID` (/reference/android/app/admin/DevicePolicyManager#ID_TYPE_MEID) is set, it is implicitly assumed

that `ID_TYPE_BASE_INFO` (/reference/android/app/admin/DevicePolicyManager#ID_TYPE_BASE_INFO) is also set.

Attestation using `ID_TYPE_INDIVIDUAL_ATTESTATION` (/reference/android/app/admin/DevicePolicyManager#ID_TYPE_INDIVIDUAL_ATTESTATION) can only be requested if key generation is done in StrongBox.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is assoc `null` if the caller is not a device admin.
`algorithm`	`String`: The key generation algorithm, see `KeyPairGenerator` (/reference/java/security/KeyPairGenerator). This value cannot be `null`.
`keySpec`	`KeyGenParameterSpec`: Specification of the key to generate, see `KeyPai` (/reference/java/security/KeyPairGenerator). This value cannot be `null`.
`idAttestationFlags`	`int`: A bitmask of the identifiers that should be included in the attestation r `TYPE_BASE_INFO`, `ID_TYPE_SERIAL`, `ID_TYPE_IMEI` and `ID_TYPE_MEI` `TYPE_INDIVIDUAL_ATTESTATION` if the attestation record should be sign individual attestation certificate. `0` should be passed in if no device identification is required in the attestatic the batch attestation certificate should be used. If any flag is specified, then an attestation challenge must be included in th Value is either `0` or a combination of `ID_TYPE_BASE_INFO` (/reference/android/app/admin/DevicePolicyManager#ID_TYPE_BASE_INFO `ID_TYPE_SERIAL` (/reference/android/app/admin/DevicePolicyManager#ID `ID_TYPE_IMEI` (/reference/android/app/admin/DevicePolicyManager#ID_T `ID_TYPE_MEID` (/reference/android/app/admin/DevicePolicyManager#ID_T `ID_TYPE_INDIVIDUAL_ATTESTATION` (/reference/android/app/admin/DevicePolicyManager#ID_TYPE_INDIVIDUA

Returns

AttestedKeyPair (/reference/android/security/AttestedKeyPair)	A non-null `AttestedKeyPair` if the key generation succeeded, null otherwise.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not `null` and not a device but the calling application is not a del credential management app. If Device ID_TYPE_SERIAL (/reference/android/app/admin/Devic , ID_TYPE_IMEI (/reference/android/app/admin/Devic ID_TYPE_MEID (/reference/android/app/admin/Devic the caller must be the Device Owner delegate.
IllegalArgumentException (/reference/java/lang/IllegalArgumentException)	in the following cases: • The alias in `keySpec` is empty. • The algorithm specification in `key` `RSAKeyGenParameterSpec` or `EC` • Device ID attestation was requeste contain an attestation challenge.
UnsupportedOperationException (/reference/java/lang/UnsupportedOperationException)	if Device ID attestation or individual at underlying hardware does not suppor
StrongBoxUnavailableException (/reference/android/security/keystore/StrongBoxUnavailableException)	if the use of StrongBox for key genera but the device does not have one.

See also:

`KeyGenParameterSpec.Builder.setAttestationChallenge(byte[])` (/reference/android/security/keystore/KeyGenParameterSpec.Builder#setAttestationChallenge(byte[]))

getAccountTypesWithManagementDisabled

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public String[] (/reference/java/lang/String) getAccountTypesWithManagementDisabled ()
```

Gets the array of accounts for which account management is disabled by the profile owner or device owner.

Account management can be disabled/enabled by calling `setAccountManagementDisabled(ComponentName, String, boolean)` (/reference/android/app/admin/DevicePolicyManager#setAccountManagementDisabled(android.content.ComponentName,%20java.lang.String,%20boolean)) .

This method may be called on the `DevicePolicyManager` instance returned from `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) . Note that only a profile owner on an organization-owned device can affect account types on the parent profile instance.

Returns

`String[]` (/reference/java/lang/String)	a list of account types for which account management has been disabled. This value may be `null`.

See also:

`setAccountManagementDisabled(ComponentName, String, boolean)` (/reference/android/app/admin/DevicePolicyManager#setAccountManagementDisabled(android.content.ComponentName,%20java.lang.String,%20boolean))

getActiveAdmins

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<ComponentName (/reference/android/content/ComponentName)>
```

Return a list of all currently active device administrators' component names. If there are no administrators `null` may be returned.

Returns

List (/reference/java/util/List)
<ComponentName
(/reference/android/content/ComponentName)
>

getAffiliationIds

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public Set (/reference/java/util/Set)<String (/reference/java/lang/String)> getAffiliationIds (Com
```

Returns the set of affiliation ids previously set via `setAffiliationIds(ComponentName, Set)` (/reference/android/app/admin/DevicePolicyManager#setAffiliationIds(android.content.ComponentName,%20java.util.Set<java.lang.String>))
, or an empty set if none have been set.

Parameters

`admin`	`ComponentName`: This value cannot be `null`.

Returns

Set (/reference/java/util/Set)<String (/reference/java/lang/String)>	This value cannot be null.

getAlwaysOnVpnLockdownWhitelist

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public Set (/reference/java/util/Set)<String (/reference/java/lang/String)> getAlwaysOnVpnLockdown
```

Called by device or profile owner to query the set of packages that are allowed to access the network directly when always-on VPN is in lockdown mode but not connected. Returns `null` when always-on VPN is not active or not in lockdown mode.

Parameters

admin	ComponentName: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be null.

Returns

Set (/reference/java/util/Set)<String (/reference/java/lang/String)>

Throws

SecurityException (/reference/java/lang/SecurityException)	if admin is not a device or a profile owner.

See also:

`setAlwaysOnVpnPackage(ComponentName, String, boolean, Set)`
(/reference/android/app/admin/DevicePolicyManager#setAlwaysOnVpnPackage(android.content.ComponentName,%20java.lang.String,%20boolean,%20java.util.Set<java.lang.String>))

getAlwaysOnVpnPackage

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public String (/reference/java/lang/String) getAlwaysOnVpnPackage (ComponentName (/reference/a
```

Called by a device or profile owner to read the name of the package administering an always-on VPN connection for the current user. If there is no such package, or the always-on VPN is provided by the system instead of by an application, `null` will be returned.

Parameters

`admin`	`ComponentName`: This value cannot be `null`.

Returns

`String` (/reference/java/lang/String)	Package name of VPN controller responsible for always-on VPN, or `null` if none is set.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or a profile owner.

getApplicationRestrictions

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public Bundle (/reference/android/os/Bundle) getApplicationRestrictions (ComponentName (/re
                String (/reference/java/lang/String) packageName)
```

Retrieves the application restrictions for a given target application running in the calling user.

The caller must be a profile or device owner on that user, or the package allowed to manage application restrictions via `setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>)) with the `DELEGATION_APP_RESTRICTIONS` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_APP_RESTRICTIONS) scope; otherwise a security exception will be thrown.

NOTE: The method performs disk I/O and shouldn't be called on the main thread
This method may take several seconds to complete, so it should only be called from a worker thread.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if called by the application restrictions managing package.
`packageName`	`String`: The name of the package to fetch restricted settings of.

Returns

`Bundle` (/reference/android/os/Bundle)	`Bundle` (/reference/android/os/Bundle) of settings corresponding to what v (/reference/android/app/admin/DevicePolicyManager#setApplicationRestri was called, or an empty `Bundle` (/reference/android/os/Bundle) if no restric

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device or profile owner.

See also:

setDelegatedScopes(ComponentName, String, List) (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

DELEGATION_APP_RESTRICTIONS (/reference/android/app/admin/DevicePolicyManager#DELEGATION_APP_RESTRICTIONS)

getApplicationRestrictionsManagingPackage

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public String (/reference/java/lang/String) getApplicationRestrictionsManagingPackage (Com
```

This method was deprecated in API level 26.

From **Build.VERSION_CODES.O** (/reference/android/os/Build.VERSION_CODES#O). Use **getDelegatePackages(ComponentName, String)** (/reference/android/app/admin/DevicePolicyManager#getDelegatePackages(android.content.ComponentName,%20java.lang.String)) with the **DELEGATION_APP_RESTRICTIONS** (/reference/android/app/admin/DevicePolicyManager#DELEGATION_APP_RESTRICTIONS) scope instead.

Called by a profile owner or device owner to retrieve the application restrictions managing package for the current user, or `null` if none is set. If there are multiple delegates this function will return one of them.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

`String` (/reference/java/lang/String)	The package name allowed to manage application restrictions on the current user, or `null` if none is set.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner.

getAutoTimeEnabled

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean getAutoTimeEnabled (ComponentName (/reference/android/content/ComponentNam
```

Returns true if auto time is enabled on the device.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.

Returns

`boolean`	true if auto time is enabled on the device.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if caller is not a device owner, a profile owner for the primary user, or a profile owner of an organization-owned managed profile.

getAutoTimeRequired

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean getAutoTimeRequired ()
```

This method was deprecated in API level 30.

From **`Build.VERSION_CODES.R`** (/reference/android/os/Build.VERSION_CODES#R). Use **`getAutoTimeEnabled(ComponentName)`** (/reference/android/app/admin/DevicePolicyManager#getAutoTimeEnabled(android.content.ComponentName))

Returns

`boolean`	true if auto time is required.

getAutoTimeZoneEnabled

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean getAutoTimeZoneEnabled (ComponentName (/reference/android/content/Componer
```

Returns true if auto time zone is enabled on the device.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.

Returns

`boolean`	true if auto time zone is enabled on the device.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if caller is not a device owner, a profile owner for the primary user, or a profile owner of an organization-owned managed profile.

getBindDeviceAdminTargetUsers

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<UserHandle (/reference/android/os/UserHandle)> getBindDevic
```

Returns the list of target users that the calling device owner or owner of secondary user can use when calling `bindDeviceAdminServiceAsUser(ComponentName, Intent, ServiceConnection, BindServiceFlags, UserHandle)` (/reference/android/app/admin/DevicePolicyManager#bindDeviceAdminServiceAsUser(android.content.ComponentName,%20android.content.Intent,%20android.content.ServiceConnection,%20android.content.Context.BindServiceFlags,%20android.os.UserHandle)).

A device owner can bind to a service from a secondary managed user and vice versa, provided that both users are affiliated. See `setAffiliationIds(ComponentName, Set)` (/reference/android/app/admin/DevicePolicyManager#setAffiliationIds(android.content.ComponentName,%20java.util.Set<java.lang.String>)) .

Parameters

`admin`	`ComponentName`: This value cannot be `null`.

Returns

`List` (/reference/java/util/List)<`UserHandle` (/reference/android/os/UserHandle)>	This value cannot be `null`.

getBluetoothContactSharingDisabled

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean getBluetoothContactSharingDisabled (ComponentName (/reference/android/co
```

Called by a profile owner of a managed profile to determine whether or not Bluetooth devices cannot access enterprise contacts.

The calling device admin must be a profile owner. If it is not, a security exception will be thrown.

This API works on managed profile only.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

`boolean`

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a profile owner.

getCameraDisabled

Added in API level 14 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean getCameraDisabled (ComponentName (/reference/android/content/ComponentName
```

Determine whether or not the device's cameras have been disabled for this user, either by the calling admin, if specified, or all admins.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance, returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) , where the caller must be the profile owner of an organization-owned managed profile.

Parameters

`admin`	`ComponentName`: The name of the admin component to check, or `null` to check whether any admins have disabled the camera

Returns

`boolean`

getCertInstallerPackage

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public String (/reference/java/lang/String) getCertInstallerPackage (ComponentName (/referenc
```

This method was deprecated in API level 26.

From `Build.VERSION_CODES.O` (/reference/android/os/Build.VERSION_CODES#O). Use `getDelegatePackages(ComponentName, String)` (/reference/android/app/admin/DevicePolicyManager#getDelegatePackages(android.content.ComponentName,%20java.lang.String)) with the `DELEGATION_CERT_INSTALL` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_INSTALL) scope instead.

Called by a profile owner or device owner to retrieve the certificate installer for the user, or `null` if none is set. If there are multiple delegates this function will return one of them.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

`String` (/reference/java/lang/String)	The package name of the current delegated certificate installer, or `null` if none is set.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or a profile owner.

getContentProtectionPolicy

Added in Android VanillaIceCream (/preview)

```
public int getContentProtectionPolicy (ComponentName (/reference/android/content/Componer
```

Returns the current content protection policy.

The returned policy will be the current resolved policy rather than the policy set by the calling admin.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.

Returns

`int`	Value is `CONTENT_PROTECTION_NOT_CONTROLLED_BY_POLICY` (/reference/android/app/admin/DevicePolicyManager#CONTENT_PROTECT , `CONTENT_PROTECTION_DISABLED` (/reference/android/app/admin/DevicePolicyManager#CONTENT_PROTECT `PROTECTION_ENABLED` (/reference/android/app/admin/DevicePolicyManager#CONTENT_PROTECT

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if **`admin`** is not the device owner, the profile owner of an affiliated us owner when no device owner is set or holder of the permission `Man MANAGE_DEVICE_POLICY_CONTENT_PROTECTION` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC .

See also:

`isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser())

`setContentProtectionPolicy(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setContentProtectionPolicy(android.content.Compon entName,%20int))

getCredentialManagerPolicy

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public PackagePolicy (/reference/android/app/admin/PackagePolicy) getCredentialManagerPolic
```

Called by a device owner or profile owner of a managed profile to retrieve the credential manager policy.

Returns

PackagePolicy (/reference/android/app/admin/PackagePolicy)	the current credential manager policy if null then this policy has not been configured.

Throws

SecurityException (/reference/java/lang/SecurityException)	if caller is not a device owner or profile owner of a managed profile.

getCrossProfileCalendarPackages

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public Set (/reference/java/util/Set)<String (/reference/java/lang/String)> getCrossProfileCalenda
```

This method was deprecated in API level 34.

Use `setCrossProfilePackages(android.content.ComponentName, java.util.Set)` (/reference/android/app/admin/DevicePolicyManager#setCrossProfilePackages(android.content.ComponentName,%20java.util.Set<java.lang.String>))

.

Gets a set of package names that are allowed to access cross-profile calendar APIs.

Called by a profile owner of a managed profile.

Parameters

admin	ComponentName: which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with This value cannot be null.

Returns

Set (/reference/java/util/Set)<String (/reference/java/lang/String)>	the set of names of packages that were previously allowed via setCrossP util.Set) (/reference/android/app/admin/DevicePolicyManager#setCrossProfileCale , or an empty set if none have been allowed This value may be **null**.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a profile owner

See also:

`setCrossProfileCalendarPackages(ComponentName, Set)` (/reference/android/app/admin/DevicePolicyManager#setCrossProfileCalendarPackages(android.content.ComponentName,%20java.util.Set<java.lang.String>))

getCrossProfileCallerIdDisabled

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean getCrossProfileCallerIdDisabled (ComponentName (/reference/android/conten
```

This method was deprecated in API level 34.
starting with **Build.VERSION_CODES.UPSIDE_DOWN_CAKE** (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), use **getManagedProfileCallerIdAccessPolicy()** (/reference/android/app/admin/DevicePolicyManager#getManagedProfileCallerIdAccessPolicy()) instead

Called by a profile owner of a managed profile to determine whether or not caller-Id information has been disabled.

The calling device admin must be a profile owner. If it is not, a security exception will be thrown.

Starting with `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), this will return true when `setManagedProfileCallerIdAccessPolicy(android.app.admin.PackagePolicy)` (/reference/android/app/admin/DevicePolicyManager#setManagedProfileCallerIdAccessPolicy(android.app.admin.PackagePolicy)) has been set with a non-null policy whose policy type is NOT `PackagePolicy#PACKAGE_POLICY_BLOCKLIST` (/reference/android/app/admin/PackagePolicy#PACKAGE_POLICY_BLOCKLIST)

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

`boolean`

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a profile owner.

getCrossProfileContactsSearchDisabled

Deprecated in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean getCrossProfileContactsSearchDisabled (ComponentName (/reference/androic
```

This method was deprecated in API level 34.

From `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE) use `getManagedProfileContactsAccessPolicy()` (/reference/android/app/admin/DevicePolicyManager#getManagedProfileContactsAccessPolicy())

Called by a profile owner of a managed profile to determine whether or not contacts search has been disabled.

The calling device admin must be a profile owner. If it is not, a security exception will be thrown.

Starting with `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), this will return true when `setManagedProfileContactsAccessPolicy(android.app.admin.PackagePolicy)` (/reference/android/app/admin/DevicePolicyManager#setManagedProfileContactsAccessPolicy(android.app.admin.PackagePolicy)) has been set with a non-null policy whose policy type is NOT `PackagePolicy#PACKAGE_POLICY_BLOCKLIST` (/reference/android/app/admin/PackagePolicy#PACKAGE_POLICY_BLOCKLIST)

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

`boolean`

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not a profile owner.

getCrossProfilePackages

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public Set (/reference/java/util/Set)<String (/reference/java/lang/String)> getCrossProfilePackage
```

Returns the set of package names that the admin has previously set as allowed to request user consent for cross-profile communication, via `setCrossProfilePackages(android.content.ComponentName, java.util.Set)` (/reference/android/app/admin/DevicePolicyManager#setCrossProfilePackages(android.content.ComponentName,%20java.util.Set<java.lang.String>)) .

Assumes that the caller is a profile owner and is the given `admin`.

Note that other apps not included in the returned set may be able to request user consent for cross-profile communication if they have been explicitly allowlisted by the OEM.

Parameters

`admin`	`ComponentName`: the DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with This value cannot be `null`.

Returns

Set (/reference/java/util/Set) <String (/reference/java/lang/String)>	the set of package names the admin has previously set as allowed to reque setCrossProfilePackages(android.content.ComponentName, ja (/reference/android/app/admin/DevicePolicyManager#setCrossProfilePack This value cannot be `null`.

getCrossProfileWidgetProviders

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<String (/reference/java/lang/String)> getCrossProfileWidget
```

Called by the profile owner of a managed profile or a holder of the permission `Manifest.permission.MANAGE_DEVICE_POLICY_PROFILE_INTERACTION` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_PROFILE_INTERACTION) to query providers from which packages are available in the parent profile.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.

Returns

`List` (/reference/java/util/List)<`String` (/reference/java/lang/String)>	The allowlisted package list. This value cannot be `null`.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a profile owner and not a holder of the permission `MANAGE_DEVICE_POLICY_PROFILE_INTERACTION` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC.

See also:

`addCrossProfileWidgetProvider(android.content.ComponentName, String)` (/reference/android/app/admin/DevicePolicyManager#addCrossProfileWidgetProvider(android.content.ComponentName,%20java.lang.String))

`removeCrossProfileWidgetProvider(android.content.ComponentName, String)` (/reference/android/app/admin/DevicePolicyManager#removeCrossProfileWidgetProvider(android.content.ComponentName,%20java.lang.String))

getCurrentFailedPasswordAttempts

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getCurrentFailedPasswordAttempts ()
```

Retrieve the number of times the user has failed at entering a password since that last successful password entry.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to retrieve the number of failed password attemts for the parent user.

The calling device admin must have requested `DeviceAdminInfo#USES_POLICY_WATCH_LOGIN` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_WATCH_LOGIN) to be able to call this method; if it has not, a security exception will be thrown.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always empty and this method always returns 0.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Returns

`int`	The number of times user has entered an incorrect password since the last correct password entry.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the calling application does not own an active administrator that u `DeviceAdminInfo#USES_POLICY_WATCH_LOGIN` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_W

getDelegatePackages Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<String (/reference/java/lang/String)> getDelegatePackages (
            String (/reference/java/lang/String) delegationScope)
```

Called by a profile owner or device owner to retrieve a list of delegate packages that were granted a delegation scope.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`delegationScope`	`String`: The scope whose delegates should be retrieved. This value cannot be `null`.

Returns

List (/reference/java/util/List)<String (/reference/java/lang/String)>	A list of package names of the current delegated packages for **delegationScope**. This value may be **null**.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device or a profile owner.

getDelegatedScopes

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<String (/reference/java/lang/String)> getDelegatedScopes (C
                String (/reference/java/lang/String) delegatedPackage)
```

Called by a profile owner or device owner to retrieve a list of the scopes given to a delegate package. Other apps can use this method to retrieve their own delegated scopes by passing `null` for `admin` and their own package name as `delegatedPackage`.

Parameters

`admin`	`ComponentName`: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or **null** if the caller is **delegatedPackage**.
`delegatedPackage`	`String`: The package name of the app whose scopes should be retrieved. This value cannot be **null**.

Returns

List (/reference/java/util/List)<String (/reference/java/lang/String)>	A list containing the scopes given to delegatedPackage.

Throws

SecurityException (/reference/java/lang/SecurityException)	if admin is not a device or a profile owner.

getDeviceOwnerLockScreenInfo

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public CharSequence (/reference/java/lang/CharSequence) getDeviceOwnerLockScreenInfo ()
```

Returns

CharSequence (/reference/java/lang/CharSequence)	The device owner information. If it is not set returns null.

getDevicePolicyManagementRoleHolderPackage

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public String (/reference/java/lang/String) getDevicePolicyManagementRoleHolderPackage ()
```

Returns the package name of the device policy management role holder.

If the device policy management role holder is not configured for this device, returns `null`.

Returns

String (/reference/java/lang/String)

getEndUserSessionMessage

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public CharSequence (/reference/java/lang/CharSequence) getEndUserSessionMessage (Compone
```

Returns the user session end message.

Parameters

`admin`	`ComponentName`: which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

CharSequence (/reference/java/lang/CharSequence)

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not a device owner.

getEnrollmentSpecificId

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public String (/reference/java/lang/String) getEnrollmentSpecificId ()
```

Returns an enrollment-specific identifier of this device, which is guaranteed to be the same value for the same device, enrolled into the same organization by the same managing app. This identifier is high-entropy, useful for uniquely identifying individual devices within the same organisation. It is available both in a work profile and on a fully-managed device. The identifier would be consistent even if the work profile is removed and enrolled again (to the same organization), or the device is factory reset and re-enrolled. Can only be called by the Profile Owner and Device Owner, and starting from Android `ERROR(/android.os.Build.VERSION_CODES#VANILLA_ICE_CREAM)` (/), holders of the permission `Manifest.permission.MANAGE_DEVICE_POLICY_CERTIFICATES` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_CERTIFICATES). If `setOrganizationId(java.lang.String)` (/reference/android/app/admin/DevicePolicyManager#setOrganizationId(java.lang.String)) was not called, then the returned value will be an empty string.

Note about access to device identifiers: a device owner, a profile owner of an organization-owned device or the delegated certificate installer (holding the `DELEGATION_CERT_INSTALL` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_INSTALL) delegation) on such a device can still obtain hardware identifiers by calling e.g. `Build.getSerial()` (/reference/android/os/Build#getSerial()), in addition to using this method. However, a profile owner on a personal (non organization-owned) device, or the delegated certificate installer on such a device, cannot obtain hardware identifiers anymore and must switch to using this method.

Returns

String (/reference/java/lang/String)	A stable, enrollment-specific identifier. This value cannot be **null**.

Throws

SecurityException (/reference/java/lang/SecurityException)	if the caller is not a profile owner, device owner or holding the Mani permission.MANAGE_DEVICE_POLICY_CERTIFICATES (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC permission

getFactoryResetProtectionPolicy

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public FactoryResetProtectionPolicy (/reference/android/app/admin/FactoryResetProtectionPolicy)
```

Callable by device owner or profile owner of an organization-owned device, to retrieve the current factory reset protection (FRP) policy set previously by `setFactoryResetProtectionPolicy(ComponentName, FactoryResetProtectionPolicy)` (/reference/android/app/admin/DevicePolicyManager#setFactoryResetProtectionPolicy(android.content.ComponentName,%20android.app.admin.FactoryResetProtectionPolicy)).

This method can also be called by the FRP management agent on device or with the permission `Manifest.permission.MASTER_CLEAR` (/reference/android/Manifest.permission#MASTER_CLEAR), in which case, it can pass `null` as the ComponentName.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with or `null` if the caller is not a device admin

Returns

`FactoryResetProtectionPolicy` (/reference/android/app/admin/FactoryResetProtectionPolicy)	The current FRP policy object or `null` if no policy is set.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device owner, a profile owner of an organization-owned device or the FRP management agent.

UnsupportedOperationException (/reference/java/lang/UnsupportedOperationException)	if factory reset protection is not supported on the device.

getGlobalPrivateDnsHost

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public String (/reference/java/lang/String) getGlobalPrivateDnsHost (ComponentName (/referenc
```

Returns the system-wide Private DNS host.

Parameters

admin	ComponentName: which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be null.

Returns

String (/reference/java/lang/String)	The hostname used for Private DNS queries, null if none is set.

Throws

SecurityException (/reference/java/lang/SecurityException)	if the caller is not the device owner.

getGlobalPrivateDnsMode

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getGlobalPrivateDnsMode (ComponentName (/reference/android/content/ComponentNa
```

Returns the system-wide Private DNS mode.

Parameters

`admin`	`ComponentName`: which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

`int`	one of `PRIVATE_DNS_MODE_OFF`, `PRIVATE_DNS_MODE_OPPORTUNISTIC`, `PRIVATE_DNS_MODE_PROVIDER_HOSTNAME` or `PRIVATE_DNS_MODE_UNKNOWN`.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not the device owner.

getInstalledCaCerts

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<byte[]> getInstalledCaCerts (ComponentName (/reference
```

Returns all CA certificates that are currently trusted, excluding system CA certificates. If a user has installed any certificates by other means than device policy these will be included too.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if calling from a delegated certificate installer.

Returns

`List` (/reference/java/util/List)`<byte[]>`	a List of byte[] arrays, each encoding one user CA certificate. This value cannot be `null`.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not `null` and not a device or profile owner.

getKeepUninstalledPackages

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<String (/reference/java/lang/String)> getKeepUninstalledPac
```

Get the list of apps to keep around as APKs even if no user has currently installed it. This function can be called by a device owner or by a delegate given the `DELEGATION_KEEP_UNINSTALLED_PACKAGES` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_KEEP_UNINSTALLED_PACKAGES) scope via `setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>)).

Please note that packages returned in this method are not automatically pre-cached.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if the caller is a keep uninstalled packages delegate.

Returns

`List` (/reference/java/util/List) `<String` (/reference/java/lang/String)`>`	List of package names to keep cached. This value may be `null`.

See also:

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`DELEGATION_KEEP_UNINSTALLED_PACKAGES` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_KEEP_UNINSTALLED_PACKAGES)

getKeyPairGrants

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public Map (/reference/java/util/Map)<Integer (/reference/java/lang/Integer), Set (/reference/java/util/S
```

Called by a device or profile owner, or delegated certificate chooser (an app that has been delegated the `DELEGATION_CERT_SELECTION` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_SELECTION) privilege), to query which apps have access to a given KeyChain key. Key are granted on a per-UID basis, so if several apps share the same UID, granting access to one of them automatically grants it to others. This method returns a map containing one entry per grantee UID. Entries have UIDs as keys and sets of corresponding package names as values. In particular, grantee packages that

don't share UID with other packages are represented by entries having singleton sets as values.

Parameters

`alias`	`String`: The alias of the key to grant access to. This value cannot be `null`.

Returns

`Map` (/reference/java/util/Map)`<Integer` (/reference/java/lang/Integer)`, Set` (/reference/java/util/Set)`<String` (/reference/java/lang/String)`>>`	apps that have access to a given key, arranged in a map from UID to sets of package names. This value cannot be `null`.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not a device owner, a profile owner or delegated certificate chooser.
`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if `alias` doesn't correspond to an existing key.

See also:

`grantKeyPairToApp(ComponentName, String, String)` (/reference/android/app/admin/DevicePolicyManager#grantKeyPairToApp(android.content.ComponentName,%20java.lang.String,%20java.lang.String))

getKeyguardDisabledFeatures

Added in API level 17 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getKeyguardDisabledFeatures (ComponentName (/reference/android/content/Compone
```

Determine whether or not features have been disabled in keyguard either by the calling admin, if specified, or all admins that set restrictions on this user and its participating profiles. Restrictions on profiles that have a separate challenge are not taken into account.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to retrieve restrictions on the parent profile.

Parameters

`admin`	`ComponentName`: The name of the admin component to check, or `null` to check whether any admins have disabled features in keyguard.

Returns

`int`	bitfield of flags. See `setKeyguardDisabledFeatures(android.conte` (/reference/android/app/admin/DevicePolicyManager#setKeyguardDisable for a list.

getLockTaskFeatures

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getLockTaskFeatures (ComponentName (/reference/android/content/ComponentName)
```

Gets which system features are enabled for LockTask mode.

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), the returned policy will be the current resolved policy rather than the policy set by the calling admin.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.

Returns

`int`	bitfield of flags. See `setLockTaskFeatures(android.content.Compo` (/reference/android/app/admin/DevicePolicyManager#setLockTaskFeature for a list. Value is either `0` or a combination of `LOCK_TASK_FEATURE_NONE` (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FEATUI `INFO` (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FE `FEATURE_NOTIFICATIONS` (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FEATUI `HOME` (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FE `OVERVIEW` (/reference/android/app/admin/DevicePolicyManager#LOCK_TA `FEATURE_GLOBAL_ACTIONS` (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FEATUI `FEATURE_KEYGUARD` (/reference/android/app/admin/DevicePolicyManager `LOCK_TASK_FEATURE_BLOCK_ACTIVITY_START_IN_TASK` (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FEATUI

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not the device owner, the profile owner of an affiliated us or the profile owner when no device owner is set or holder of the pe `Manifest.permission.MANAGE_DEVICE_POLICY_LOCK_TASK`

(/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC
.

See also:

`isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser())

`setLockTaskFeatures(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setLockTaskFeatures(android.content.ComponentName,%20int))

getLockTaskPackages

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public String[] (/reference/java/lang/String) getLockTaskPackages (ComponentName (/reference/a
```

Returns the list of packages allowed to start the lock task mode.

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), the returned policy will be the current resolved policy rather than the policy set by the calling admin.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.

Returns

`String[]` (/reference/java/lang/String)	This value cannot be `null`.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not the device owner, the profile owner of an affiliated us or the profile owner when no device owner is set or holder of the pe Manifest.permission.MANAGE_DEVICE_POLICY_LOCK_TASK (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC .

See also:

`isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser())

`setLockTaskPackages(ComponentName, String)` (/reference/android/app/admin/DevicePolicyManager#setLockTaskPackages(android.content.ComponentNa me,%20java.lang.String[]))

getLongSupportMessage

added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public CharSequence (/reference/java/lang/CharSequence) getLongSupportMessage (ComponentN
```

Called by a device admin to get the long support message.

Parameters

admin	`ComponentName`: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be **null**.

Returns

CharSequence (/reference/java/lang/CharSequence)	The message set by setLongSupportMessage(android.content. (/reference/android/app/admin/DevicePolicyManager#setLongSupport or null if no message has been set.

Throws

SecurityException (/reference/java/lang/SecurityException)	if admin is not an active administrator.

getManagedProfileCallerIdAccessPolicy

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public PackagePolicy (/reference/android/app/admin/PackagePolicy) getManagedProfileCallerId
```

Called by a profile owner of a managed profile to retrieve the caller id policy.

The calling device admin must be a profile owner of a managed profile. If it is not, a SecurityException (/reference/java/lang/SecurityException) will be thrown.

Returns

PackagePolicy (/reference/android/app/admin/PackagePolicy)	the current caller id policy This value may be null.

Throws

SecurityException (/reference/java/lang/SecurityException)	if caller is not a profile owner of a managed profile.

getManagedProfileContactsAccessPolicy

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public PackagePolicy (/reference/android/app/admin/PackagePolicy) getManagedProfileContacts
```

Called by a profile owner of a managed profile to determine the current policy applied to managed profile contacts.

The calling device admin must be a profile owner of a managed profile. If it is not, a `SecurityException` (/reference/java/lang/SecurityException) will be thrown.

Returns

PackagePolicy (/reference/android/app/admin/PackagePolicy)	the current contacts search policy This value may be **null**.

Throws

SecurityException (/reference/java/lang/SecurityException)	if caller is not a profile owner of a managed profile.

getManagedProfileMaximumTimeOff

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public long getManagedProfileMaximumTimeOff (ComponentName (/reference/android/content/Co
```

Called by a profile owner of an organization-owned managed profile to get maximum time the profile is allowed to be turned off.

Parameters

admin	ComponentName: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is

associated with This value cannot be `null`.

Returns

`long`	Maximum time the profile is allowed to be off in milliseconds or 0 if not limited.

See also:

`setPersonalAppsSuspended(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setPersonalAppsSuspended(android.content.ComponentName,%20boolean))

getManagedSubscriptionsPolicy

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public ManagedSubscriptionsPolicy (/reference/android/app/admin/ManagedSubscriptionsPolicy) g
```

Returns the current `ManagedSubscriptionsPolicy` (/reference/android/app/admin/ManagedSubscriptionsPolicy). If the policy has not been set, it will return a default policy of Type `ManagedSubscriptionsPolicy.TYPE_ALL_PERSONAL_SUBSCRIPTIONS` (/reference/android/app/admin/ManagedSubscriptionsPolicy#TYPE_ALL_PERSONAL_SUBSCRIPTIONS).

Returns

`ManagedSubscriptionsPolicy` (/reference/android/app/admin/ManagedSubscriptionsPolicy)	This value cannot be `null`.

See also:

`setManagedSubscriptionsPolicy(ManagedSubscriptionsPolicy)` (/reference/android/app/admin/DevicePolicyManager#setManagedSubscriptionsPolicy(android.app.admin.ManagedSubscriptionsPolicy))

getMaximumFailedPasswordsForWipe

Added in API level 2 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getMaximumFailedPasswordsForWipe (ComponentName (/reference/android/content/Co
```

Retrieve the current maximum number of login attempts that are allowed before the device or profile is wiped, for a particular admin or all admins that set restrictions on this user and its participating profiles. Restrictions on profiles that have a separate challenge are not taken into account.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to retrieve the value for the parent profile.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always empty and this method returns a default value (0) indicating that the policy is not set.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Parameters

`admin`	`ComponentName`: The name of the admin component to check, or `null` to aggregate all admins.

Returns

`int`

getMaximumTimeToLock

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public long getMaximumTimeToLock (ComponentName (/reference/android/content/ComponentName
```

Retrieve the current maximum time to unlock for a particular admin or all admins that set restrictions on this user and its participating profiles. Restrictions on profiles that have a separate challenge are not taken into account.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to retrieve restrictions on the parent profile.

Parameters

`admin`	`ComponentName`: The name of the admin component to check, or `null` to aggregate all admins.

Returns

`long`	time in milliseconds for the given admin or the minimum value (strictest) of all admins if admin is null. Returns 0 if there are no restrictions.

getMeteredDataDisabledPackages

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<String (/reference/java/lang/String)> getMeteredDataDisable
```

Called by a device or profile owner to retrieve the list of packages which are restricted by the admin from using metered data.

Parameters

`admin`	`ComponentName`: which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

`List` (/reference/java/util/List)<`String` (/reference/java/lang/String)>	the list of restricted package names. This value cannot be `null`.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner.

getMinimumRequiredWifiSecurityLevel

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getMinimumRequiredWifiSecurityLevel ()
```

Returns the current Wi-Fi minimum security level.

Returns

`int`	Value is `WIFI_SECURITY_OPEN` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_OPE `SECURITY_PERSONAL` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_PER `WIFI_SECURITY_ENTERPRISE_EAP` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_ENT , or `WIFI_SECURITY_ENTERPRISE_192` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_ENT

See also:

`setMinimumRequiredWifiSecurityLevel(int)` (/reference/android/app/admin/DevicePolicyManager#setMinimumRequiredWifiSecurityLevel(int))

getMtePolicy

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getMtePolicy ()
```

Called by a device owner, profile owner of an organization-owned device to get the Memory Tagging Extension (MTE) policy Learn more about MTE (https://source.android.com/docs/security/test/memory-safety/arm-mte)

Returns

`int`	the currently set MTE policy Value is `MTE_ENABLED` (/reference/android/app/admin/DevicePolicyManager#MTE_ENABLED), `MT` (/reference/android/app/admin/DevicePolicyManager#MTE_DISABLED), or `CONTROLLED_BY_POLICY` (/reference/android/app/admin/DevicePolicyManager#MTE_NOT_CONTRO

Throws

SecurityException (/reference/java/lang/SecurityException)	if caller is not permitted to set Mte policy

getNearbyAppStreamingPolicy

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getNearbyAppStreamingPolicy ()
```

Returns the current runtime nearby app streaming policy set by the device or profile owner.

The caller must be the target user's device owner/profile owner or hold the READ_NEARBY_STREAMING_POLICY (/reference/android/Manifest.permission#READ_NEARBY_STREAMING_POLICY) permission.

Returns

int	Value is NEARBY_STREAMING_NOT_CONTROLLED_BY_POLICY (/reference/android/app/admin/DevicePolicyManager#NEARBY_STREAMIN NEARBY_STREAMING_DISABLED (/reference/android/app/admin/DevicePolicyManager#NEARBY_STREAMIN ENABLED (/reference/android/app/admin/DevicePolicyManager#NEARBY_S STREAMING_SAME_MANAGED_ACCOUNT_ONLY (/reference/android/app/admin/DevicePolicyManager#NEARBY_STREAMIN

getNearbyNotificationStreamingPolicy

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getNearbyNotificationStreamingPolicy ()
```

Returns the current runtime nearby notification streaming policy set by the device or profile owner.

The caller must be the target user's device owner/profile owner or hold the `READ_NEARBY_STREAMING_POLICY` (/reference/android/Manifest.permission#READ_NEARBY_STREAMING_POLICY) permission.

Returns

`int`	Value is `NEARBY_STREAMING_NOT_CONTROLLED_BY_POLICY` (/reference/android/app/admin/DevicePolicyManager#NEARBY_STREAMIN `NEARBY_STREAMING_DISABLED` (/reference/android/app/admin/DevicePolicyManager#NEARBY_STREAMIN `ENABLED` (/reference/android/app/admin/DevicePolicyManager#NEARBY_S `STREAMING_SAME_MANAGED_ACCOUNT_ONLY` (/reference/android/app/admin/DevicePolicyManager#NEARBY_STREAMIN

getOrganizationColor

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getOrganizationColor (ComponentName (/reference/android/content/ComponentName)
```

This method was deprecated in API level 31.
From `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R), the organization color is never used as the background color of the confirm credentials screen.

Called by a profile owner of a managed profile to retrieve the color used for customization. This color is used as background color of the confirm credentials screen for that user.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

`int`	The 24bit (0xRRGGBB) representation of the color to be used.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if **`admin`** is not a profile owner.

getOrganizationName Added in API level 24} (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public CharSequence (/reference/java/lang/CharSequence) getOrganizationName (ComponentNam
```

Called by the device owner (since API 26) or profile owner (since API 24) or holders of the permission {@link android.Manifest.permission#MANAGE_DEVICE_POLICY_ORGANIZATION_IDENTITY to retrieve the name of the organization under management.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin.

Returns

`CharSequence` (/reference/java/lang/CharSequence)	The organization name or **`null`** if none is set.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** if **admin** is not a device or profile owner or holder of the p permission.MANAGE_DEVICE_POLICY_ORGANIZATION_IDENTI (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC .

getOverrideApns

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<ApnSetting (/reference/android/telephony/data/ApnSetting)> ge
```

Called by device owner or managed profile owner to get all override APNs inserted by device owner or managed profile owner previously using `addOverrideApn(ComponentName, ApnSetting)` (/reference/android/app/admin/DevicePolicyManager#addOverrideApn(android.content.ComponentName, %20android.telephony.data.ApnSetting)) .

Parameters

admin	ComponentName: which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with This value cannot be **null**.

Returns

List (/reference/java/util/List)<ApnSetting (/reference/android/telephony/data/ApnSetting) >	A list of override APNs inserted by device owner.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device owner.

See also:

`setOverrideApnsEnabled(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setOverrideApnsEnabled(android.content.ComponentName,%20boolean))

getParentProfileInstance

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public DevicePolicyManager (/reference/android/app/admin/DevicePolicyManager) getParentProfi
```

Called by the profile owner of a managed profile to obtain a `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) whose calls act on the parent profile.

The following methods are supported for the parent instance, all other methods will throw a SecurityException when called on the parent instance:

- `getPasswordQuality(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPasswordQuality(android.content.ComponentName))
- `setPasswordQuality(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int))
- `getPasswordMinimumLength(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPasswordMinimumLength(android.content.ComponentName))
- `setPasswordMinimumLength(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumLength(android.content.ComponentName,%20int))

- `getPasswordMinimumUpperCase(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPasswordMinimumUpperCase(android.content.ComponentName))
- `setPasswordMinimumUpperCase(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumUpperCase(android.content.ComponentName,%20int))
- `getPasswordMinimumLowerCase(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPasswordMinimumLowerCase(android.content.ComponentName))
- `setPasswordMinimumLowerCase(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumLowerCase(android.content.ComponentName,%20int))
- `getPasswordMinimumLetters(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPasswordMinimumLetters(android.content.ComponentName))
- `setPasswordMinimumLetters(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumLetters(android.content.ComponentName,%20int))
- `getPasswordMinimumNumeric(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPasswordMinimumNumeric(android.content.ComponentName))
- `setPasswordMinimumNumeric(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumNumeric(android.content.ComponentName,%20int))
- `getPasswordMinimumSymbols(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPasswordMinimumSymbols(android.content.ComponentName))
- `setPasswordMinimumSymbols(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumSymbols(android.content.ComponentName,%20int))
- `getPasswordMinimumNonLetter(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPasswordMinimumNonLetter(android.content.ComponentName))
- `setPasswordMinimumNonLetter(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumNonLetter(android.content.ComponentName,%20int))

- `getPasswordHistoryLength(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPasswordHistoryLength(android.content.ComponentName))
- `setPasswordHistoryLength(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordHistoryLength(android.content.ComponentName,%20int))
- `getPasswordExpirationTimeout(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPasswordExpirationTimeout(android.content.ComponentName))
- `setPasswordExpirationTimeout(ComponentName, long)` (/reference/android/app/admin/DevicePolicyManager#setPasswordExpirationTimeout(android.content.ComponentName,%20long))
- `getPasswordExpiration(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPasswordExpiration(android.content.ComponentName))
- `getPasswordMaximumLength(int)` (/reference/android/app/admin/DevicePolicyManager#getPasswordMaximumLength(int))
- `isActivePasswordSufficient()` (/reference/android/app/admin/DevicePolicyManager#isActivePasswordSufficient())
- `getCurrentFailedPasswordAttempts()` (/reference/android/app/admin/DevicePolicyManager#getCurrentFailedPasswordAttempts())
- `getMaximumFailedPasswordsForWipe(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getMaximumFailedPasswordsForWipe(android.content.ComponentName))
- `setMaximumFailedPasswordsForWipe(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setMaximumFailedPasswordsForWipe(android.content.ComponentName,%20int))
- `getMaximumTimeToLock(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getMaximumTimeToLock(android.content.ComponentName))
- `setMaximumTimeToLock(ComponentName, long)` (/reference/android/app/admin/DevicePolicyManager#setMaximumTimeToLock(android.content.ComponentName,%20long))
- `lockNow()` (/reference/android/app/admin/DevicePolicyManager#lockNow())

- `getKeyguardDisabledFeatures(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getKeyguardDisabledFeatures(android.content.ComponentName))
- `setKeyguardDisabledFeatures(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setKeyguardDisabledFeatures(android.content.ComponentName,%20int))
- `getTrustAgentConfiguration(ComponentName, ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getTrustAgentConfiguration(android.content.ComponentName,%20android.content.ComponentName))
- `setTrustAgentConfiguration(ComponentName, ComponentName, PersistableBundle)` (/reference/android/app/admin/DevicePolicyManager#setTrustAgentConfiguration(android.content.ComponentName,%20android.content.ComponentName,%20android.os.PersistableBundle))
- `getRequiredStrongAuthTimeout(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getRequiredStrongAuthTimeout(android.content.ComponentName))
- `setRequiredStrongAuthTimeout(ComponentName, long)` (/reference/android/app/admin/DevicePolicyManager#setRequiredStrongAuthTimeout(android.content.ComponentName,%20long))
- `getAccountTypesWithManagementDisabled()` (/reference/android/app/admin/DevicePolicyManager#getAccountTypesWithManagementDisabled())
- `setRequiredPasswordComplexity(int)` (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswordComplexity(int))
- `getRequiredPasswordComplexity()` (/reference/android/app/admin/DevicePolicyManager#getRequiredPasswordComplexity())

The following methods are supported for the parent instance but can only be called by the profile owner of a managed profile that was created during the device provisioning flow:

- `getPasswordComplexity()` (/reference/android/app/admin/DevicePolicyManager#getPasswordComplexity())
- `setCameraDisabled(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setCameraDisabled(android.content.ComponentName,%20boolean))

- `getCameraDisabled(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getCameraDisabled(android.content.ComponentName))
- `setAccountManagementDisabled(android.content.ComponentName, java.lang.String, boolean)` (/reference/android/app/admin/DevicePolicyManager#setAccountManagementDisabled(android.content.ComponentName,%20java.lang.String,%20boolean))
- `setPermittedInputMethods(ComponentName, List)` (/reference/android/app/admin/DevicePolicyManager#setPermittedInputMethods(android.content.ComponentName,%20java.util.List<java.lang.String>))
- `getPermittedInputMethods(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPermittedInputMethods(android.content.ComponentName))

The following methods can be called by the profile owner of a managed profile on an organization-owned device:

- `wipeData(int)` (/reference/android/app/admin/DevicePolicyManager#wipeData(int))

Parameters

`admin`	`ComponentName`: This value cannot be `null`.

Returns

`DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager)	a new instance of `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) that acts on the parent profile. This value cannot be `null`.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a profile owner.

getPasswordComplexity

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getPasswordComplexity ()
```

Returns how complex the current user's screen lock is.

Note that when called from a profile which uses an unified challenge with its parent, the screen lock complexity of the parent will be returned.

Apps need the `permission#REQUEST_PASSWORD_COMPLEXITY` (/reference/android/Manifest.permission#REQUEST_PASSWORD_COMPLEXITY) permission to call this method. On Android `Build.VERSION_CODES.S` (/reference/android/os/Build.VERSION_CODES#S) and above, the calling application does not need this permission if it is a device owner or a profile owner.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to retrieve restrictions on the parent profile.

Returns

`int`	Value is `PASSWORD_COMPLEXITY_NONE` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPL `PASSWORD_COMPLEXITY_LOW` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPL `PASSWORD_COMPLEXITY_MEDIUM` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPL , or `PASSWORD_COMPLEXITY_HIGH` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPL

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if the user is not unlocked.
`SecurityException` (/reference/java/lang/SecurityException)	if the calling application does not have the permission `permissi` `PASSWORD_COMPLEXITY` (/reference/android/Manifest.permission#REQUEST_PASSWORD , and is not a device owner or a profile owner.

getPasswordExpiration Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public long getPasswordExpiration (ComponentName (/reference/android/content/ComponentNam
```

Get the current password expiration time for a particular admin or all admins that set restrictions on this user and its participating profiles. Restrictions on profiles that have a separate challenge are not taken into account. If admin is `null`, then a composite of all expiration times is returned - which will be the minimum of all of them.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to retrieve the password expiration for the parent profile.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password expiration is always disabled and this method always returns 0.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Parameters

`admin`	**`ComponentName`**: The name of the admin component to check, or **`null`** to aggregate all admins.

Returns

`long`	The password expiration time, in milliseconds since epoch.

getPasswordExpirationTimeout

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public long getPasswordExpirationTimeout (ComponentName (/reference/android/content/Compo
```

Get the password expiration timeout for the given admin. The expiration timeout is the recurring expiration timeout provided in the call to `setPasswordExpirationTimeout(android.content.ComponentName, long)` (/reference/android/app/admin/DevicePolicyManager#setPasswordExpirationTimeout(android.content.ComponentName,%20long)) for the given admin or the aggregate of all participating policy administrators if `admin` is null. Admins that have set restrictions on profiles that have a separate challenge are not taken into account.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to retrieve restrictions on the parent profile.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password expiration is always disabled and this method always returns 0.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN`

(/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Parameters

`admin`	**`ComponentName`**: The name of the admin component to check, or **`null`** to aggregate all admins.

Returns

`long`	The timeout for the given admin or the minimum of all timeouts

getPasswordHistoryLength

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getPasswordHistoryLength (ComponentName (/reference/android/content/ComponentN
```

Retrieve the current password history length for a particular admin or all admins that set restrictions on this user and its participating profiles. Restrictions on profiles that have a separate challenge are not taken into account.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to retrieve restrictions on the parent profile.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password history length is always 0.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN`

(/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Parameters

`admin`	**`ComponentName`**: The name of the admin component to check, or **`null`** to aggregate all admins.

Returns

`int`	The length of the password history

getPasswordMaximumLength

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getPasswordMaximumLength (int quality)
```

Return the maximum password length that the device supports for a particular password quality.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always empty and this method always returns 0.

Parameters

`quality`	**`int`**: The quality being interrogated.

Returns

`int`	Returns the maximum length that the user can enter.

getPasswordMinimumLength

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getPasswordMinimumLength (ComponentName (/reference/android/content/ComponentN
```

This method was deprecated in API level 31.

see `setPasswordQuality(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int))
for details.

Retrieve the current minimum password length for a particular admin or all admins that set restrictions on this user and its participating profiles. Restrictions on profiles that have a separate challenge are not taken into account.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName))
in order to retrieve restrictions on the parent profile.

Parameters

`admin`	`ComponentName`: The name of the admin component to check, or `null` to aggregate all admins.

Returns

`int`

getPasswordMinimumLetters

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getPasswordMinimumLetters (ComponentName (/reference/android/content/Component
```

This method was deprecated in API level 31.

see `setPasswordQuality(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) for details.

Retrieve the current number of letters required in the password for a particular admin or all admins that set restrictions on this user and its participating profiles. Restrictions on profiles that have a separate challenge are not taken into account. This is the same value as set by `setPasswordMinimumLetters(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumLetters(android.content.ComponentName,%20int)) and only applies when the password quality is `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX).

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)`

(/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName))
in order to retrieve restrictions on the parent profile.

Parameters

`admin`	**`ComponentName`**: The name of the admin component to check, or **`null`** to aggregate all admins.

Returns

`int`	The minimum number of letters required in the password.

getPasswordMinimumLowerCase

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getPasswordMinimumLowerCase (ComponentName (/reference/android/content/Compone
```

This method was deprecated in API level 31.

see **`setPasswordQuality(android.content.ComponentName, int)`**
(/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int))
for details.

Retrieve the current number of lower case letters required in the password for a particular admin or all admins that set restrictions on this user and its participating profiles. Restrictions on profiles that have a separate challenge are not taken into account. This is the same value as set by `setPasswordMinimumLowerCase(android.content.ComponentName, int)`
(/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumLowerCase(android.content.ComponentName,%20int))
and only applies when the password quality is `PASSWORD_QUALITY_COMPLEX`
(/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX).

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to retrieve restrictions on the parent profile.

Parameters

`admin`	**`ComponentName`**: The name of the admin component to check, or **`null`** to aggregate all admins.

Returns

`int`	The minimum number of lower case letters required in the password.

getPasswordMinimumNonLetter

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getPasswordMinimumNonLetter (ComponentName (/reference/android/content/Compone
```

This method was deprecated in API level 31.

see **`setPasswordQuality(android.content.ComponentName, int)`** (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) for details.

Retrieve the current number of non-letter characters required in the password for a particular admin or all admins that set restrictions on this user and its participating profiles. Restrictions on profiles that have a separate challenge are not taken into account. This is the same value as set by `setPasswordMinimumNonLetter(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumNonLetter(android.content.ComponentName,%20int)) and only applies when the password quality is `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX).

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to retrieve restrictions on the parent profile.

Parameters

`admin`	**`ComponentName`**: The name of the admin component to check, or **`null`** to aggregate all admins.

Returns

`int`	The minimum number of letters required in the password.

getPasswordMinimumNumeric

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getPasswordMinimumNumeric (ComponentName (/reference/android/content/Component
```

This method was deprecated in API level 31.

see `setPasswordQuality(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) for details.

Retrieve the current number of numerical digits required in the password for a particular admin or all admins that set restrictions on this user and its participating profiles. Restrictions on profiles that have a separate challenge are not taken into account. This is the same value as set by `setPasswordMinimumNumeric(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumNumeric(android.content.ComponentName,%20int)) and only applies when the password quality is `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX).

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to retrieve restrictions on the parent profile.

Parameters

`admin`	`ComponentName`: The name of the admin component to check, or `null` to aggregate all admins.

Returns

`int`	The minimum number of numerical digits required in the password.

getPasswordMinimumSymbols

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getPasswordMinimumSymbols (ComponentName (/reference/android/content/Component
```

This method was deprecated in API level 31.

see `setPasswordQuality(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int))
for details.

Retrieve the current number of symbols required in the password for a particular admin or all admins that set restrictions on this user and its participating profiles. Restrictions on profiles that have a separate challenge are not taken into account. This is the same value as set by `setPasswordMinimumSymbols(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumSymbols(android.content.ComponentName,%20int)) and only applies when the password quality is `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX).

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to retrieve restrictions on the parent profile.

Parameters

`admin`	`ComponentName`: The name of the admin component to check, or `null` to aggregate all admins.

Returns

`int`	The minimum number of symbols required in the password.

getPasswordMinimumUpperCase

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getPasswordMinimumUpperCase (ComponentName (/reference/android/content/Compone
```

This method was deprecated in API level 31.

see **`setPasswordQuality(android.content.ComponentName, int)`** (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int))
for details.

Retrieve the current number of upper case letters required in the password for a particular admin or all admins that set restrictions on this user and its participating profiles. Restrictions on profiles that have a separate challenge are not taken into account. This is the same value as set by `setPasswordMinimumUpperCase(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordMinimumUpperCase(android.content.ComponentName,%20int))
and only applies when the password quality is `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX).

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName))
in order to retrieve restrictions on the parent profile.

Parameters

`admin`	`ComponentName`: The name of the admin component to check, or `null` to aggregate all admins.

Returns

`int`	The minimum number of upper case letters required in the password.

getPasswordQuality

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getPasswordQuality (ComponentName (/reference/android/content/ComponentName) a
```

This method was deprecated in API level 31.

see **`setPasswordQuality(android.content.ComponentName, int)`** (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) for details.

Retrieve the current minimum password quality for a particular admin or all admins that set restrictions on this user and its participating profiles. Restrictions on profiles that have a separate challenge are not taken into account.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to retrieve restrictions on the parent profile.

Note: on devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty.

Parameters

`admin`	**`ComponentName`**: The name of the admin component to check, or **`null`** to aggregate all admins.

Returns

`int`

getPendingSystemUpdate

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public SystemUpdateInfo (/reference/android/app/admin/SystemUpdateInfo) getPendingSystemUpd
```

Get information about a pending system update. Can be called by device or profile owners, and starting from Android `ERROR(/android.os.Build.VERSION_CODES#VANILLA_ICE_CREAM)` (/), holders of the permission `Manifest.permission.MANAGE_DEVICE_POLICY_QUERY_SYSTEM_UPDATES` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_QUERY_SYSTEM_UPDATES).

Parameters

`admin`	**`ComponentName`**: Which profile or device owner this request is associated with. This value may be **`null`**.

Returns

SystemUpdateInfo (/reference/android/app/admin/SystemUpdateInfo)	Information about a pending system update or **null** if no update pending.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device, profile owner or holders of Manifest.permission#POLICY_QUERY_SYSTEM_UPDATES (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC.

See also:

`DeviceAdminReceiver.onSystemUpdatePending(Context, Intent, long)` (/reference/android/app/admin/DeviceAdminReceiver#onSystemUpdatePending(android.content.Context,%20android.content.Intent,%20long))

getPermissionGrantState

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getPermissionGrantState (ComponentName (/reference/android/content/ComponentNa
                String (/reference/java/lang/String) packageName,
                String (/reference/java/lang/String) permission)
```

Returns the current grant state of a runtime permission for a specific application. This function can be called by a device owner, profile owner, or by a delegate given the `DELEGATION_PERMISSION_GRANT` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PERMISSION_GRANT) scope via `setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.
`packageName`	`String`: The application to check the grant state for. This value cannot be `null`.
`permission`	`String`: The permission to check for. This value cannot be `null`.

Returns

`int`	the current grant state specified by device policy. If admins have not set a g grant state, the return value is `PERMISSION_GRANT_STATE_DEFAULT` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRAN This does not indicate whether or not the permission is currently granted fo If a grant state was set by the profile or device owner, then the return value `PERMISSION_GRANT_STATE_DENIED` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRAN or `PERMISSION_GRANT_STATE_GRANTED` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRAN , which indicates if the permission is currently denied or granted. Value is `PERMISSION_GRANT_STATE_DEFAULT` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRAN `PERMISSION_GRANT_STATE_GRANTED` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRAN , or `PERMISSION_GRANT_STATE_DENIED` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRAN

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device or profile owner.

See also:

setPermissionGrantState(ComponentName, String, String, int) (/reference/android/app/admin/DevicePolicyManager#setPermissionGrantState(android.content.ComponentName,%20java.lang.String,%20java.lang.String,%20int))

PackageManager.checkPermission(String, String) (/reference/android/content/pm/PackageManager#checkPermission(java.lang.String,%20java.lang.String))

setDelegatedScopes(ComponentName, String, List) (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

DELEGATION_PERMISSION_GRANT (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PERMISSION_GRANT)

getPermissionPolicy

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getPermissionPolicy (ComponentName (/reference/android/content/ComponentName)
```

Returns the current runtime permission policy set by the device or profile owner. The default is PERMISSION_POLICY_PROMPT (/reference/android/app/admin/DevicePolicyManager#PERMISSION_POLICY_PROMPT).

Parameters

admin	**ComponentName**: Which profile or device owner this request is associated with.

Returns

`int`	the current policy for future permission requests.

getPermittedAccessibilityServices

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<String (/reference/java/lang/String)> getPermittedAccessibi
```

Returns the list of permitted accessibility services set by this device or profile owner.

An empty list means no accessibility services except system services are allowed. `null` means all accessibility services are allowed.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

`List` (/reference/java/util/List) `<String` (/reference/java/lang/String)`>`	List of accessiblity service package names.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner.

getPermittedCrossProfileNotificationListeners

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<String (/reference/java/lang/String)> getPermittedCrossProf
```

Returns the list of packages installed on the primary user that allowed to use a `NotificationListenerService` (/reference/android/service/notification/NotificationListenerService) to receive notifications from this managed profile, as set by the profile owner.

An empty list means no notification listener services except system ones are allowed. A `null` return value indicates that all notification listeners are allowed.

Parameters

`admin`	`ComponentName`: This value cannot be `null`.

Returns

`List` (/reference/java/util/List)
<`String`
(/reference/java/lang/String)>

getPermittedInputMethods

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<String (/reference/java/lang/String)> getPermittedInputMeth
```

Returns the list of permitted input methods set by this device or profile owner.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance, returned by `getParentProfileInstance(android.content.ComponentName)`

(/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName))

, where the caller must be a profile owner of an organization-owned managed profile. If called on the parent instance, then the returned list of permitted input methods are those which are applied on the personal profile.

An empty list means no input methods except system input methods are allowed. Null means all input methods are allowed.

Parameters

`admin`	**`ComponentName`**: Which **`DeviceAdminReceiver`** (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin This value may be **`null`**.

Returns

`List` (/reference/java/util/List) <**`String`** (/reference/java/lang/String)>	List of input method package names. This value may be **`null`**.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if **`admin`** is not a device, profile owner or if called on the parent profile and the **`admin`** is not a profile owner of an organization-owned managed profile.

getPersonalAppsSuspendedReasons

Added in **`API level 33`** (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getPersonalAppsSuspendedReasons (ComponentName (/reference/android/content/Com
```

Called by profile owner of an organization-owned managed profile to check whether personal apps are suspended.

Parameters

`admin`	`ComponentName`: This value cannot be `null`.

Returns

`int`	a bitmask of reasons for personal apps suspension or PERSONAL_APPS_NO (/reference/android/app/admin/DevicePolicyManager#PERSONAL_APPS_N not suspended. Value is either `0` or a combination of PERSONAL_APPS_NOT (/reference/android/app/admin/DevicePolicyManager#PERSONAL_APPS_N APPS_SUSPENDED_EXPLICITLY (/reference/android/app/admin/DevicePolicyManager#PERSONAL_APPS_S PERSONAL_APPS_SUSPENDED_PROFILE_TIMEOUT (/reference/android/app/admin/DevicePolicyManager#PERSONAL_APPS_S

See also:

setPersonalAppsSuspended(ComponentName, boolean) (/reference/android/app/admin/DevicePolicyManager#setPersonalAppsSuspended(android.content.ComponentName,%20boolean))

getPreferentialNetworkServiceConfigs

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<PreferentialNetworkServiceConfig (/reference/android/app/
```

Get preferential network configuration

Returns

List (/reference/java/util/List)<PreferentialNetworkServiceConfig (/reference/android/app/admin/PreferentialNetworkServiceConfig)>	preferential network configuration. This value cannot be **null**.

Throws

SecurityException (/reference/java/lang/SecurityException)	if the caller is not the profile owner or device owner.

See also:

`PreferentialNetworkServiceConfig` (/reference/android/app/admin/PreferentialNetworkServiceConfig)

getRequiredPasswordComplexity

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getRequiredPasswordComplexity ()
```

Gets the password complexity requirement set by `setRequiredPasswordComplexity(int)` (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswordComplexity(int)), for the current user.

The difference between this method and `getPasswordComplexity()` (/reference/android/app/admin/DevicePolicyManager#getPasswordComplexity()) is that this method simply returns the value set by `setRequiredPasswordComplexity(int)` (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswordComplexity(int)) while `getPasswordComplexity()` (/reference/android/app/admin/DevicePolicyManager#getPasswordComplexity()) returns the complexity of the actual password.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to get restrictions on the parent profile.

Returns

`int`	Value is PASSWORD_COMPLEXITY_NONE (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPL PASSWORD_COMPLEXITY_LOW (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPL PASSWORD_COMPLEXITY_MEDIUM (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPL, or PASSWORD_COMPLEXITY_HIGH (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPL

Throws

SecurityException (/reference/java/lang/SecurityException)	if the calling application is not a device owner or a profile owner.

getRequiredStrongAuthTimeout

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public long getRequiredStrongAuthTimeout (ComponentName (/reference/android/content/Compo
```

Determine for how long the user will be able to use secondary, non strong auth for authentication, since last strong method authentication (password, pin or pattern) was used. After the returned timeout the user is required to use strong authentication method.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by

`getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to retrieve restrictions on the parent profile.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, 0 is returned to indicate that no timeout is configured.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Parameters

`admin`	**`ComponentName`**: The name of the admin component to check, or **`null`** to aggregate across all participating admins.

Returns

`long`	The timeout in milliseconds or 0 if not configured for the provided admin.

getResources

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public DevicePolicyResourcesManager (/reference/android/app/admin/DevicePolicyResourcesManag
```

Returns a `DevicePolicyResourcesManager` (/reference/android/app/admin/DevicePolicyResourcesManager) containing the required APIs to set, reset, and get device policy related resources.

Returns

DevicePolicyResourcesManager (/reference/android/app/admin/DevicePolicyResourcesManager)	This value cannot be **null**.

getScreenCaptureDisabled

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean getScreenCaptureDisabled (ComponentName (/reference/android/content/Compor
```

Determine whether or not screen capture has been disabled by the calling admin, if specified, or all admins.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance, returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) , where the caller must be the profile owner of an organization-owned managed profile (the calling admin must be specified).

Parameters

`admin`	**ComponentName**: The name of the admin component to check, or **null** to check whether any admins have disabled screen capture.

Returns

`boolean`

getSecondaryUsers

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<UserHandle (/reference/android/os/UserHandle)> getSecondary
```

Called by a device owner to list all secondary users on the device. Managed profiles are not considered as secondary users.

Used for various user management APIs, including `switchUser(ComponentName, UserHandle)` (/reference/android/app/admin/DevicePolicyManager#switchUser(android.content.ComponentName,%20android.os.UserHandle)), `removeUser(ComponentName, UserHandle)` (/reference/android/app/admin/DevicePolicyManager#removeUser(android.content.ComponentName,%20android.os.UserHandle)) and `stopUser(ComponentName, UserHandle)` (/reference/android/app/admin/DevicePolicyManager#stopUser(android.content.ComponentName,%20android.os.UserHandle)).

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

`List` (/reference/java/util/List)<`UserHandle` (/reference/android/os/UserHandle)>	list of other `UserHandle` (/reference/android/os/UserHandle)s on the device.

Throws

SecurityException (/reference/java/lang/SecurityException)	if admin is not a device owner.

See also:

removeUser(ComponentName, UserHandle) (/reference/android/app/admin/DevicePolicyManager#removeUser(android.content.ComponentName,%20android.os.UserHandle))

switchUser(ComponentName, UserHandle) (/reference/android/app/admin/DevicePolicyManager#switchUser(android.content.ComponentName,%20android.os.UserHandle))

startUserInBackground(ComponentName, UserHandle) (/reference/android/app/admin/DevicePolicyManager#startUserInBackground(android.content.ComponentName,%20android.os.UserHandle))

stopUser(ComponentName, UserHandle) (/reference/android/app/admin/DevicePolicyManager#stopUser(android.content.ComponentName,%20android.os.UserHandle))

getShortSupportMessage added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public CharSequence (/reference/java/lang/CharSequence) getShortSupportMessage (Component
```

Called by a device admin or holder of the permission Manifest.permission.MANAGE_DEVICE_POLICY_SUPPORT_MESSAGE (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_SUPPORT_MESSAGE) to get the short support message.

Parameters

admin	ComponentName: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be null.

Returns

CharSequence (/reference/java/lang/CharSequence)	The message set by setShortSupportMessage(android.content (/reference/android/app/admin/DevicePolicyManager#setShortSuppor or null if no message has been set.

Throws

SecurityException (/reference/java/lang/SecurityException)	if admin is not an active administrator and not a holder of the perm permission.MANAGE_DEVICE_POLICY_SUPPORT_MESSAGE (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC ..

getStartUserSessionMessage

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public CharSequence (/reference/java/lang/CharSequence) getStartUserSessionMessage (Compo
```

Returns the user session start message.

Parameters

admin	ComponentName: which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be null.

Returns

CharSequence (/reference/java/lang/CharSequence)

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device owner.

getStorageEncryption

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean getStorageEncryption (ComponentName (/reference/android/content/ComponentN
```

This method was deprecated in API level 30.

This method only returns the value set by **setStorageEncryption(ComponentName, boolean)** (/reference/android/app/admin/DevicePolicyManager#setStorageEncryption(android.content.ComponentName,%20boolean)) . It does not actually reflect the storage encryption status. Use **getStorageEncryptionStatus()** (/reference/android/app/admin/DevicePolicyManager#getStorageEncryptionStatus()) for that. Called by an application that is administering the device to determine the requested setting for secure storage.

Parameters

admin	ComponentName: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. If null, this will return the requested encryption setting as an aggregate of all active administrators.

Returns

`boolean`	true if the admin(s) are requesting encryption, false if not.

getStorageEncryptionStatus added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getStorageEncryptionStatus ()
```

Called by an application that is administering the device to determine the current encryption status of the device.

Depending on the returned status code, the caller may proceed in different ways. If the result is `ENCRYPTION_STATUS_UNSUPPORTED` (/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATUS_UNSUPPORTED), the storage system does not support encryption. If the result is `ENCRYPTION_STATUS_INACTIVE` (/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATUS_INACTIVE), use `ACTION_START_ENCRYPTION` (/reference/android/app/admin/DevicePolicyManager#ACTION_START_ENCRYPTION) to begin the process of encrypting or decrypting the storage. If the result is `ENCRYPTION_STATUS_ACTIVE_DEFAULT_KEY` (/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATUS_ACTIVE_DEFAULT_KEY), the storage system has enabled encryption but no password is set so further action may be required. If the result is `ENCRYPTION_STATUS_ACTIVATING` (/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATUS_ACTIVATING), `ENCRYPTION_STATUS_ACTIVE` (/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATUS_ACTIVE) or `ENCRYPTION_STATUS_ACTIVE_PER_USER` (/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATUS_ACTIVE_PER_USER), no further action is required.

Returns

`int`	current status of encryption. The value will be one of `ENCRYPTION_STATUS` (/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATU `ENCRYPTION_STATUS_INACTIVE` (/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATU `ENCRYPTION_STATUS_ACTIVATING`

(/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATU
ENCRYPTION_STATUS_ACTIVE_DEFAULT_KEY
(/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATU
, ENCRYPTION_STATUS_ACTIVE
(/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATU
ENCRYPTION_STATUS_ACTIVE_PER_USER
(/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATU

Throws

SecurityException (/reference/java/lang/SecurityException)	if called on a parent instance.

getSystemUpdatePolicy

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public SystemUpdatePolicy (/reference/android/app/admin/SystemUpdatePolicy) getSystemUpdateP
```

Retrieve a local system update policy set previously by `setSystemUpdatePolicy(ComponentName, SystemUpdatePolicy)` (/reference/android/app/admin/DevicePolicyManager#setSystemUpdatePolicy(android.content.ComponentName,%20android.app.admin.SystemUpdatePolicy)).

Returns

SystemUpdatePolicy (/reference/android/app/admin/SystemUpdatePolicy)	The current policy object, or **null** if no policy is set.

getTransferOwnershipBundle

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public PersistableBundle (/reference/android/os/PersistableBundle) getTransferOwnershipBundl
```

Returns the data passed from the current administrator to the new administrator during an ownership transfer. This is the same `bundle` passed in `transferOwnership(android.content.ComponentName, android.content.ComponentName, android.os.PersistableBundle)` (/reference/android/app/admin/DevicePolicyManager#transferOwnership(android.content.ComponentName,%20android.content.ComponentName,%20android.os.PersistableBundle)). The bundle is persisted until the profile owner or device owner is removed.

This is the same `bundle` received in the `DeviceAdminReceiver#onTransferOwnershipComplete(Context, PersistableBundle)` (/reference/android/app/admin/DeviceAdminReceiver#onTransferOwnershipComplete(android.content.Context,%20android.os.PersistableBundle)). Use this method to retrieve it after the transfer as long as the new administrator is the active device or profile owner.

Returns `null` if no ownership transfer was started for the calling user.

Returns

`PersistableBundle` (/reference/android/os/PersistableBundle)

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not a device or profile owner.

See also:

`transferOwnership(ComponentName, ComponentName, PersistableBundle)` (/reference/android/app/admin/DevicePolicyManager#transferOwnership(android.content.ComponentName,%20android.content.ComponentName,%20android.os.PersistableBundle))

`DeviceAdminReceiver.onTransferOwnershipComplete(Context, PersistableBundle)` (/reference/android/app/admin/DeviceAdminReceiver#onTransferOwnershipComplete(android.content.Context,%20android.os.PersistableBundle))

getTrustAgentConfiguration

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<PersistableBundle (/reference/android/os/PersistableBundle)>
            ComponentName (/reference/android/content/ComponentName) agent)
```

Gets configuration for the given trust agent based on aggregating all calls to `setTrustAgentConfiguration(android.content.ComponentName, android.content.ComponentName, android.os.PersistableBundle)` (/reference/android/app/admin/DevicePolicyManager#setTrustAgentConfiguration(android.content.ComponentName,%20android.content.ComponentName,%20android.os.PersistableBundle)) for all device admins.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to retrieve the configuration set on the parent profile.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, null is always returned.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Parameters

`admin`	**ComponentName**: Which `DeviceAdminReceiver` (/reference/android/app/ all admins that declare `KEYGUARD_DISABLE_TRUST_AGENTS` (/reference/a `DISABLE_TRUST_AGENTS` (/reference/android/app/admin/DevicePolicyMar `content.ComponentName, android.content.ComponentName, and`

(/reference/android/app/admin/DevicePolicyManager#setTrustAgentConfig
for this or calls it with a null configuration, null is returned.

`agent`	`ComponentName`: Which component to get enabled features for. This value

Returns

`List` (/reference/java/util/List)<`PersistableBundle` (/reference/android/os/PersistableBundle)>	configuration for the given trust agent.

getUserControlDisabledPackages

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<String (/reference/java/lang/String)> getUserControlDisable
```

Returns the list of packages over which user control is disabled by a device or profile owner or holders of the permission `Manifest.permission.MANAGE_DEVICE_POLICY_APPS_CONTROL` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_APPS_CONTROL).

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), the returned policy will be the current resolved policy rather than the policy set by the calling admin.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.

Returns

List (/reference/java/util/List)<String (/reference/java/lang/String)>	This value cannot be null.

Throws

SecurityException (/reference/java/lang/SecurityException)	if admin is not a device or profile owner or holder of the permission permission.MANAGE_DEVICE_POLICY_APPS_CONTROL (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC.

getUserRestrictions

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public Bundle (/reference/android/os/Bundle) getUserRestrictions (ComponentName (/reference/a
```

Called by an admin to get user restrictions set by themselves with addUserRestriction(android.content.ComponentName, java.lang.String) (/reference/android/app/admin/DevicePolicyManager#addUserRestriction(android.content.ComponentName,%20java.lang.String)).

The target user may have more restrictions set by the system or other admin. To get all the user restrictions currently set, use UserManager#getUserRestrictions() (/reference/android/os/UserManager#getUserRestrictions()).

The profile owner of an organization-owned managed profile may invoke this method on the DevicePolicyManager (/reference/android/app/admin/DevicePolicyManager) instance it obtained from getParentProfileInstance(android.content.ComponentName) (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)), for retrieving device-wide restrictions it previously set with addUserRestriction(android.content.ComponentName, java.lang.String)

(/reference/android/app/admin/DevicePolicyManager#addUserRestriction(android.content.ComponentName,%20java.lang.String))

.

For callers targeting Android `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE) or above, this API will return the local restrictions set on the calling user, or on the parent profile if called from the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance obtained from `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) . To get global restrictions set by admin, call `getUserRestrictionsGlobally()` (/reference/android/app/admin/DevicePolicyManager#getUserRestrictionsGlobally()) instead.

Note that this is different that the returned restrictions for callers targeting pre Android `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), were this API returns all local/global restrictions set by the admin on the calling user using `addUserRestriction(android.content.ComponentName, java.lang.String)` (/reference/android/app/admin/DevicePolicyManager#addUserRestriction(android.content.ComponentName,%20java.lang.String)) or the parent user if called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance it obtained from `getParentProfileInstance(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) .

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

Bundle (/reference/android/os/Bundle)	a **Bundle** (/reference/android/os/Bundle) whose keys are the user restrictions, and the values a **boolean** indicating whether the restriction is set. This value cannot be **null**.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device or profile owner.

getUserRestrictionsGlobally

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public Bundle (/reference/android/os/Bundle) getUserRestrictionsGlobally ()
```

Called by a profile or device owner to get global user restrictions set with `addUserRestrictionGlobally(java.lang.String)` (/reference/android/app/admin/DevicePolicyManager#addUserRestrictionGlobally(java.lang.String)).

To get all the user restrictions currently set for a certain user, use `UserManager#getUserRestrictions()` (/reference/android/os/UserManager#getUserRestrictions()).

Returns

Bundle (/reference/android/os/Bundle)	a **Bundle** (/reference/android/os/Bundle) whose keys are the user restrictions, and the values a **boolean** indicating whether the restriction is set. This value cannot be **null**.

Throws

SecurityException	if **admin** is not a device or profile owner.

(/reference/java/lang/SecurityException)

IllegalStateException (/reference/java/lang/IllegalStateException)	if caller is not targeting Android Build.VERSION_CODES.UPSIDE_DOWN_CAKE (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_C or above.

getWifiMacAddress

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public String (/reference/java/lang/String) getWifiMacAddress (ComponentName (/reference/androi
```

Called by a device owner or profile owner on organization-owned device to get the MAC address of the Wi-Fi device. NOTE: The MAC address returned here should only be used for inventory management and is not likely to be the MAC address used by the device to connect to Wi-Fi networks: MAC addresses used for scanning and connecting to Wi-Fi networks are randomized by default. To get the randomized MAC address used, call `WifiConfiguration.getRandomizedMacAddress()` (/reference/android/net/wifi/WifiConfiguration#getRandomizedMacAddress()).

Parameters

`admin`	`ComponentName`: Which admin this request is associated with. Null if the caller is not a device admin This value may be `null`.

Returns

String (/reference/java/lang/String)	the MAC address of the Wi-Fi device, or null when the information is not available. (For example, Wi-Fi hasn't been enabled, or the device doesn't support Wi-Fi.) The address will be in the `XX:XX:XX:XX:XX:XX` format.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not permitted to get wifi mac addresses

getWifiSsidPolicy

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public WifiSsidPolicy (/reference/android/app/admin/WifiSsidPolicy) getWifiSsidPolicy ()
```

Returns the current Wi-Fi SSID policy. If the policy has not been set, it will return NULL.

Returns

WifiSsidPolicy (/reference/android/app/admin/WifiSsidPolicy)	This value may be **null**.

Throws

SecurityException (/reference/java/lang/SecurityException)	if the caller is not a device owner or a profile owner on an organization-owned managed profile.

See also:

setWifiSsidPolicy(WifiSsidPolicy) (/reference/android/app/admin/DevicePolicyManager#setWifiSsidPolicy(android.app.admin.WifiSsidPolicy))

grantKeyPairToApp

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean grantKeyPairToApp (ComponentName (/reference/android/content/ComponentName
                String (/reference/java/lang/String) alias,
```

`String` (/reference/java/lang/String) `packageName`)

Called by a device or profile owner, or delegated certificate chooser (an app that has been delegated the `DELEGATION_CERT_SELECTION` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_SELECTION) privilege), to grant an application access to an already-installed (or generated) KeyChain key. This is useful (in combination with `installKeyPair(ComponentName, PrivateKey, Certificate, String)` (/reference/android/app/admin/DevicePolicyManager#installKeyPair(android.content.ComponentName,%20java.security.PrivateKey,%20java.security.cert.Certificate,%20java.lang.String)) or `generateKeyPair(ComponentName, String, KeyGenParameterSpec, int)` (/reference/android/app/admin/DevicePolicyManager#generateKeyPair(android.content.ComponentName,%20java.lang.String,%20android.security.keystore.KeyGenParameterSpec,%20int))) to let an application call `KeyChain.getPrivateKey(Context, String)` (/reference/android/security/KeyChain#getPrivateKey(android.content.Context,%20java.lang.String)) without having to call `KeyChain.choosePrivateKeyAlias(Activity, KeyChainAliasCallback, String, Principal, Uri, String)` (/reference/android/security/KeyChain#choosePrivateKeyAlias(android.app.Activity,%20android.security.KeyChainAliasCallback,%20java.lang.String[],%20java.security.Principal[],%20android.net.Uri,%20java.lang.String)) first. The grantee app will receive the `KeyChain.ACTION_KEY_ACCESS_CHANGED` (/reference/android/security/KeyChain#ACTION_KEY_ACCESS_CHANGED) broadcast when access to a key is granted. Starting from `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE) throws an `IllegalArgumentException` (/reference/java/lang/IllegalArgumentException) if `alias` doesn't correspond to an existing key.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if calling from a delegated certificate chooser.
`alias`	`String`: The alias of the key to grant access to. This value cannot be `null`.
`packageName`	`String`: The name of the (already installed) package to grant access to. This value cannot be `null`.

Returns

`boolean`	`true` if the grant was set successfully, `false` otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not a device owner, a profile owner or delegated certificate chooser.
`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if `packageName` or `alias` are empty, or if `packageName` is not a name of an installed package.

See also:

`revokeKeyPairFromApp(ComponentName, String, String)` (/reference/android/app/admin/DevicePolicyManager#revokeKeyPairFromApp(android.content.ComponentName,%20java.lang.String,%20java.lang.String))

grantKeyPairToWifiAuth

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean grantKeyPairToWifiAuth (String (/reference/java/lang/String) alias)
```

Called by a device or profile owner, or delegated certificate chooser (an app that has been delegated the `DELEGATION_CERT_SELECTION` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_SELECTION) privilege), to allow using a KeyChain key pair for authentication to Wifi networks. The key can then be used in configurations passed to `WifiManager.addNetwork(WifiConfiguration)` (/reference/android/net/wifi/WifiManager#addNetwork(android.net.wifi.WifiConfiguration)). Starting from `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE) throws an

`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException) if `alias` doesn't correspond to an existing key.

Parameters

`alias`	`String`: The alias of the key pair. This value cannot be `null`.

Returns

`boolean`	`true` if the operation was set successfully, `false` otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not a device owner, a profile owner or delegated certificate chooser.

See also:

`revokeKeyPairFromWifiAuth(String)` (/reference/android/app/admin/DevicePolicyManager#revokeKeyPairFromWifiAuth(java.lang.String))

hasCaCertInstalled

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean hasCaCertInstalled (ComponentName (/reference/android/content/ComponentNam
                byte[] certBuffer)
```

Returns whether this certificate is installed as a trusted CA.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if calling from a delegated certificate installer.
`certBuffer`	`byte`: encoded form of the certificate to look up.

Returns

`boolean`

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not `null` and not a device or profile owner.

hasGrantedPolicy

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean hasGrantedPolicy (ComponentName (/reference/android/content/ComponentName)
                int usesPolicy)
```

Returns true if an administrator has been granted a particular device policy. This can be used to check whether the administrator was activated under an earlier set of policies, but requires additional policies after an upgrade.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Must be an active administrator, or an exception will be thrown. This value cannot be `null`.
`usesPolicy`	`int`: Which uses-policy to check, as defined in `DeviceAdminInfo` (/reference/android/app/admin/DeviceAdminInfo).

Returns

`boolean`

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not an active administrator.

hasKeyPair

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean hasKeyPair (String (/reference/java/lang/String) alias)
```

This API can be called by the following to query whether a certificate and private key are installed under a given alias:

- Device owner
- Profile owner
- Delegated certificate installer
- Credential management app

- An app that holds the `Manifest.permission.MANAGE_DEVICE_POLICY_CERTIFICATES` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_CERTIFICATES) permission

If called by the credential management app, the alias must exist in the credential management app's `AppUriAuthenticationPolicy` (/reference/android/security/AppUriAuthenticationPolicy).

Parameters

`alias`	`String`: The alias under which the key pair is installed. This value cannot be `null`.

Returns

`boolean`	`true` if a key pair with this alias exists, `false` otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not a device or profile owner, a delegated certificate i credential management app and does not have the `Manifest.per MANAGE_DEVICE_POLICY_CERTIFICATES` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC permission.

See also:

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`DELEGATION_CERT_INSTALL` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_INSTALL)

hasLockdownAdminConfiguredNetworks

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean hasLockdownAdminConfiguredNetworks (ComponentName (/reference/android/co
```

Called by a device owner or a profile owner of an organization-owned managed profile to determine whether the user is prevented from modifying networks configured by the admin.

Parameters

`admin`	`ComponentName`: admin Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value may be `null`.

Returns

`boolean`

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if caller is not a device owner or a profile owner of an organization-owned managed profile.

installCaCert

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean installCaCert (ComponentName (/reference/android/content/ComponentName) ad
                byte[] certBuffer)
```

Installs the given certificate as a user CA.

Inserted user CAs aren't automatically trusted by apps in Android 7.0 (API level 24) and higher. App developers can change the default behavior for an app by adding a Security Configuration File (/training/articles/security-config) to the app manifest file. The caller must be a profile or device owner on that user, or a delegate package given the `DELEGATION_CERT_INSTALL` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_INSTALL) scope via `setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>)) ; otherwise a security exception will be thrown.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if calling from a delegated certificate installer.
`certBuffer`	`byte`: encoded form of the certificate to install.

Returns

`boolean`	false if the certBuffer cannot be parsed or installation is interrupted, true otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not `null` and not a device or profile owner.

See also:

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`DELEGATION_CERT_INSTALL` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_INSTALL)

installExistingPackage

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean installExistingPackage (ComponentName (/reference/android/content/Componer
                String (/reference/java/lang/String) packageName)
```

Install an existing package that has been installed in another user, or has been kept after removal via `setKeepUninstalledPackages(ComponentName, List)` (/reference/android/app/admin/DevicePolicyManager#setKeepUninstalledPackages(android.content.ComponentName,%20java.util.List<java.lang.String>)). This function can be called by a device owner, profile owner or a delegate given the `DELEGATION_INSTALL_EXISTING_PACKAGE` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_INSTALL_EXISTING_PACKAGE) scope via `setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>)). When called in a secondary user or managed profile, the user/profile must be affiliated with the device. See `isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser()).

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`packageName`	`String`: The package to be installed in the calling profile.

Returns

`boolean`	`true` if the app is installed; `false` otherwise.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not the device owner, or the profile owner of an affiliated user or profile.

See also:

`setKeepUninstalledPackages(ComponentName, List)` (/reference/android/app/admin/DevicePolicyManager#setKeepUninstalledPackages(android.content.ComponentName,%20java.util.List<java.lang.String>))

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser())

`DELEGATION_PACKAGE_ACCESS` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PACKAGE_ACCESS)

installKeyPair

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean installKeyPair (ComponentName (/reference/android/content/ComponentName) a
                PrivateKey (/reference/java/security/PrivateKey) privKey,
                Certificate[] (/reference/java/security/cert/Certificate) certs,
                String (/reference/java/lang/String) alias,
                int flags)
```

This API can be called by the following to install a certificate chain and corresponding private key for the leaf certificate:

- Device owner
- Profile owner
- Delegated certificate installer
- Credential management app
- An app that holds the `Manifest.permission.MANAGE_DEVICE_POLICY_CERTIFICATES` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_CERTIFICATES) permission

All apps within the profile will be able to access the certificate chain and use the private key, given direct user approval (if the user is allowed to select the private key).

From Android `Build.VERSION_CODES.S` (/reference/android/os/Build.VERSION_CODES#S), the credential management app can call this API. If called by the credential management app:

- The componentName must be `null`r
- The alias must exist in the credential management app's `AppUriAuthenticationPolicy` (/reference/android/security/AppUriAuthenticationPolicy)
- The key pair must not be user selectable

Note, there can only be a credential management app on an unmanaged device.

The caller of this API may grant itself access to the certificate and private key immediately, without user approval. It is a best practice not to request this unless strictly necessary since it opens up additional security vulnerabilities.

Include `INSTALLKEY_SET_USER_SELECTABLE` (/reference/android/app/admin/DevicePolicyManager#INSTALLKEY_SET_USER_SELECTABLE) in the `flags` argument to allow the user to select the key from a dialog.

Note: If the provided `alias` is of an existing alias, all former grants that apps have been given to access the key and certificates associated with this alias will be revoked.

Parameters

`admin`	**`ComponentName`**: Which `DeviceAdminReceiver` (/reference/android/app this request is associated with, or **`null`** if the caller is not a device admin.

`privKey`	`PrivateKey`: The private key to install. This value cannot be `null`.
`certs`	`Certificate`: The certificate chain to install. The chain should start with t the chain of trust in order. This will be returned by `KeyChain.getCertifi String)` (/reference/android/security/KeyChain#getCertificateChain(android.conte . This value cannot be `null`.
`alias`	`String`: The private key alias under which to install the certificate. If a cert exists, it will be overwritten. This value cannot be `null`.
`flags`	`int`: Flags to request that the calling app be granted access to the credent selectable. See `INSTALLKEY_SET_USER_SELECTABLE` (/reference/android/app/admin/DevicePolicyManager#INSTALLKEY_SET_US `INSTALLKEY_REQUEST_CREDENTIALS_ACCESS` (/reference/android/app/admin/DevicePolicyManager#INSTALLKEY_REQUE

Returns

`boolean`	`true` if the keys were installed, `false` otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not `null` and not a device or profile owner, or `admin` is n calling application is not a delegated certificate installer, credential app and does not have the `Manifest.permission.MANAGE_DEV CERTIFICATES` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC permission.

See also:

`KeyChain.getCertificateChain(Context, String)` (/reference/android/security/KeyChain#getCertificateChain(android.content.Context,%20java.lang.String))

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`DELEGATION_CERT_INSTALL` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_INSTALL)

installKeyPair

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean installKeyPair (ComponentName (/reference/android/content/ComponentName) a
                PrivateKey (/reference/java/security/PrivateKey) privKey,
                Certificate[] (/reference/java/security/cert/Certificate) certs,
                String (/reference/java/lang/String) alias,
                boolean requestAccess)
```

This API can be called by the following to install a certificate chain and corresponding private key for the leaf certificate:

- Device owner
- Profile owner
- Delegated certificate installer
- Credential management app
- An app that holds the `Manifest.permission.MANAGE_DEVICE_POLICY_CERTIFICATES` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_CERTIFICATES) permission

All apps within the profile will be able to access the certificate chain and use the private key, given direct user approval.

From Android `Build.VERSION_CODES.S` (/reference/android/os/Build.VERSION_CODES#S), the credential management app can call this API. However, this API sets the key pair as user selectable by default, which is not permitted when called by the credential management app. Instead, `installKeyPair(android.content.ComponentName, java.security.PrivateKey, java.security.cert.Certificate[], java.lang.String, int)`

(/reference/android/app/admin/DevicePolicyManager#installKeyPair(android.content.ComponentName,%20java.security.PrivateKey,%20java.security.cert.Certificate[],%20java.lang.String,%20int)) should be called with `INSTALLKEY_SET_USER_SELECTABLE` (/reference/android/app/admin/DevicePolicyManager#INSTALLKEY_SET_USER_SELECTABLE) not set as a flag. Note, there can only be a credential management app on an unmanaged device.

The caller of this API may grant itself access to the certificate and private key immediately, without user approval. It is a best practice not to request this unless strictly necessary since it opens up additional security vulnerabilities.

Note: If the provided `alias` is of an existing alias, all former grants that apps have been given to access the key and certificates associated with this alias will be revoked.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app this request is associated with, or `null` if the caller is not a device admin.
`privKey`	`PrivateKey`: The private key to install. This value cannot be `null`.
`certs`	`Certificate`: The certificate chain to install. The chain should start with t the chain of trust in order. This will be returned by `KeyChain.getCertifi String)` (/reference/android/security/KeyChain#getCertificateChain(android.conte . This value cannot be `null`.
`alias`	`String`: The private key alias under which to install the certificate. If a cert exists, it will be overwritten. This value cannot be `null`.
`requestAccess`	`boolean`: `true` to request that the calling app be granted access to the cr Otherwise, access to the credentials will be gated by user approval.

Returns

`boolean`	`true` if the keys were installed, `false` otherwise.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not `null` and not a device or profile owner, or `admin` is r calling application does not have the Manifest.permission.MAN POLICY_CERTIFICATES (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC permission.

See also:

KeyChain.getCertificateChain(Context, String) (/reference/android/security/KeyChain#getCertificateChain(android.content.Context,%20java.lang.String))

setDelegatedScopes(ComponentName, String, List) (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

DELEGATION_CERT_INSTALL (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_INSTALL)

installKeyPair

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean installKeyPair (ComponentName (/reference/android/content/ComponentName) a
                PrivateKey (/reference/java/security/PrivateKey) privKey,
                Certificate (/reference/java/security/cert/Certificate) cert,
                String (/reference/java/lang/String) alias)
```

This API can be called by the following to install a certificate and corresponding private key:

- Device owner

- Profile owner
- Delegated certificate installer
- Credential management app
- An app that holds the `Manifest.permission.MANAGE_DEVICE_POLICY_CERTIFICATES` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_CERTIFICATES) permission

All apps within the profile will be able to access the certificate and use the private key, given direct user approval.

From Android `Build.VERSION_CODES.S` (/reference/android/os/Build.VERSION_CODES#S), the credential management app can call this API. However, this API sets the key pair as user selectable by default, which is not permitted when called by the credential management app. Instead, `installKeyPair(android.content.ComponentName, java.security.PrivateKey, java.security.cert.Certificate[], java.lang.String, int)` (/reference/android/app/admin/DevicePolicyManager#installKeyPair(android.content.ComponentName,%20java.security.PrivateKey,%20java.security.cert.Certificate[],%20java.lang.String,%20int)) should be called with `INSTALLKEY_SET_USER_SELECTABLE` (/reference/android/app/admin/DevicePolicyManager#INSTALLKEY_SET_USER_SELECTABLE) not set as a flag.

Access to the installed credentials will not be granted to the caller of this API without direct user approval. This is for security - should a certificate installer become compromised, certificates it had already installed will be protected.

If the installer must have access to the credentials, call `installKeyPair(android.content.ComponentName, java.security.PrivateKey, java.security.cert.Certificate[], java.lang.String, boolean)` (/reference/android/app/admin/DevicePolicyManager#installKeyPair(android.content.ComponentName,%20java.security.PrivateKey,%20java.security.cert.Certificate[],%20java.lang.String,%20boolean)) instead.

Note: If the provided `alias` is of an existing alias, all former grants that apps have been given to access the key and certificates associated with this alias will be revoked.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if the caller is not a device admin.

`privKey`	`PrivateKey`: The private key to install. This value cannot be `null`.
`cert`	`Certificate`: The certificate to install. This value cannot be `null`.
`alias`	`String`: The private key alias under which to install the certificate. If a certificate with that alias already exists, it will be overwritten. This value cannot be `null`.

Returns

`boolean`	`true` if the keys were installed, `false` otherwise.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not `null` and not a device or profile owner, or `admin` is r calling application does not have the Manifest.permission.MAN POLICY_CERTIFICATES (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC permission.

See also:

setDelegatedScopes(ComponentName, String, List) (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

DELEGATION_CERT_INSTALL (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_INSTALL)

installSystemUpdate

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void installSystemUpdate (ComponentName (/reference/android/content/ComponentName)
                Uri (/reference/android/net/Uri) updateFilePath,
                Executor (/reference/java/util/concurrent/Executor) executor,
                DevicePolicyManager.InstallSystemUpdateCallback (/reference/android/app/
```

Called by device owner or profile owner of an organization-owned managed profile to install a system update from the given file. The device will be rebooted in order to finish installing the update. Note that if the device is rebooted, this doesn't necessarily mean that the update has been applied successfully. The caller should additionally check the system version with `Build.FINGERPRINT` (/reference/android/os/Build#FINGERPRINT) or `Build.VERSION` (/reference/android/os/Build.VERSION). If an error occurs during processing the OTA before the reboot, the caller will be notified by `InstallSystemUpdateCallback` (/reference/android/app/admin/DevicePolicyManager.InstallSystemUpdateCallback). If device does not have sufficient battery level, the installation will fail with error `DevicePolicyManager.InstallSystemUpdateCallback.UPDATE_ERROR_BATTERY_LOW` (/reference/android/app/admin/DevicePolicyManager.InstallSystemUpdateCallback#UPDATE_ERROR_BATTERY_LOW)

.

Parameters

`admin`	`ComponentName`: The `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) that this request is associated with. Null if the caller is not a device admin This value may be `null`.
`updateFilePath`	`Uri`: A Uri of the file that contains the update. The file should be readable by the calling app. This value cannot be `null`.
`executor`	`Executor`: The executor through which the callback should be invoked. This value cannot be `null`. Callback and listener events are dispatched through this `Executor` (/reference/java/util/concurrent/Executor), providing an easy way to control which thread is used. To dispatch events through the main thread of your application, you can use `Context.getMainExecutor()`

	(/reference/android/content/Context#getMainExecutor()). Otherwise, provide an `Executor` (/reference/java/util/concurrent/Executor) that dispatches to an appropriate thread.
`callback`	`DevicePolicyManager.InstallSystemUpdateCallback`: A callback object that will inform the caller when installing an update fails. This value cannot be `null`.

isActivePasswordSufficient

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isActivePasswordSufficient ()
```

Determines whether the calling user's current password meets policy requirements (e.g. quality, minimum length). The user must be unlocked to perform this check.

Policy requirements which affect this check can be set by admins of the user, but also by the admin of a managed profile associated with the calling user (when the managed profile doesn't have a separate work challenge). When a managed profile has a separate work challenge, its policy requirements only affect the managed profile.

Depending on the user, this method checks the policy requirement against one of the following passwords:

- For the primary user or secondary users: the personal keyguard password.
- For managed profiles: a work challenge if set, otherwise the parent user's personal keyguard password.

  In other words, it's always checking the requirement against the password that is protecting the calling user.

  Note that this method considers all policy requirements targeting the password in question. For example a profile owner might set a requirement on the parent profile i.e. personal keyguard but not on the profile itself. When the device has a weak personal keyguard password and no separate work challenge, calling this method will return `false` despite the profile owner not setting a policy on the profile itself. This is because the profile's current password is the personal keyguard password, and it does not meet all policy requirements.

Device admins must request `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD) before calling this method. Note, this policy type is deprecated for device admins in Android 9.0 (API level 28) or higher.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to determine if the password set on the parent profile is sufficient.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty - i.e. this method will always return false on such devices, provided any password requirements were set.

Returns

`boolean`	`true` if the password meets the policy requirements, `false` otherwise

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the calling application isn't an active admin that us `POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#US
`IllegalStateException` (/reference/java/lang/IllegalStateException)	if the user isn't unlocked

isActivePasswordSufficientForDeviceRequirement

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isActivePasswordSufficientForDeviceRequirement ()
```

Called by profile owner of a managed profile to determine whether the current device password meets policy requirements set explicitly device-wide.

This API is similar to `isActivePasswordSufficient()` (/reference/android/app/admin/DevicePolicyManager#isActivePasswordSufficient()), with two notable differences:

- this API always targets the device password. As a result it should always be called on the `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) instance.
- password policy requirement set on the managed profile is not taken into consideration by this API, even if the device currently does not have a separate work challenge set.

This API is designed to facilite progressive password enrollment flows when the DPC imposes both device and profile password policies. DPC applies profile password policy by calling `setPasswordQuality(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) or `setRequiredPasswordComplexity(int)` (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswordComplexity(int)) on the regular `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance, while it applies device-wide policy by calling `setRequiredPasswordComplexity(int)` (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswordComplexity(int)) on the `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) instance. The DPC can utilize this check to guide the user to set a device password first taking into consideration the device-wide policy only, and then prompt the user to either upgrade it to be fully compliant, or enroll a separate work challenge to satisfy the profile password policy only.

The device user must be unlocked (@link `UserManager#isUserUnlocked(UserHandle)` (/reference/android/os/UserManager#isUserUnlocked(android.os.UserHandle))) to perform this check.

Returns

`boolean`	`true` if the device password meets explicit requirement set on it, `false` otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the calling application is not a profile owner of a managed profile, or if this API is not called on the parent DevicePolicyManager instance.
`IllegalStateException` (/reference/java/lang/IllegalStateException)	if the user isn't unlocked

See also:

`EXTRA_DEVICE_PASSWORD_REQUIREMENT_ONLY` (/reference/android/app/admin/DevicePolicyManager#EXTRA_DEVICE_PASSWORD_REQUIREMENT_ONLY)

isAdminActive

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isAdminActive (ComponentName (/reference/android/content/ComponentName) ad
```

Return true if the given administrator component is currently active (enabled) in the system.

Parameters

`admin`	`ComponentName`: The administrator component to check for. This value cannot be `null`.

Returns

`boolean`	`true` if `admin` is currently enabled in the system, `false` otherwise

isAffiliatedUser

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isAffiliatedUser ()
```

Returns whether this user is affiliated with the device.

By definition, the user that the device owner runs on is always affiliated with the device. Any other user is considered affiliated with the device if the set specified by its profile owner via `setAffiliationIds(ComponentName, Set)` (/reference/android/app/admin/DevicePolicyManager#setAffiliationIds(android.content.ComponentName,%20java.util.Set<java.lang.String>)) intersects with the device owner's.

Returns

`boolean`

See also:

`setAffiliationIds(ComponentName, Set)` (/reference/android/app/admin/DevicePolicyManager#setAffiliationIds(android.content.ComponentName,%20java.util.Set<java.lang.String>))

isAlwaysOnVpnLockdownEnabled

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isAlwaysOnVpnLockdownEnabled (ComponentName (/reference/android/content/Co
```

Called by device or profile owner to query whether current always-on VPN is configured in lockdown mode. Returns `false` when no always-on configuration is set.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

`boolean`

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or a profile owner.

See also:

`setAlwaysOnVpnPackage(ComponentName, String, boolean)` (/reference/android/app/admin/DevicePolicyManager#setAlwaysOnVpnPackage(android.content.ComponentName,%20java.lang.String,%20boolean))

isApplicationHidden

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isApplicationHidden (ComponentName (/reference/android/content/ComponentNa
                String (/reference/java/lang/String) packageName)
```

Determine if a package is hidden. This function can be called by a device owner, profile owner, or by a delegate given the `DELEGATION_PACKAGE_ACCESS` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PACKAGE_ACCESS) scope via `setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>)) .

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance, returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) , where the caller must be the profile owner of an organization-owned managed profile and the package must be a system package. If called on the parent instance, this will determine whether the package is hidden or unhidden in the personal profile.

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), the returned policy will be the current resolved policy rather than the policy set by the calling admin.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if the caller is not a device admin.
`packageName`	`String`: The name of the package to retrieve the hidden status of.

Returns

`boolean`	boolean `true` if the package is hidden, `false` otherwise.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device or profile owner or if called on the parent profile and the **admin** is not a profile owner of an organization-owned managed profile.
IllegalArgumentException (/reference/java/lang/IllegalArgumentException)	if called on the parent profile and the package provided is not a system package.

See also:

setDelegatedScopes(ComponentName, String, List) (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

DELEGATION_PACKAGE_ACCESS (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PACKAGE_ACCESS)

isBackupServiceEnabled

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isBackupServiceEnabled (ComponentName (/reference/android/content/Componer
```

Return whether the backup service is enabled by the device owner or profile owner for the current user, as previously set by setBackupServiceEnabled(android.content.ComponentName, boolean) (/reference/android/app/admin/DevicePolicyManager#setBackupServiceEnabled(android.content.ComponentName,%20boolean)).

Whether the backup functionality is actually enabled or not depends on settings from both the current user and the device owner, please see setBackupServiceEnabled(android.content.ComponentName, boolean) (/reference/android/app/admin/DevicePolicyManager#setBackupServiceEnabled(android.content.ComponentName,%20boolean)) for details.

Backup service manages all backup and restore mechanisms on the device.

Parameters

`admin`	`ComponentName`: This value cannot be `null`.

Returns

`boolean`	`true` if backup service is enabled, `false` otherwise.

See also:

`setBackupServiceEnabled(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setBackupServiceEnabled(android.content.ComponentName,%20boolean))

isCallerApplicationRestrictionsManagingPackage

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isCallerApplicationRestrictionsManagingPackage ()
```

This method was deprecated in API level 26.

From `Build.VERSION_CODES.O` (/reference/android/os/Build.VERSION_CODES#O). Use `getDelegatedScopes(ComponentName, String)` (/reference/android/app/admin/DevicePolicyManager#getDelegatedScopes(android.content.ComponentName,%20java.lang.String)) instead.

Called by any application to find out whether it has been granted permission via `setApplicationRestrictionsManagingPackage(ComponentName, String)`

(/reference/android/app/admin/DevicePolicyManager#setApplicationRestrictionsManagingPackage(android.content.ComponentName,%20java.lang.String))
to manage application restrictions for the calling user.

This is done by comparing the calling Linux uid with the uid of the package specified by that method.

Returns

`boolean`

isCommonCriteriaModeEnabled

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isCommonCriteriaModeEnabled (ComponentName (/reference/android/content/Com
```

Returns whether Common Criteria mode is currently enabled. Device owner and profile owner of an organization-owned managed profile can query its own Common Criteria mode setting by calling this method with its admin `ComponentName` (/reference/android/content/ComponentName). Any caller can obtain the aggregated device-wide Common Criteria mode state by passing `null` as the `admin` argument.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.

Returns

`boolean`	`true` if Common Criteria mode is enabled, `false` otherwise.

isComplianceAcknowledgementRequired

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isComplianceAcknowledgementRequired ()
```

Called by a profile owner of an organization-owned managed profile to query whether it needs to acknowledge device compliance to allow the user to turn the profile off if needed according to the maximum profile time off policy. Normally when acknowledgement is needed the DPC gets a `DeviceAdminReceiver#onComplianceAcknowledgementRequired(Context, Intent)` (/reference/android/app/admin/DeviceAdminReceiver#onComplianceAcknowledgementRequired(android.content.Context,%20android.content.Intent)) callback. But if the callback was not delivered or handled for some reason, this method can be used to verify if acknowledgement is needed.

Returns

`boolean`

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if the user isn't unlocked

See also:

`acknowledgeDeviceCompliant()` (/reference/android/app/admin/DevicePolicyManager#acknowledgeDeviceCompliant())

`setManagedProfileMaximumTimeOff(ComponentName, long)` (/reference/android/app/admin/DevicePolicyManager#setManagedProfileMaximumTimeOff(android.content.ComponentName,%20long))

`DeviceAdminReceiver.onComplianceAcknowledgementRequired(Context, Intent)` (/reference/android/app/admin/DeviceAdminReceiver#onComplianceAcknowledgementRequired(android.content.Context,%20android.content.Intent))

isDeviceFinanced

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isDeviceFinanced ()
```

Returns `true` if this device is marked as a financed device.

A financed device can be entered into lock task mode (see `setLockTaskPackages(ComponentName, String)` (/reference/android/app/admin/DevicePolicyManager#setLockTaskPackages(android.content.ComponentName,%20java.lang.String[]))) by the holder of the role `android.app.role.RoleManager#ROLE_FINANCED_DEVICE_KIOSK`. If this occurs, Device Owners and Profile Owners that have set lock task packages or features, or that attempt to set lock task packages or features, will receive a callback indicating that it could not be set. See `PolicyUpdateReceiver#onPolicyChanged` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicyChanged(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) and `PolicyUpdateReceiver#onPolicySetResult` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicySetResult(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) .

To be informed of changes to this status you can subscribe to the broadcast `ACTION_DEVICE_FINANCING_STATE_CHANGED` (/reference/android/app/admin/DevicePolicyManager#ACTION_DEVICE_FINANCING_STATE_CHANGED).

Returns

`boolean`

Throws

SecurityException (/reference/java/lang/SecurityException)	if the caller is not a device owner, profile owner of an organization-owned managed profile, profile owner on the primary user or holder of one of the following roles: `android.app.role.RoleManager.ROLE_DEVICE_POLICY_MANAGEMENT`, `android.app.role.RoleManager.ROLE_SYSTEM_SUPERVISION`.

isDeviceIdAttestationSupported

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isDeviceIdAttestationSupported ()
```

Returns `true` if the device supports attestation of device identifiers in addition to key attestation. See `generateKeyPair(android.content.ComponentName, java.lang.String, android.security.keystore.KeyGenParameterSpec, int)` (/reference/android/app/admin/DevicePolicyManager#generateKeyPair(android.content.ComponentName,%20java.lang.String,%20android.security.keystore.KeyGenParameterSpec,%20int))

Returns

`boolean`	`true` if Device ID attestation is supported.

isDeviceOwnerApp

Added in API level 18 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isDeviceOwnerApp (String (/reference/java/lang/String) packageName)
```

Used to determine if a particular package has been registered as a Device Owner app. A device owner app is a special device admin that cannot be deactivated by the user, once activated as a device admin. It also cannot be uninstalled. To check whether a particular package is currently registered as the device owner app, pass in the package name from

`Context#getPackageName()` (/reference/android/content/Context#getPackageName()) to this method.

This is useful for device admin apps that want to check whether they are also registered as the device owner app. The exact mechanism by which a device admin app is registered as a device owner app is defined by the setup process.

Parameters

packageName	**String**: the package name of the app, to compare with the registered device owner app, if any.

Returns

boolean	whether or not the package is registered as the device owner app.

isEphemeralUser

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isEphemeralUser (ComponentName (/reference/android/content/ComponentName)
```

Checks if the profile owner is running in an ephemeral user.

Parameters

admin	**ComponentName**: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be **null**.

Returns

`boolean`	whether the profile owner is running in an ephemeral user.

isKeyPairGrantedToWifiAuth

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isKeyPairGrantedToWifiAuth (String (/reference/java/lang/String) alias)
```

Called by a device or profile owner, or delegated certificate chooser (an app that has been delegated the `DELEGATION_CERT_SELECTION` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_SELECTION) privilege), to query whether a KeyChain key pair can be used for authentication to Wifi networks.

Parameters

`alias`	`String`: The alias of the key pair. This value cannot be `null`.

Returns

`boolean`	`true` if the key pair can be used, `false` otherwise.

Throws

SecurityException (/reference/java/lang/SecurityException)	if the caller is not a device owner, a profile owner or delegated certificate chooser.

See also:

`grantKeyPairToWifiAuth(String)`
(/reference/android/app/admin/DevicePolicyManager#grantKeyPairToWifiAuth(java.lang.String))

isLockTaskPermitted

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isLockTaskPermitted (String (/reference/java/lang/String) pkg)
```

This function lets the caller know whether the given component is allowed to start the lock task mode.

Parameters

`pkg`	`String`: The package to check

Returns

`boolean`

isLogoutEnabled

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isLogoutEnabled ()
```

Returns whether logout is enabled by a device owner.

Returns

`boolean`	`true` if logout is enabled by device owner, `false` otherwise.

isManagedProfile

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isManagedProfile (ComponentName (/reference/android/content/ComponentName)
```

Return if this user is a managed profile of another user. An admin can become the profile owner of a managed profile with `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION_MANAGED_PROFILE) and of a managed user with `createAndManageUser(ComponentName, String, ComponentName, PersistableBundle, int)` (/reference/android/app/admin/DevicePolicyManager#createAndManageUser(android.content.ComponentName,%20java.lang.String,%20android.content.ComponentName,%20android.os.PersistableBundle,%20int))

Parameters

`admin`	`ComponentName`: Which profile owner this request is associated with. This value cannot be `null`.

Returns

`boolean`	if this user is a managed profile of another user.

isMasterVolumeMuted

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isMasterVolumeMuted (ComponentName (/reference/android/content/ComponentNa
```

Called by profile or device owners to check whether the global volume mute is on or off.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

`boolean`	`true` if global volume is muted, `false` if it's not.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner.

isMtePolicyEnforced

Added in Android VanillaIceCream (/preview)

```
public static boolean isMtePolicyEnforced ()
```

Get the current MTE state of the device. Learn more about MTE (https://source.android.com/docs/security/test/memory-safety/arm-mte)

Returns

`boolean`	whether MTE is currently enabled on the device.

isNetworkLoggingEnabled

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isNetworkLoggingEnabled (ComponentName (/reference/android/content/Compone
```

Return whether network logging is enabled by a device owner or profile owner of a managed profile.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is assoc only be `null` if the caller is a delegated app with `DELEGATION_NETWORK_` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_NETW or has MANAGE_USERS permission.

Returns

`boolean`	`true` if network logging is enabled by device owner or profile owner, `false` otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device owner or profile owner and caller has no MANAGE_USERS permission

isOrganizationOwnedDeviceWithManagedProfile

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isOrganizationOwnedDeviceWithManagedProfile ()
```

Apps can use this method to find out if the device was provisioned as organization-owend device with a managed profile. This, together with checking whether the device has a device

owner (by calling `isDeviceOwnerApp(String)` (/reference/android/app/admin/DevicePolicyManager#isDeviceOwnerApp(java.lang.String))), could be used to learn whether the device is owned by an organization or an individual: If this method returns true OR `isDeviceOwnerApp(String)` (/reference/android/app/admin/DevicePolicyManager#isDeviceOwnerApp(java.lang.String)) returns true (for any package), then the device is owned by an organization. Otherwise, it's owned by an individual.

Returns

`boolean`	`true` if the device was provisioned as organization-owned device, `false` otherwise.

isOverrideApnEnabled Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isOverrideApnEnabled (ComponentName (/reference/android/content/ComponentN
```

Called by device owner to check if override APNs are currently enabled.

Parameters

`admin`	`ComponentName`: which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with This value cannot be `null`.

Returns

`boolean`	`true` if override APNs are currently enabled, `false` otherwise.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device owner.

See also:

setOverrideApnsEnabled(ComponentName, boolean) (/reference/android/app/admin/DevicePolicyManager#setOverrideApnsEnabled(android.content.ComponentName,%20boolean))

isPackageSuspended

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isPackageSuspended (ComponentName (/reference/android/content/ComponentNam
                String (/reference/java/lang/String) packageName)
```

Determine if a package is suspended. This function can be called by a device owner, profile owner, or by a delegate given the DELEGATION_PACKAGE_ACCESS (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PACKAGE_ACCESS) scope via setDelegatedScopes(ComponentName, String, List) (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>)) or by holders of the permission Manifest.permission.MANAGE_DEVICE_POLICY_PACKAGE_STATE (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_PACKAGE_STATE).

Parameters

admin	ComponentName: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be **null**.

`packageName`	`String`: The name of the package to retrieve the suspended status of.

Returns

`boolean`	`true` if the package is suspended or `false` if the package is not suspended, could not be found or an error occurred.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profil `Manifest.permission.MANA` (/reference/android/Manifest.pe .
`PackageManager.NameNotFoundException` (/reference/android/content/pm/PackageManager.NameNotFoundException)	if the package could not be four

See also:

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`DELEGATION_PACKAGE_ACCESS` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PACKAGE_ACCESS)

isPreferentialNetworkServiceEnabled

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isPreferentialNetworkServiceEnabled ()
```

Indicates whether preferential network service is enabled.

Before Android version `Build.VERSION_CODES.TIRAMISU` (/reference/android/os/Build.VERSION_CODES#TIRAMISU): This method can be called by the profile owner of a managed profile.

Starting from Android version `Build.VERSION_CODES.TIRAMISU` (/reference/android/os/Build.VERSION_CODES#TIRAMISU): This method can be called by the profile owner of a managed profile or device owner.

Returns

`boolean`	whether preferential network service is enabled.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not the profile owner or device owner.

isProfileOwnerApp

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isProfileOwnerApp (String (/reference/java/lang/String) packageName)
```

Used to determine if a particular package is registered as the profile owner for the user. A profile owner is a special device admin that has additional privileges within the profile.

Parameters

`packageName`	`String`: The package name of the app to compare with the registered profile owner.

Returns

`boolean`	Whether or not the package is registered as the profile owner.

isProvisioningAllowed Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isProvisioningAllowed (String (/reference/java/lang/String) action)
```

Returns whether it is possible for the caller to initiate provisioning of a managed profile or device, setting itself as the device or profile owner.

Parameters

`action`	`String`: One of `ACTION_PROVISION_MANAGED_DEVICE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION `ACTION_PROVISION_MANAGED_PROFILE` (/reference/android/app/admin/DevicePolicyManager#ACTION_PROVISION . This value cannot be `null`.

Returns

`boolean`	whether provisioning a managed profile or device is possible.

Throws

`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if the supplied action is not valid.

isResetPasswordTokenActive

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isResetPasswordTokenActive (ComponentName (/reference/android/content/Comp
```

Called by a profile, device owner or a holder of the permission `Manifest.permission.MANAGE_DEVICE_POLICY_RESET_PASSWORD` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_RESET_PASSWORD) to check if the current reset password token is active.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, false is always returned.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.

Returns

`boolean`	true if the token is active, false otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if admin is not a device or profile owner and not a holder of the p `permission.MANAGE_DEVICE_POLICY_RESET_PASSWORD`

(/reference/android/Manifest.permission#MANAGE_DEVICE_POI

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if no token has been set.

isSafeOperation

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isSafeOperation (int reason)
```

Checks if it's safe to run operations that can be affected by the given `reason`.

Note: notice that the operation safety state might change between the time this method returns and the operation's method is called, so calls to the latter could still throw a `UnsafeStateException` (/reference/android/app/admin/UnsafeStateException) even when this method returns `true`.

Parameters

`reason`	`int`: currently, only supported reason is `OPERATION_SAFETY_REASON_DR` (/reference/android/app/admin/DevicePolicyManager#OPERATION_SAFETY . Value is android.app.admin.DevicePolicyManager.OPERATION_SAFETY_RE `SAFETY_REASON_DRIVING_DISTRACTION` (/reference/android/app/admin/DevicePolicyManager#OPERATION_SAFETY

Returns

`boolean`	whether it's safe to run operations that can be affected by the given `reason`.

isSecurityLoggingEnabled

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isSecurityLoggingEnabled (ComponentName (/reference/android/content/Compor
```

Return whether security logging is enabled or not by the admin.

Can only be called by the device owner or a profile owner of an organization-owned managed profile, otherwise a `SecurityException` (/reference/java/lang/SecurityException) will be thrown.

Parameters

`admin`	`ComponentName`: Which device admin this request is associated with. Null if the caller is not a device admin This value may be `null`.

Returns

`boolean`	`true` if security logging is enabled, `false` otherwise.

Throws

SecurityException (/reference/java/lang/SecurityException)	if the caller is not allowed to control security logging.

isStatusBarDisabled

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isStatusBarDisabled ()
```

Returns whether the status bar is disabled/enabled, see `setStatusBarDisabled(ComponentName, boolean)`

(/reference/android/app/admin/DevicePolicyManager#setStatusBarDisabled(android.content.ComponentName,%20boolean))

.

Callable by device owner or profile owner of secondary users that is affiliated with the device owner.

This policy has no effect in LockTask mode. The behavior of the status bar in LockTask mode can be configured with `setLockTaskFeatures(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setLockTaskFeatures(android.content.ComponentName,%20int))

.

This policy also does not have any effect while on the lock screen, where the status bar will not be disabled.

Returns

`boolean`

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not the device owner, or a profile owner of secondary user that is affiliated with the device.

See also:

`isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser())

`getSecondaryUsers(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getSecondaryUsers(android.content.ComponentName))

isUninstallBlocked

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isUninstallBlocked (ComponentName (/reference/android/content/ComponentNam
                String (/reference/java/lang/String) packageName)
```

Check whether the user has been blocked by device policy from uninstalling a package. Requires the caller to be the profile owner if checking a specific admin's policy.

Note: Starting from `Build.VERSION_CODES.LOLLIPOP_MR1` (/reference/android/os/Build.VERSION_CODES#LOLLIPOP_MR1), the behavior of this API is changed such that passing `null` as the `admin` parameter will return if any admin has blocked the uninstallation. Before L MR1, passing `null` will cause a NullPointerException to be raised.

Note: If your app targets Android 11 (API level 30) or higher, this method returns a filtered result. Learn more about how to manage package visibility (/training/basics/intents/package-visibility).

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), the returned policy will be the current resolved policy rather than the policy set by the calling admin.

Parameters

`admin`	`ComponentName`: The name of the admin component whose blocking policy will be checked, or `null` to check whether any admin has blocked the uninstallation. Starting from `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE) admin will be ignored and assumed `null`.
`packageName`	`String`: package to check.

Returns

`boolean`	true if uninstallation is blocked and the given package is visible to you, false otherwise if uninstallation isn't blocked or the given package isn't visible to you.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device or profile owner.

isUniqueDeviceAttestationSupported

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isUniqueDeviceAttestationSupported ()
```

Returns `true` if the StrongBox Keymaster implementation on the device was provisioned with an individual attestation certificate and can sign attestation records using it (as attestation using an individual attestation certificate is a feature only Keymaster implementations with StrongBox security level can implement). For use prior to calling `generateKeyPair(android.content.ComponentName, java.lang.String, android.security.keystore.KeyGenParameterSpec, int)` (/reference/android/app/admin/DevicePolicyManager#generateKeyPair(android.content.ComponentName,%20java.lang.String,%20android.security.keystore.KeyGenParameterSpec,%20int)) .

Returns

`boolean`	`true` if individual attestation is supported.

isUsbDataSignalingEnabled

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isUsbDataSignalingEnabled ()
```

Returns whether USB data signaling is currently enabled.

When called by a device owner or profile owner of an organization-owned managed profile, this API returns whether USB data signaling is currently enabled by that admin. When called by any other app, returns whether USB data signaling is currently enabled on the device.

Returns

`boolean`	`true` if USB data signaling is enabled, `false` otherwise.

isUsingUnifiedPassword

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isUsingUnifiedPassword (ComponentName (/reference/android/content/Componer
```

When called by a profile owner of a managed profile returns true if the profile uses unified challenge with its parent user. **Note**: This method is not concerned with password quality and will return false if the profile has empty password as a separate challenge.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

`boolean`

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a profile owner of a managed profile.

See also:

`UserManager.DISALLOW_UNIFIED_PASSWORD` (/reference/android/os/UserManager#DISALLOW_UNIFIED_PASSWORD)

listForegroundAffiliatedUsers

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<UserHandle (/reference/android/os/UserHandle)> listForegrou
```

Gets the list of `affiliated` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser()) users running on foreground.

Returns

List (/reference/java/util/List)<UserHandle (/reference/android/os/UserHandle)>	list of `affiliated` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser()) users running on foreground. This value cannot be **null**.

Throws

SecurityException (/reference/java/lang/SecurityException)	if the calling application is not a device owner

lockNow

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void lockNow ()
```

Make the device lock immediately, as if the lock screen timeout has expired at the point of this call.

This method secures the device in response to an urgent situation, such as a lost or stolen device. After this method is called, the device must be unlocked using strong authentication (PIN, pattern, or password). This API is intended for use only by device admins.

From version `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) onwards, the caller must either have the LOCK_DEVICE permission or the device must have the device admin feature; if neither is true, then the method will return without completing any action. Before version `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R), the device needed the device admin feature, regardless of the caller's permissions.

The calling device admin must have requested `DeviceAdminInfo#USES_POLICY_FORCE_LOCK` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_FORCE_LOCK) to be able to call this method; if it has not, a security exception will be thrown.

If there's no lock type set, this method forces the device to go to sleep but doesn't lock the device. Device admins who find the device in this state can lock an otherwise-insecure device by first calling `resetPassword(String, int)` (/reference/android/app/admin/DevicePolicyManager#resetPassword(java.lang.String,%20int)) to set the password and then lock the device.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to lock the parent profile.

NOTE: on `automotive builds` (/reference/android/content/pm/PackageManager#FEATURE_AUTOMOTIVE), this method doesn't turn off the screen as it would be a driving safety distraction.

Equivalent to calling `lockNow(int)` (/reference/android/app/admin/DevicePolicyManager#lockNow(int)) with no flags.

Throws

SecurityException (/reference/java/lang/SecurityException)	if the calling application does not own an active administrator that u DeviceAdminInfo#USES_POLICY_FORCE_LOCK (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_FC

lockNow

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void lockNow (int flags)
```

Make the device lock immediately, as if the lock screen timeout has expired at the point of this call.

This method secures the device in response to an urgent situation, such as a lost or stolen device. After this method is called, the device must be unlocked using strong authentication (PIN, pattern, or password). This API is intended for use only by device admins.

From version `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) onwards, the caller must either have the LOCK_DEVICE permission or the device must have the device admin feature; if neither is true, then the method will return without completing any action. Before version `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R), the device needed the device admin feature, regardless of the caller's permissions.

A calling device admin must have requested `DeviceAdminInfo#USES_POLICY_FORCE_LOCK` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_FORCE_LOCK) to be able to call this method; if it has not, a security exception will be thrown.

If there's no lock type set, this method forces the device to go to sleep but doesn't lock the device. Device admins who find the device in this state can lock an otherwise-insecure device by first calling `resetPassword(String, int)` (/reference/android/app/admin/DevicePolicyManager#resetPassword(java.lang.String,%20int)) to set the password and then lock the device.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to lock the parent profile as well as the managed profile.

NOTE: In order to lock the parent profile and evict the encryption key of the managed profile, `lockNow()` (/reference/android/app/admin/DevicePolicyManager#lockNow()) must be called twice: First, `lockNow()` (/reference/android/app/admin/DevicePolicyManager#lockNow()) should be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) , then `lockNow(int)` (/reference/android/app/admin/DevicePolicyManager#lockNow(int)) should be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance associated with the managed profile, with the `FLAG_EVICT_CREDENTIAL_ENCRYPTION_KEY` (/reference/android/app/admin/DevicePolicyManager#FLAG_EVICT_CREDENTIAL_ENCRYPTION_KEY) flag. Calling the method twice in this order ensures that all users are locked and does not stop the device admin on the managed profile from issuing a second call to lock its own profile.

NOTE: on `automotive builds` (/reference/android/content/pm/PackageManager#FEATURE_AUTOMOTIVE), this method doesn't turn off the screen as it would be a driving safety distraction.

Parameters

`flags`	`int`: May be 0 or `FLAG_EVICT_CREDENTIAL_ENCRYPTION_KEY` (/reference/android/app/admin/DevicePolicyManager#FLAG_EVICT_CREDE . Value is either `0` or `FLAG_EVICT_CREDENTIAL_ENCRYPTION_KEY` (/reference/android/app/admin/DevicePolicyManager#FLAG_EVICT_CREDE

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the calling application does not own an active adm `POLICY_FORCE_LOCK` (/reference/android/app/admin/DeviceAdminInfo#US hold the `ERROR(/android.Manifest.permissio EVICT_CREDENTIAL_ENCRYPTION_KEY` (/reference/android/app/admin/DevicePolicyManage flag is passed by an application that is not a profile o

IllegalArgumentException (/reference/java/lang/IllegalArgumentException)	if the FLAG_EVICT_CREDENTIAL_ENCRYPTION_KE (/reference/android/app/admin/DevicePolicyManage flag is passed when locking the parent profile.
UnsupportedOperationException (/reference/java/lang/UnsupportedOperationException)	if the FLAG_EVICT_CREDENTIAL_ENCRYPTION_KE (/reference/android/app/admin/DevicePolicyManage flag is passed when getStorageEncryptionStat (/reference/android/app/admin/DevicePolicyManage return ENCRYPTION_STATUS_ACTIVE_PER_USER (/reference/android/app/admin/DevicePolicyManage

logoutUser

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int logoutUser (ComponentName (/reference/android/content/ComponentName) admin)
```

Called by a profile owner of secondary user that is affiliated with the device to stop the calling user and switch back to primary user (when the user was `switchUser(android.content.ComponentName, android.os.UserHandle)` (/reference/android/app/admin/DevicePolicyManager#switchUser(android.content.ComponentName,%20android.os.UserHandle)) switched to) or stop the user (when it was `started in background` (/reference/android/app/admin/DevicePolicyManager#startUserInBackground(android.content.ComponentName,%20android.os.UserHandle)) .

Notice that on devices running with `headless system user mode` (/reference/android/os/UserManager#isHeadlessSystemUserMode()), there is no primary user, so it switches back to the user that was in the foreground before the first call to `switchUser(android.content.ComponentName, android.os.UserHandle)` (/reference/android/app/admin/DevicePolicyManager#switchUser(android.content.ComponentName,%20android.os.UserHandle)) (or fails with `UserManager#USER_OPERATION_ERROR_UNKNOWN` (/reference/android/os/UserManager#USER_OPERATION_ERROR_UNKNOWN) if that method was not called prior to this call).

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

`int`	one of the following result codes: `UserManager#USER_OPERATION_ERRO` (/reference/android/os/UserManager#USER_OPERATION_ERROR_UNKNOW `UserManager#USER_OPERATION_SUCCESS` (/reference/android/os/UserManager#USER_OPERATION_SUCCESS), `UserManager#USER_OPERATION_ERROR_MANAGED_PROFILE` (/reference/android/os/UserManager#USER_OPERATION_ERROR_MANAGE `UserManager#USER_OPERATION_ERROR_CURRENT_USER` (/reference/android/os/UserManager#USER_OPERATION_ERROR_CURREN is `UserManager.USER_OPERATION_SUCCESS` (/reference/android/os/UserManager#USER_OPERATION_SUCCESS), `User USER_OPERATION_ERROR_UNKNOWN` (/reference/android/os/UserManager#USER_OPERATION_ERROR_UNKNOW `UserManager.USER_OPERATION_ERROR_MANAGED_PROFILE` (/reference/android/os/UserManager#USER_OPERATION_ERROR_MANAGE `UserManager.USER_OPERATION_ERROR_MAX_RUNNING_USERS` (/reference/android/os/UserManager#USER_OPERATION_ERROR_MAX_RU , `UserManager.USER_OPERATION_ERROR_CURRENT_USER` (/reference/android/os/UserManager#USER_OPERATION_ERROR_CURREN `UserManager.USER_OPERATION_ERROR_LOW_STORAGE` (/reference/android/os/UserManager#USER_OPERATION_ERROR_LOW_ST `UserManager.USER_OPERATION_ERROR_MAX_USERS` (/reference/android/os/UserManager#USER_OPERATION_ERROR_MAX_US android.os.UserManager.USER_OPERATION_ERROR_USER_ACCOUNT_ALRE android.os.UserManager.USER_OPERATION_ERROR_DISABLED_USER, or android.os.UserManager.USER_OPERATION_ERROR_PRIVATE_PROFILE

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a profile owner affiliated with the device.

See also:

`getSecondaryUsers(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getSecondaryUsers(android.content.ComponentName))

reboot

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void reboot (ComponentName (/reference/android/content/ComponentName) admin)
```

Called by device owner to reboot the device. If there is an ongoing call on the device, throws an `IllegalStateException` (/reference/java/lang/IllegalStateException).

Parameters

admin	**ComponentName**: Which device owner the request is associated with. This value cannot be **null**.

Throws

IllegalStateException (/reference/java/lang/IllegalStateException)	if device has an ongoing call.
SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device owner.

See also:

`TelephonyManager.CALL_STATE_IDLE` (/reference/android/telephony/TelephonyManager#CALL_STATE_IDLE)

removeActiveAdmin

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void removeActiveAdmin (ComponentName (/reference/android/content/ComponentName) a
```

Remove a current administration component. This can only be called by the application that owns the administration component; if you try to remove someone else's component, a security exception will be thrown.

Note that the operation is not synchronous and the admin might still be active (as indicated by `getActiveAdmins()` (/reference/android/app/admin/DevicePolicyManager#getActiveAdmins())) by the time this method returns.

Parameters

`admin`	**`ComponentName`**: The administration compononent to remove. This value cannot be **`null`**.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not in the owner application of **`admin`**.

removeCrossProfileWidgetProvider

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean removeCrossProfileWidgetProvider (ComponentName (/reference/android/conte
                String (/reference/java/lang/String) packageName)
```

Called by the profile owner of a managed profile or a holder of the permission `Manifest.permission.MANAGE_DEVICE_POLICY_PROFILE_INTERACTION` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_PROFILE_INTERACTION) to disable widget providers from a given package to be available in the parent profile. For this method to take effect the package should have been added via `addCrossProfileWidgetProvider(android.content.ComponentName, java.lang.String)` (/reference/android/app/admin/DevicePolicyManager#addCrossProfileWidgetProvider(android.content.ComponentName,%20java.lang.String)) .

Note: By default no widget provider package is allowlisted.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.
`packageName`	`String`: The package from which widget providers are no longer allowlisted.

Returns

`boolean`	Whether the package was removed.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a profile owner and not a holder of the permission `MANAGE_DEVICE_POLICY_PROFILE_INTERACTION` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC .

See also:

`addCrossProfileWidgetProvider(android.content.ComponentName, String)` (/reference/android/app/admin/DevicePolicyManager#addCrossProfileWidgetProvider(android.content.ComponentName,%20java.lang.String))

`getCrossProfileWidgetProviders(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getCrossProfileWidgetProviders(android.content.ComponentName))

removeKeyPair

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean removeKeyPair (ComponentName (/reference/android/content/ComponentName) ad
                String (/reference/java/lang/String) alias)
```

This API can be called by the following to remove a certificate and private key pair installed under a given alias:

- Device owner
- Profile owner
- Delegated certificate installer
- Credential management app

From Android `Build.VERSION_CODES.S` (/reference/android/os/Build.VERSION_CODES#S), the credential management app can call this API. If called by the credential management app, the componentName must be `null`. Note, there can only be a credential management app on an unmanaged device.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if the caller is not a device admin.

`alias`	`String`: The private key alias under which the certificate is installed. This value cannot be `null`.

Returns

`boolean`	`true` if the private key alias no longer exists, `false` otherwise.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not `null` and not a device or profile owner, or `admin` is null but the calling application is not a delegated certificate installer or credential management app.

See also:

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`DELEGATION_CERT_INSTALL` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_INSTALL)

removeOverrideApn

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean removeOverrideApn (ComponentName (/reference/android/content/ComponentName
                int apnId)
```

Called by device owner or managed profile owner to remove an override APN.

This method may returns `false` if there is no override APN with the given `apnId`.

Before Android version `Build.VERSION_CODES.TIRAMISU` (/reference/android/os/Build.VERSION_CODES#TIRAMISU): Only device owners can remove APNs.

Starting from Android version `Build.VERSION_CODES.TIRAMISU` (/reference/android/os/Build.VERSION_CODES#TIRAMISU): Both device owners and managed profile owners can remove enterprise APNs (`ApnSetting#TYPE_ENTERPRISE` (/reference/android/telephony/data/ApnSetting#TYPE_ENTERPRISE)), while only device owners can remove other type of APNs.

Parameters

`admin`	`ComponentName`: which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with This value cannot be `null`.
`apnId`	`int`: the `id` of the override APN to remove

Returns

`boolean`	`true` if the required override APN is successfully removed, `false` otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	If request is for enterprise APN `admin` is either device owner or profile owner and in all other types of APN if `admin` is not a device owner.

See also:

`setOverrideApnsEnabled(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setOverrideApnsEnabled(android.content.ComponentName,%20boolean))

removeUser

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean removeUser (ComponentName (/reference/android/content/ComponentName) admin
                UserHandle (/reference/android/os/UserHandle) userHandle)
```

Called by a device owner to remove a user/profile and all associated data. The primary user can not be removed.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`userHandle`	`UserHandle`: the user to remove. This value cannot be `null`.

Returns

`boolean`	`true` if the user was removed, `false` otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device owner.

requestBugreport

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean requestBugreport (ComponentName (/reference/android/content/ComponentName)
```

Called by a device owner to request a bugreport.

If the device contains secondary users or profiles, they must be affiliated with the device. Otherwise a `SecurityException` (/reference/java/lang/SecurityException) will be thrown. See `isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser()).

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

`boolean`	`true` if the bugreport collection started successfully, or `false` if it wasn't triggered because a previous bugreport operation is still active (either the bugreport is still running or waiting for the user to share or decline)

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device owner, or there is at least one profile or secondary user that is not affiliated with the device.

See also:

`isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser())

resetPassword

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean resetPassword (String (/reference/java/lang/String) password,
                int flags)
```

This method was deprecated in API level 30.

Please use `resetPasswordWithToken(ComponentName, String, byte, int)` (/reference/android/app/admin/DevicePolicyManager#resetPasswordWithToken(android.content.ComponentName,%20java.lang.String,%20byte[],%20int)) instead.

Force a new password for device unlock (the password needed to access the entire device) or the work profile challenge on the current user. This takes effect immediately.

Before `Build.VERSION_CODES.N` (/reference/android/os/Build.VERSION_CODES#N), this API is available to device admin, profile owner and device owner. Starting from `Build.VERSION_CODES.N` (/reference/android/os/Build.VERSION_CODES#N), legacy device admin (who is not also profile owner or device owner) can only call this API to set a new password if there is currently no password set. Profile owner and device owner can continue to force change an existing password as long as the target user is unlocked, although device owner will not be able to call this API at all if there is also a managed profile on the device.

Between `Build.VERSION_CODES.O` (/reference/android/os/Build.VERSION_CODES#O), `Build.VERSION_CODES.P` (/reference/android/os/Build.VERSION_CODES#P) and `Build.VERSION_CODES.Q` (/reference/android/os/Build.VERSION_CODES#Q), profile owner and devices owner targeting SDK level `Build.VERSION_CODES.O` (/reference/android/os/Build.VERSION_CODES#O) or above who attempt to call this API will receive `SecurityException` (/reference/java/lang/SecurityException); they are encouraged to migrate to the new `resetPasswordWithToken(ComponentName, String, byte, int)` (/reference/android/app/admin/DevicePolicyManager#resetPasswordWithToken(android.content.ComponentName,%20java.lang.String,%20byte[],%20int)) API instead. Profile owner and device owner targeting older SDK levels are not affected: they continue to experience the existing behaviour described in the previous paragraph.

Starting from `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R), this API is no longer supported in most cases. Device owner and profile owner calling this API will receive `SecurityException` (/reference/java/lang/SecurityException) if they target SDK level `Build.VERSION_CODES.O` (/reference/android/os/Build.VERSION_CODES#O) or above, or they will

receive a silent failure (API returning `false`) if they target lower SDK level. For legacy device admins, this API throws `SecurityException` (/reference/java/lang/SecurityException) if they target SDK level `Build.VERSION_CODES.N` (/reference/android/os/Build.VERSION_CODES#N) or above, and returns `false` otherwise. Only privileged apps holding RESET_PASSWORD permission which are part of the system factory image can still call this API to set a new password if there is currently no password set. In this case, if the device already has a password, this API will throw `SecurityException` (/reference/java/lang/SecurityException).

The given password must be sufficient for the current password quality and length constraints as returned by `getPasswordQuality(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPasswordQuality(android.content.ComponentName)) and `getPasswordMinimumLength(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPasswordMinimumLength(android.content.ComponentName)); if it does not meet these constraints, then it will be rejected and false returned. Note that the password may be a stronger quality (containing alphanumeric characters when the requested quality is only numeric), in which case the currently active quality will be increased to match.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, this methods does nothing.

The calling device admin must have requested `DeviceAdminInfo#USES_POLICY_RESET_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_RESET_PASSWORD) to be able to call this method; if it has not, a security exception will be thrown.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Parameters

`password`	`String`: The new password for the user. Null or empty clears the password
`flags`	`int`: May be 0 or combination of `RESET_PASSWORD_REQUIRE_ENTRY` (/reference/android/app/admin/DevicePolicyManager#RESET_PASSWORD_

	PASSWORD_DO_NOT_ASK_CREDENTIALS_ON_BOOT (/reference/android/app/admin/DevicePolicyManager#RESET_PASSWORD_ .

Returns

boolean	Returns true if the password was applied, or false if it is not acceptable for the current constraints.

Throws

SecurityException (/reference/java/lang/SecurityException)	if the calling application does not own an active administrator tha DeviceAdminInfo#USES_POLICY_RESET_PASSWORD (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_
IllegalStateException (/reference/java/lang/IllegalStateException)	if the calling user is locked or has a managed profile.

resetPasswordWithToken

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean resetPasswordWithToken (ComponentName (/reference/android/content/Componer
                String (/reference/java/lang/String) password,
                byte[] token,
                int flags)
```

Called by device or profile owner to force set a new device unlock password or a managed profile challenge on current user. This takes effect immediately.

Unlike `resetPassword(String, int)` (/reference/android/app/admin/DevicePolicyManager#resetPassword(java.lang.String,%20int)), this API can change the password even before the user or device is unlocked or decrypted. The supplied token must have been previously provisioned via

`setResetPasswordToken(ComponentName, byte)` (/reference/android/app/admin/DevicePolicyManager#setResetPasswordToken(android.content.ComponentName,%20byte[])), and in active state `isResetPasswordTokenActive(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#isResetPasswordTokenActive(android.content.ComponentName)).

The given password must be sufficient for the current password quality and length constraints as returned by `getPasswordQuality(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPasswordQuality(android.content.ComponentName)) and `getPasswordMinimumLength(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getPasswordMinimumLength(android.content.ComponentName)); if it does not meet these constraints, then it will be rejected and false returned. Note that the password may be a stronger quality, for example, a password containing alphanumeric characters when the requested quality is only numeric.

Calling with a `null` or empty password will clear any existing PIN, pattern or password if the current password constraints allow it.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, calling this methods has no effect - the password is always empty - and false is returned.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/ associated with. Null if the caller is not a device admin. This value may be n
`password`	`String`: The new password for the user. `null` or empty clears the passwor

`token`	`byte`: the password reset token previously provisioned by setResetPassw (/reference/android/app/admin/DevicePolicyManager#setResetPasswordTo .
`flags`	`int`: May be 0 or combination of RESET_PASSWORD_REQUIRE_ENTRY (/reference/android/app/admin/DevicePolicyManager#RESET_PASSWORD_ ASK_CREDENTIALS_ON_BOOT (/reference/android/app/admin/DevicePolicyManager#RESET_PASSWORD_

Returns

`boolean`	Returns true if the password was applied, or false if it is not acceptable for the current constraints.

Throws

SecurityException (/reference/java/lang/SecurityException)	if admin is not a device or profile owner.
IllegalStateException (/reference/java/lang/IllegalStateException)	if the provided token is not valid.

retrieveNetworkLogs

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<NetworkEvent (/reference/android/app/admin/NetworkEvent)> re
            long batchToken)
```

Called by device owner, profile owner of a managed profile or delegated app with `DELEGATION_NETWORK_LOGGING`

(/reference/android/app/admin/DevicePolicyManager#DELEGATION_NETWORK_LOGGING) to retrieve the most recent batch of network logging events.

When network logging is enabled by a profile owner, the network logs will only include work profile network activity, not activity on the personal profile. A device owner or profile owner has to provide a batchToken provided as part of `DeviceAdminReceiver#onNetworkLogsAvailable` (/reference/android/app/admin/DeviceAdminReceiver#onNetworkLogsAvailable(android.content.Context,%20android.content.Intent,%20long,%20int)) callback. If the token doesn't match the token of the most recent available batch of logs, `null` will be returned.

`NetworkEvent` (/reference/android/app/admin/NetworkEvent) can be one of `DnsEvent` (/reference/android/app/admin/DnsEvent) or `ConnectEvent` (/reference/android/app/admin/ConnectEvent).

The list of network events is sorted chronologically, and contains at most 1200 events.

Access to the logs is rate limited and this method will only return a new batch of logs after the device device owner has been notified via `DeviceAdminReceiver#onNetworkLogsAvailable` (/reference/android/app/admin/DeviceAdminReceiver#onNetworkLogsAvailable(android.content.Context,%20android.content.Intent,%20long,%20int)) .

If the caller is not a profile owner and a secondary user or profile is created, calling this method will throw a `SecurityException` (/reference/java/lang/SecurityException) until all users become affiliated again. It will also no longer be possible to retrieve the network logs batch with the most recent batchToken provided by `DeviceAdminReceiver#onNetworkLogsAvailable` (/reference/android/app/admin/DeviceAdminReceiver#onNetworkLogsAvailable(android.content.Context,%20android.content.Intent,%20long,%20int)) . See `DevicePolicyManager#setAffiliationIds` (/reference/android/app/admin/DevicePolicyManager#setAffiliationIds(android.content.ComponentName,%20java.util.Set<java.lang.String>)) .

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is

	associated with, or `null` if called by a delegated app.
`batchToken`	`long`: A token of the batch to retrieve

Returns

`List` (/reference/java/util/List)<`NetworkEvent` (/reference/android/app/admin/NetworkEvent)>	A new batch of network logs which is a list of `NetworkEvent` (/reference/android/app/admin/NetworkEvent). Returns `null` if the batch represented by batchToken is no longer available or if logging is disabled.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device owner, profile owner or if the `admin` is not a profile owner and there is at least one profile or secondary user that is not affiliated with the device.

See also:

`setAffiliationIds(ComponentName, Set)` (/reference/android/app/admin/DevicePolicyManager#setAffiliationIds(android.content.ComponentName,%20java.util.Set<java.lang.String>))

`DeviceAdminReceiver.onNetworkLogsAvailable(Context, Intent, long, int)` (/reference/android/app/admin/DeviceAdminReceiver#onNetworkLogsAvailable(android.content.Context,%20android.content.Intent,%20long,%20int))

retrievePreRebootSecurityLogs

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<SecurityLog.SecurityEvent (/reference/android/app/admin/Se
```

Called by device owner or profile owner of an organization-owned managed profile to retrieve device logs from before the device's last reboot.

This API is not supported on all devices. Calling this API on unsupported devices will result in `null` being returned. The device logs are retrieved from a RAM region which is not guaranteed to be corruption-free during power cycles, as a result be cautious about data corruption when parsing.

When called by a device owner, if there is any other user or profile on the device, it must be affiliated with the device. Otherwise a `SecurityException` (/reference/java/lang/SecurityException) will be thrown. See `isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser()).

Parameters

`admin`	`ComponentName`: Which device admin this request is associated with, or `null` if called by a delegated app.

Returns

`List` (/reference/java/util/List)<`SecurityLog.SecurityEvent` (/reference/android/app/admin/SecurityLog.SecurityEvent)>	Device logs from before the latest reboot of the system, or `null` if this API is not supported on the device.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not allowed to access security logging, or there is at least one profile or secondary user that is not affiliated with the device.

See also:

`isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser())

`retrieveSecurityLogs(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#retrieveSecurityLogs(android.content.ComponentName))

retrieveSecurityLogs

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<SecurityLog.SecurityEvent (/reference/android/app/admin/Se
```

Called by device owner or profile owner of an organization-owned managed profile to retrieve all new security logging entries since the last call to this API after device boots.

Access to the logs is rate limited and it will only return new logs after the admin has been notified via `DeviceAdminReceiver#onSecurityLogsAvailable` (/reference/android/app/admin/DeviceAdminReceiver#onSecurityLogsAvailable(android.content.Context,%20android.content.Intent)) .

When called by a device owner, if there is any other user or profile on the device, it must be affiliated with the device. Otherwise a `SecurityException` (/reference/java/lang/SecurityException) will be thrown. See `isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser()).

Parameters

`admin`	`ComponentName`: Which device admin this request is associated with, or `null` if called by a delegated app.

Returns

List (/reference/java/util/List)<SecurityLog.SecurityEvent (/reference/android/app/admin/SecurityLog.SecurityEvent)>	the new batch of security logs which is a list of SecurityEvent (/reference/android/app/admin/SecurityLog.Secu, or `null` if rate limitation is exceeded or if logging currently disabled.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not allowed to access security logging, or there is at least one profile or secondary user that is not affiliated with the device.

See also:

`isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser())

`DeviceAdminReceiver.onSecurityLogsAvailable(Context, Intent)` (/reference/android/app/admin/DeviceAdminReceiver#onSecurityLogsAvailable(android.content.Context,%20android.content.Intent))

revokeKeyPairFromApp

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean revokeKeyPairFromApp (ComponentName (/reference/android/content/ComponentN
                String (/reference/java/lang/String) alias,
                String (/reference/java/lang/String) packageName)
```

Called by a device or profile owner, or delegated certificate chooser (an app that has been delegated the `DELEGATION_CERT_SELECTION` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_SELECTION) privilege), to revoke an application's grant to a KeyChain key pair. Calls by the application to `KeyChain.getPrivateKey(Context, String)` (/reference/android/security/KeyChain#getPrivateKey(android.content.Context,%20java.lang.String)) will fail after the grant is revoked. The grantee app will receive the `KeyChain.ACTION_KEY_ACCESS_CHANGED` (/reference/android/security/KeyChain#ACTION_KEY_ACCESS_CHANGED) broadcast when access to a key is revoked. Starting from `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE) throws an `IllegalArgumentException` (/reference/java/lang/IllegalArgumentException) if `alias` doesn't correspond to an existing key.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if calling from a delegated certificate chooser.
`alias`	`String`: The alias of the key to revoke access from. This value cannot be `null`.
`packageName`	`String`: The name of the (already installed) package to revoke access from. This value cannot be `null`.

Returns

`boolean`	`true` if the grant was revoked successfully, `false` otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not a device owner, a profile owner or delegated certificate chooser.
`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if `packageName` or `alias` are empty, or if `packageName` is not a name of an installed package.

See also:

`grantKeyPairToApp(ComponentName, String, String)` (/reference/android/app/admin/DevicePolicyManager#grantKeyPairToApp(android.content.ComponentName,%20java.lang.String,%20java.lang.String))

revokeKeyPairFromWifiAuth

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean revokeKeyPairFromWifiAuth (String (/reference/java/lang/String) alias)
```

Called by a device or profile owner, or delegated certificate chooser (an app that has been delegated the `DELEGATION_CERT_SELECTION` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_SELECTION) privilege), to deny using a KeyChain key pair for authentication to Wifi networks. Configured networks using this key won't be able to authenticate. Starting from `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE) throws an `IllegalArgumentException` (/reference/java/lang/IllegalArgumentException) if `alias` doesn't correspond to an existing key.

Parameters

`alias`	`String`: The alias of the key pair. This value cannot be `null`.

Returns

`boolean`	`true` if the operation was set successfully, `false` otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not a device owner, a profile owner or delegated certificate chooser.

See also:

`grantKeyPairToWifiAuth(String)` (/reference/android/app/admin/DevicePolicyManager#grantKeyPairToWifiAuth(java.lang.String))

setAccountManagementDisabled

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setAccountManagementDisabled (ComponentName (/reference/android/content/Compo
                String (/reference/java/lang/String) accountType,
                boolean disabled)
```

Called by a device owner or profile owner to disable account management for a specific type of account.

The calling device admin must be a device owner or profile owner. If it is not, a security exception will be thrown.

When account management is disabled for an account type, adding or removing an account of that type will not be possible.

From `Build.VERSION_CODES.N` (/reference/android/os/Build.VERSION_CODES#N) the profile or device owner can still use `AccountManager` (/reference/android/accounts/AccountManager) APIs to add or remove accounts when account management for a specific type is disabled.

This method may be called on the `DevicePolicyManager` instance returned from `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) by the profile owner on an organization-owned device, to restrict accounts that may not be managed on the primary profile.

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), after the account management disabled policy has been set, `PolicyUpdateReceiver#onPolicySetResult(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicySetResult(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin on whether the policy was successfully set or not. This callback will contain:

- The policy identifier `DevicePolicyIdentifiers#ACCOUNT_MANAGEMENT_DISABLED_POLICY` (/reference/android/app/admin/DevicePolicyIdentifiers#ACCOUNT_MANAGEMENT_DISABLED_POLICY)

- The additional policy params bundle, which contains `PolicyUpdateReceiver#EXTRA_ACCOUNT_TYPE` (/reference/android/app/admin/PolicyUpdateReceiver#EXTRA_ACCOUNT_TYPE) the account type the policy applies to
- The `TargetUser` (/reference/android/app/admin/TargetUser) that this policy relates to
- The `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult), which will be `PolicyUpdateResult#RESULT_POLICY_SET` (/reference/android/app/admin/PolicyUpdateResult#RESULT_POLICY_SET) if the policy was successfully set or the reason the policy failed to be set (e.g. `PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY` (/reference/android/app/admin/PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY))

If there has been a change to the policy, `PolicyUpdateReceiver#onPolicyChanged(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicyChanged(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin of this change. This callback will contain the same parameters as PolicyUpdateReceiver#onPolicySetResult and the `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult) will contain the reason why the policy changed.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.
`accountType`	`String`: For which account management is disabled or enabled.
`disabled`	`boolean`: The boolean indicating that account management will be disabled (true) or enabled (false).

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device or profile owner.

setAffiliationIds

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setAffiliationIds (ComponentName (/reference/android/content/ComponentName) a
                Set (/reference/java/util/Set)<String (/reference/java/lang/String)> ids)
```

Indicates the entity that controls the device. Two users are affiliated if the set of ids set by the device owner and the admin of the secondary user.

A user that is affiliated with the device owner user is considered to be affiliated with the device.

Note: Features that depend on user affiliation (such as security logging or `bindDeviceAdminServiceAsUser(ComponentName, Intent, ServiceConnection, BindServiceFlags, UserHandle)` (/reference/android/app/admin/DevicePolicyManager#bindDeviceAdminServiceAsUser(android.content.ComponentName,%20android.content.Intent,%20android.content.ServiceConnection,%20android.content.Context.BindServiceFlags,%20android.os.UserHandle))) won't be available when a secondary user is created, until it becomes affiliated. Therefore it is recommended that the appropriate affiliation ids are set by its owner as soon as possible after the user is created.

Note: This method used to be available for affiliating device owner and profile owner. However, since Android 11, this combination is not possible. This method is now only useful for affiliating the primary user with managed secondary users.

Parameters

admin	**ComponentName**: Which device owner, or owner of secondary user, this request is associated with. This value cannot be **null**.

`ids`	`Set`: A set of opaque non-empty affiliation ids. This value cannot be `null`.

Throws

`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if `ids` is null or contains an empty string.

See also:

`isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser())

setAlwaysOnVpnPackage added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setAlwaysOnVpnPackage (ComponentName (/reference/android/content/ComponentNam
                String (/reference/java/lang/String) vpnPackage,
                boolean lockdownEnabled)
```

Called by a device or profile owner to configure an always-on VPN connection through a specific application for the current user. This connection is automatically granted and persisted after a reboot.

To support the always-on feature, an app must

- declare a `VpnService` (/reference/android/net/VpnService) in its manifest, guarded by `Manifest.permission.BIND_VPN_SERVICE` (/reference/android/Manifest.permission#BIND_VPN_SERVICE);
- target `API 24` (/reference/android/os/Build.VERSION_CODES#N) or above; and
- *not* explicitly opt out of the feature through `VpnService.SERVICE_META_DATA_SUPPORTS_ALWAYS_ON` (/reference/android/net/VpnService#SERVICE_META_DATA_SUPPORTS_ALWAYS_ON).

The call will fail if called with the package name of an unsupported VPN app.

Enabling lockdown via `lockdownEnabled` argument carries the risk that any failure of the VPN provider could break networking for all apps. This method clears any lockdown allowlist set by `setAlwaysOnVpnPackage(android.content.ComponentName, java.lang.String, boolean, java.util.Set)` (/reference/android/app/admin/DevicePolicyManager#setAlwaysOnVpnPackage(android.content.ComponentName,%20java.lang.String,%20boolean,%20java.util.Set<java.lang.String>)) .

Starting from `API 31` (/reference/android/os/Build.VERSION_CODES#S) calling this method with `vpnPackage` set to `null` only removes the existing configuration if it was previously created by this admin. To remove VPN configuration created by the user use `UserManager#DISALLOW_CONFIG_VPN` (/reference/android/os/UserManager#DISALLOW_CONFIG_VPN).

Parameters

`admin`	`ComponentName`: This value cannot be `null`.
`vpnPackage`	`String`: The package name for an installed VPN app on the device, or `null` to remove an existing always-on VPN configuration.
`lockdownEnabled`	`boolean`: `true` to disallow networking when the VPN is not connected or `false` otherwise. This has no effect when clearing.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or a profile owner.
`PackageManager.NameNotFoundException` (/reference/android/content/pm/PackageManager.NameNotFoundException)	if `vpnPackage` is not installed.
`UnsupportedOperationException` (/reference/java/lang/UnsupportedOperationException)	if `vpnPackage` exists but does not support being set as always

on, or if always-on VPN is not available.

See also:

`setAlwaysOnVpnPackage(ComponentName, String, boolean, Set)` (/reference/android/app/admin/DevicePolicyManager#setAlwaysOnVpnPackage(android.content.ComponentName,%20java.lang.String,%20boolean,%20java.util.Set<java.lang.String>))

setAlwaysOnVpnPackage

added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setAlwaysOnVpnPackage (ComponentName (/reference/android/content/ComponentNam
                String (/reference/java/lang/String) vpnPackage,
                boolean lockdownEnabled,
                Set (/reference/java/util/Set)<String (/reference/java/lang/String)> lockdownAllow
```

A version of `setAlwaysOnVpnPackage(android.content.ComponentName, java.lang.String, boolean)` (/reference/android/app/admin/DevicePolicyManager#setAlwaysOnVpnPackage(android.content.ComponentName,%20java.lang.String,%20boolean)) that allows the admin to specify a set of apps that should be able to access the network directly when VPN is not connected. When VPN connects these apps switch over to VPN if allowed to use that VPN. System apps can always bypass VPN.

Note that the system doesn't update the allowlist when packages are installed or uninstalled, the admin app must call this method to keep the list up to date.

When `lockdownEnabled` is false `lockdownAllowlist` is ignored . When `lockdownEnabled` is `true` and `lockdownAllowlist` is `null` or empty, only system apps can bypass VPN.

Setting always-on VPN package to `null` or using `setAlwaysOnVpnPackage(android.content.ComponentName, java.lang.String, boolean)` (/reference/android/app/admin/DevicePolicyManager#setAlwaysOnVpnPackage(android.content.ComponentName,%20java.lang.String,%20boolean)) clears lockdown allowlist.

Parameters

`admin`	`ComponentName`: This value cannot be `null`.
`vpnPackage`	`String`: package name for an installed VPN app on the device, or `null` to remove an existing always-on VPN configuration
`lockdownEnabled`	`boolean`: `true` to disallow networking when the VPN is not connected or `false` otherwise. This has no effect when clearing.
`lockdownAllowlist`	`Set`: Packages that will be able to access the network directly when VPN is in lockdown mode but not connected. Has no effect when clearing. This value may be `null`.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or a profile owner.
`PackageManager.NameNotFoundException` (/reference/android/content/pm/PackageManager.NameNotFoundException)	if `vpnPackage` or one of `lockdownAllowlist` is not installed.
`UnsupportedOperationException` (/reference/java/lang/UnsupportedOperationException)	if `vpnPackage` exists but does not support being set as always on, or if always-on VPN is not available.

setApplicationHidden

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean setApplicationHidden (ComponentName (/reference/android/content/ComponentN
                String (/reference/java/lang/String) packageName,
                boolean hidden)
```

Hide or unhide packages. When a package is hidden it is unavailable for use, but the data and actual package file remain. This function can be called by a device owner, profile owner, or by a delegate given the `DELEGATION_PACKAGE_ACCESS` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PACKAGE_ACCESS) scope via `setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>)) .

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance, returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) , where the caller must be the profile owner of an organization-owned managed profile and the package must be a system package. If called on the parent instance, then the package is hidden or unhidden in the personal profile.

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), after the application hidden policy has been set, `PolicyUpdateReceiver#onPolicySetResult(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicySetResult(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin on whether the policy was successfully set or not. This callback will contain:

- The policy identifier `DevicePolicyIdentifiers#APPLICATION_HIDDEN_POLICY` (/reference/android/app/admin/DevicePolicyIdentifiers#APPLICATION_HIDDEN_POLICY)
- The additional policy params bundle, which contains `PolicyUpdateReceiver#EXTRA_PACKAGE_NAME` (/reference/android/app/admin/PolicyUpdateReceiver#EXTRA_PACKAGE_NAME) the package name the policy applies to
- The `TargetUser` (/reference/android/app/admin/TargetUser) that this policy relates to

- The `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult), which will be `PolicyUpdateResult#RESULT_POLICY_SET` (/reference/android/app/admin/PolicyUpdateResult#RESULT_POLICY_SET) if the policy was successfully set or the reason the policy failed to be set (e.g. `PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY` (/reference/android/app/admin/PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY))
)

If there has been a change to the policy, `PolicyUpdateReceiver#onPolicyChanged(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicyChanged(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin of this change. This callback will contain the same parameters as PolicyUpdateReceiver#onPolicySetResult and the `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult) will contain the reason why the policy changed.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if the caller is not a device admin.
`packageName`	`String`: The name of the package to hide or unhide.
`hidden`	`boolean`: `true` if the package should be hidden, `false` if it should be unhidden.

Returns

`boolean`	boolean Whether the hidden setting of the package was successfully updated.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if **`admin`** is not a device or profile owner or if called on the parent profile and the **`admin`** is not a profile owner of an organization-owned managed profile.
`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if called on the parent profile and the package provided is not a system package.

See also:

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`DELEGATION_PACKAGE_ACCESS` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PACKAGE_ACCESS)

setApplicationRestrictions

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setApplicationRestrictions (ComponentName (/reference/android/content/Compone
                String (/reference/java/lang/String) packageName,
                Bundle (/reference/android/os/Bundle) settings)
```

Sets the application restrictions for a given target application running in the calling user.

The caller must be a profile or device owner on that user, or the package allowed to manage application restrictions via `setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>)) with the `DELEGATION_APP_RESTRICTIONS` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_APP_RESTRICTIONS) scope; otherwise a security exception will be thrown.

The provided `Bundle` (/reference/android/os/Bundle) consists of key-value pairs, where the types of values may be:

- `boolean`
- `int`
- `String` or `String[]`
- From `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M), `Bundle` or `Bundle[]`

If the restrictions are not available yet, but may be applied in the near future, the caller can notify the target application of that by adding `UserManager#KEY_RESTRICTIONS_PENDING` (/reference/android/os/UserManager#KEY_RESTRICTIONS_PENDING) to the settings parameter.

The application restrictions are only made visible to the target application via `UserManager#getApplicationRestrictions(String)` (/reference/android/os/UserManager#getApplicationRestrictions(java.lang.String)), in addition to the profile or device owner, and the application restrictions managing package via `getApplicationRestrictions(ComponentName, String)` (/reference/android/app/admin/DevicePolicyManager#getApplicationRestrictions(android.content.ComponentName,%20java.lang.String))
.

Starting from Android Version `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), multiple admins can set app restrictions for the same application, the target application can get the list of app restrictions set by each admin via `RestrictionsManager.getApplicationRestrictionsPerAdmin()` (/reference/android/content/RestrictionsManager#getApplicationRestrictionsPerAdmin()).

NOTE: The method performs disk I/O and shouldn't be called on the main thread
This method may take several seconds to complete, so it should only be called from a worker thread.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if called by the application restrictions managing package.

`packageName`	`String`: The name of the package to update restricted settings for.
`settings`	`Bundle`: A `Bundle` (/reference/android/os/Bundle) to be parsed by the receiving application, conveying a new set of active restrictions.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner.

See also:

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`DELEGATION_APP_RESTRICTIONS` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_APP_RESTRICTIONS)

`UserManager.KEY_RESTRICTIONS_PENDING` (/reference/android/os/UserManager#KEY_RESTRICTIONS_PENDING)

setApplicationRestrictionsManagingPackage

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setApplicationRestrictionsManagingPackage (ComponentName (/reference/andro
            String (/reference/java/lang/String) packageName)
```

This method was deprecated in API level 26.

From `Build.VERSION_CODES.O` (/reference/android/os/Build.VERSION_CODES#O). Use `setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

with the **DELEGATION_APP_RESTRICTIONS** (/reference/android/app/admin/DevicePolicyManager#DELEGATION_APP_RESTRICTIONS) scope instead.

Called by a profile owner or device owner to grant permission to a package to manage application restrictions for the calling user via `setApplicationRestrictions(ComponentName, String, Bundle)` (/reference/android/app/admin/DevicePolicyManager#setApplicationRestrictions(android.content.ComponentName,%20java.lang.String,%20android.os.Bundle)) and `getApplicationRestrictions(ComponentName, String)` (/reference/android/app/admin/DevicePolicyManager#getApplicationRestrictions(android.content.ComponentName,%20java.lang.String)).

This permission is persistent until it is later cleared by calling this method with a `null` value or uninstalling the managing package.

The supplied application restriction managing package must be installed when calling this API, otherwise an `NameNotFoundException` (/reference/android/content/pm/PackageManager.NameNotFoundException) will be thrown.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`packageName`	`String`: The package name which will be given access to application restrictions APIs. If `null` is given the current package will be cleared.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profil owner.

`PackageManager.NameNotFoundException` (/reference/android/content/pm/PackageManager.NameNotFoundException)	if `packageName` is not found

setAutoTimeEnabled

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setAutoTimeEnabled (ComponentName (/reference/android/content/ComponentName)
                boolean enabled)
```

Called by a device owner, a profile owner for the primary user or a profile owner of an organization-owned managed profile to turn auto time on and off. Callers are recommended to use `UserManager#DISALLOW_CONFIG_DATE_TIME` (/reference/android/os/UserManager#DISALLOW_CONFIG_DATE_TIME) to prevent the user from changing this setting.

If user restriction `UserManager#DISALLOW_CONFIG_DATE_TIME` (/reference/android/os/UserManager#DISALLOW_CONFIG_DATE_TIME) is used, no user will be able set the date and time. Instead, the network date and time will be used.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.
`enabled`	`boolean`: Whether time should be obtained automatically from the network or not.

Throws

SecurityException (/reference/java/lang/SecurityException)	if caller is not a device owner, a profile owner for the primary user, or a profile owner of an organization-owned managed profile.

setAutoTimeRequired

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setAutoTimeRequired (ComponentName (/reference/android/content/ComponentName)
                boolean required)
```

This method was deprecated in API level 30.
From `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R). Use `setAutoTimeEnabled(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setAutoTimeEnabled(android.content.ComponentName,%20boolean)) to turn auto time on or off and use `UserManager#DISALLOW_CONFIG_DATE_TIME` (/reference/android/os/UserManager#DISALLOW_CONFIG_DATE_TIME) to prevent the user from changing this setting.

Called by a device owner, or alternatively a profile owner from Android 8.0 (API level 26) or higher, to set whether auto time is required. If auto time is required, no user will be able set the date and time and network date and time will be used.

Note: If auto time is required the user can still manually set the time zone. Staring from Android 11, if auto time is required, the user cannot manually set the time zone.

The calling device admin must be a device owner, or alternatively a profile owner from Android 8.0 (API level 26) or higher. If it is not, a security exception will be thrown.

Staring from Android 11, this API switches to use `UserManager#DISALLOW_CONFIG_DATE_TIME` (/reference/android/os/UserManager#DISALLOW_CONFIG_DATE_TIME) to enforce the auto time settings. Calling this API to enforce auto time will result in `UserManager#DISALLOW_CONFIG_DATE_TIME` (/reference/android/os/UserManager#DISALLOW_CONFIG_DATE_TIME) being set, while calling this API to lift the requirement will result in `UserManager#DISALLOW_CONFIG_DATE_TIME` (/reference/android/os/UserManager#DISALLOW_CONFIG_DATE_TIME) being cleared. From Android 11, this API can also no longer be called on a managed profile.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`required`	`boolean`: Whether auto time is set required or not.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device owner, not a profile owner or if this API is called on a managed profile.

setAutoTimeZoneEnabled added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setAutoTimeZoneEnabled (ComponentName (/reference/android/content/ComponentNa
                boolean enabled)
```

Called by a device owner, a profile owner for the primary user or a profile owner of an organization-owned managed profile to turn auto time zone on and off. Callers are recommended to use `UserManager#DISALLOW_CONFIG_DATE_TIME` (/reference/android/os/UserManager#DISALLOW_CONFIG_DATE_TIME) to prevent the user from changing this setting.

If user restriction `UserManager#DISALLOW_CONFIG_DATE_TIME` (/reference/android/os/UserManager#DISALLOW_CONFIG_DATE_TIME) is used, no user will be able set the date and time zone. Instead, the network date and time zone will be used.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with or Null if the caller is not a device admin. This value may be `null`.
`enabled`	`boolean`: Whether time zone should be obtained automatically from the network or not.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if caller is not a device owner, a profile owner for the primary user, or a profile owner of an organization-owned managed profile.

setBackupServiceEnabled

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setBackupServiceEnabled (ComponentName (/reference/android/content/ComponentN
                boolean enabled)
```

Allows the device owner or profile owner to enable or disable the backup service.

Each user has its own backup service which manages the backup and restore mechanisms in that user. Disabling the backup service will prevent data from being backed up or restored.

Device owner calls this API to control backup services across all users on the device. Profile owner can use this API to enable or disable the profile's backup service. However, for a managed profile its backup functionality is only enabled if both the device owner and the profile owner have enabled the backup service.

By default, backup service is disabled on a device with device owner, and within a managed profile.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`enabled`	`boolean`: `true` to enable the backup service, `false` to disable it.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device owner or a profile owner.

setBluetoothContactSharingDisabled

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setBluetoothContactSharingDisabled (ComponentName (/reference/android/conten
                boolean disabled)
```

Called by a profile owner of a managed profile to set whether bluetooth devices can access enterprise contacts.

The calling device admin must be a profile owner. If it is not, a security exception will be thrown.

This API works on managed profile only.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

`disabled`	**`boolean`**: If true, bluetooth devices cannot access enterprise contacts.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **`admin`** is not a profile owner.

setCameraDisabled

Added in API level 14 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCameraDisabled (ComponentName (/reference/android/content/ComponentName) a
                boolean disabled)
```

Called by an application that is administering the device to disable all cameras on the device, for this user. After setting this, no applications running as this user will be able to access any cameras on the device.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance, returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)), where the caller must be the profile owner of an organization-owned managed profile.

If the caller is device owner, then the restriction will be applied to all users. If called on the parent instance, then the restriction will be applied on the personal profile.

The calling device admin must have requested `DeviceAdminInfo#USES_POLICY_DISABLE_CAMERA` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_DISABLE_CAMERA) to be able to call this method; if it has not, a security exception will be thrown.

Note, this policy type is deprecated for legacy device admins since `Build.VERSION_CODES.Q` (/reference/android/os/Build.VERSION_CODES#Q). On Android `Build.VERSION_CODES.Q`

(/reference/android/os/Build.VERSION_CODES#Q) devices, legacy device admins targeting SDK version `Build.VERSION_CODES.P` (/reference/android/os/Build.VERSION_CODES#P) or below can still call this API to disable camera, while legacy device admins targeting SDK version `Build.VERSION_CODES.Q` (/reference/android/os/Build.VERSION_CODES#Q) will receive a SecurityException. Starting from Android `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R), requests to disable camera from legacy device admins targeting SDK version `Build.VERSION_CODES.P` (/reference/android/os/Build.VERSION_CODES#P) or below will be silently ignored.

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), after the camera disabled policy has been set, `PolicyUpdateReceiver#onPolicySetResult(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicySetResult(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin on whether the policy was successfully set or not. This callback will contain:

- The policy identifier: userRestriction_no_camera
- The `TargetUser` (/reference/android/app/admin/TargetUser) that this policy relates to
- The `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult), which will be `PolicyUpdateResult#RESULT_POLICY_SET` (/reference/android/app/admin/PolicyUpdateResult#RESULT_POLICY_SET) if the policy was successfully set or the reason the policy failed to be set (e.g. `PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY` (/reference/android/app/admin/PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY)))

If there has been a change to the policy, `PolicyUpdateReceiver#onPolicyChanged(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicyChanged(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin of this change. This callback will contain the same parameters as PolicyUpdateReceiver#onPolicySetResult and the `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult) will contain the reason why the policy changed.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with or null if the caller is not a device admin
`disabled`	`boolean`: Whether or not the camera should be disabled.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not an active administrator or does not use `DeviceAdmi` `POLICY_DISABLE_CAMERA` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_DI .

setCertInstallerPackage

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCertInstallerPackage (ComponentName (/reference/android/content/ComponentN
                String (/reference/java/lang/String) installerPackage)
```

This method was deprecated in API level 26.

From `Build.VERSION_CODES.O` (/reference/android/os/Build.VERSION_CODES#O). Use `setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>)) with the `DELEGATION_CERT_INSTALL` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_INSTALL) scope instead.

Called by a profile owner or device owner to grant access to privileged certificate manipulation APIs to a third-party certificate installer app. Granted APIs include

`getInstalledCaCerts(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getInstalledCaCerts(android.content.ComponentName)), `hasCaCertInstalled(ComponentName, byte)` (/reference/android/app/admin/DevicePolicyManager#hasCaCertInstalled(android.content.ComponentName,%20byte[])), `installCaCert(ComponentName, byte)` (/reference/android/app/admin/DevicePolicyManager#installCaCert(android.content.ComponentName,%20byte[])), `uninstallCaCert(ComponentName, byte)` (/reference/android/app/admin/DevicePolicyManager#uninstallCaCert(android.content.ComponentName,%20byte[])), `uninstallAllUserCaCerts(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#uninstallAllUserCaCerts(android.content.ComponentName)) and `installKeyPair(ComponentName, PrivateKey, Certificate, String)` (/reference/android/app/admin/DevicePolicyManager#installKeyPair(android.content.ComponentName,%20java.security.PrivateKey,%20java.security.cert.Certificate,%20java.lang.String)).

Delegated certificate installer is a per-user state. The delegated access is persistent until it is later cleared by calling this method with a null value or uninstallling the certificate installer.

Note: Starting from `Build.VERSION_CODES.N` (/reference/android/os/Build.VERSION_CODES#N), if the caller application's target SDK version is `Build.VERSION_CODES.N` (/reference/android/os/Build.VERSION_CODES#N) or newer, the supplied certificate installer package must be installed when calling this API, otherwise an `IllegalArgumentException` (/reference/java/lang/IllegalArgumentException) will be thrown.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`installerPackage`	`String`: The package name of the certificate installer which will be given access. If `null` is given the current package will be cleared.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not a device or a profile owner.

setCommonCriteriaModeEnabled

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCommonCriteriaModeEnabled (ComponentName (/reference/android/content/Compo
                boolean enabled)
```

Called by device owner or profile owner of an organization-owned managed profile to toggle Common Criteria mode for the device. When the device is in Common Criteria mode, certain device functionalities are tuned to meet the higher security level required by Common Criteria certification. For example:

- Bluetooth long term key material is additionally integrity-protected with AES-GCM.
- WiFi configuration store is additionally integrity-protected with AES-GCM.

Common Criteria mode is disabled by default.

Note: if Common Critera mode is turned off after being enabled previously, all existing WiFi configurations will be lost.

Parameters

`admin`	`ComponentName`: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.
`enabled`	`boolean`: whether Common Criteria mode should be enabled or not.

setConfiguredNetworksLockdownState

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setConfiguredNetworksLockdownState (ComponentName (/reference/android/conten
                boolean lockdown)
```

Called by a device owner or a profile owner of an organization-owned managed profile to control whether the user can change networks configured by the admin. When this lockdown is enabled, the user can still configure and connect to other Wi-Fi networks, or use other Wi-Fi capabilities such as tethering.

WiFi network configuration lockdown is controlled by a global settings `Settings.Global.WIFI_DEVICE_OWNER_CONFIGS_LOCKDOWN` (/reference/android/provider/Settings.Global#WIFI_DEVICE_OWNER_CONFIGS_LOCKDOWN) and calling this API effectively modifies the global settings. Previously device owners can also control this directly via `setGlobalSetting(ComponentName, String, String)` (/reference/android/app/admin/DevicePolicyManager#setGlobalSetting(android.content.ComponentName,%20java.lang.String,%20java.lang.String)) but they are recommended to switch to this API.

Parameters

`admin`	`ComponentName`: admin Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.
`lockdown`	`boolean`: Whether the admin configured networks should be unmodifiable by the user.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if caller is not a device owner or a profile owner of an organization-owned managed profile.

setContentProtectionPolicy

Added in Android VanillaIceCream (/preview)

```
public void setContentProtectionPolicy (ComponentName (/reference/android/content/Compone
                int policy)
```

Sets the content protection policy which controls scanning for deceptive apps.

This function can only be called by the device owner, a profile owner of an affiliated user or profile, or the profile owner when no device owner is set or holders of the permission `Manifest.permission.MANAGE_DEVICE_POLICY_CONTENT_PROTECTION` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_CONTENT_PROTECTION). See `isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser()). Any policy set via this method will be cleared if the user becomes unaffiliated.

After the content protection policy has been set, `PolicyUpdateReceiver#onPolicySetResult(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicySetResult(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin on whether the policy was successfully set or not. This callback will contain:

- The policy identifier `DevicePolicyIdentifiers#CONTENT_PROTECTION_POLICY` (/reference/android/app/admin/DevicePolicyIdentifiers#CONTENT_PROTECTION_POLICY)
- The `TargetUser` (/reference/android/app/admin/TargetUser) that this policy relates to
- The `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult), which will be `PolicyUpdateResult#RESULT_POLICY_SET` (/reference/android/app/admin/PolicyUpdateResult#RESULT_POLICY_SET) if the policy was successfully set or the reason the policy failed to be set (e.g. `PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY` (/reference/android/app/admin/PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY)))

If there has been a change to the policy, `PolicyUpdateReceiver#onPolicyChanged(Context, String, Bundle, TargetUser, PolicyUpdateResult)`

(/reference/android/app/admin/PolicyUpdateReceiver#onPolicyChanged(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult))
will notify the admin of this change. This callback will contain the same parameters as PolicyUpdateReceiver#onPolicySetResult and the `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult) will contain the reason why the policy changed.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/ request is associated with. Null if the caller is not a device admin. This value
`policy`	`int`: The content protection policy to set. One of `CONTENT_PROTECTION_D` (/reference/android/app/admin/DevicePolicyManager#CONTENT_PROTECT , `CONTENT_PROTECTION_DISABLED` (/reference/android/app/admin/DevicePolicyManager#CONTENT_PROTECT `PROTECTION_ENABLED` (/reference/android/app/admin/DevicePolicyManager#CONTENT_PROTECT `PROTECTION_NOT_CONTROLLED_BY_POLICY` (/reference/android/app/admin/DevicePolicyManager#CONTENT_PROTECT , `CONTENT_PROTECTION_DISABLED` (/reference/android/app/admin/DevicePolicyManager#CONTENT_PROTECT `PROTECTION_ENABLED` (/reference/android/app/admin/DevicePolicyManager#CONTENT_PROTECT

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not the device owner, the profile owner of an affiliated us owner when no device owner is set or holder of the permission `MANAGE_DEVICE_POLICY_CONTENT_PROTECTION` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC .

See also:

`isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser())

setCredentialManagerPolicy

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCredentialManagerPolicy (PackagePolicy (/reference/android/app/admin/Packa
```

Called by a device owner or profile owner of a managed profile to set the credential manager policy.

Affects APIs exposed by `CredentialManager` (/reference/android/credentials/CredentialManager).

A `PackagePolicy#PACKAGE_POLICY_ALLOWLIST` (/reference/android/app/admin/PackagePolicy#PACKAGE_POLICY_ALLOWLIST) policy type will limit the credential providers that the user can use to the list of packages in the policy.

A `PackagePolicy#PACKAGE_POLICY_ALLOWLIST_AND_SYSTEM` (/reference/android/app/admin/PackagePolicy#PACKAGE_POLICY_ALLOWLIST_AND_SYSTEM) policy type allows access from the OEM default credential providers and the allowlist of credential providers.

A `PackagePolicy#PACKAGE_POLICY_BLOCKLIST` (/reference/android/app/admin/PackagePolicy#PACKAGE_POLICY_BLOCKLIST) policy type will block the credential providers listed in the policy from being used by the user.

Parameters

`policy`	`PackagePolicy`: the policy to set, setting this value to `null` will allow all packages

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if caller is not a device owner or profile owner of a managed profile

setCrossProfileCalendarPackages

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCrossProfileCalendarPackages (ComponentName (/reference/android/content/Co
                Set (/reference/java/util/Set)<String (/reference/java/lang/String)> packageNames)
```

This method was deprecated in API level 34.

Use **`setCrossProfilePackages(android.content.ComponentName, java.util.Set)`** (/reference/android/app/admin/DevicePolicyManager#setCrossProfilePackages(android.content.ComponentName,%20java.util.Set<java.lang.String>))

.

Allows a set of packages to access cross-profile calendar APIs.

Called by a profile owner of a managed profile.

Calling with a `null` value for the set disables the restriction so that all packages are allowed to access cross-profile calendar APIs. Calling with an empty set disallows all packages from accessing cross-profile calendar APIs. If this method isn't called, no package is allowed to access cross-profile calendar APIs by default.

Parameters

`admin`	`ComponentName`: which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with This value cannot be `null`.
`packageNames`	`Set`: set of packages to be allowlisted This value may be `null`.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a profile owner

See also:

`getCrossProfileCalendarPackages(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getCrossProfileCalendarPackages(android.content.ComponentName))

setCrossProfileCallerIdDisabled

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCrossProfileCallerIdDisabled (ComponentName (/reference/android/content/Co
                boolean disabled)
```

This method was deprecated in API level 34.
starting with `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), use `setManagedProfileCallerIdAccessPolicy(android.app.admin.PackagePolicy)` (/reference/android/app/admin/DevicePolicyManager#setManagedProfileCallerIdAccessPolicy(android.app.admin.PackagePolicy))
instead

Called by a profile owner of a managed profile to set whether caller-Id information from the managed profile will be shown in the parent profile, for incoming calls.

The calling device admin must be a profile owner. If it is not, a security exception will be thrown.

Starting with `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), calling this function is similar to calling `setManagedProfileCallerIdAccessPolicy(android.app.admin.PackagePolicy)` (/reference/android/app/admin/DevicePolicyManager#setManagedProfileCallerIdAccessPolicy(android.app.admin.PackagePolicy)) with a `PackagePolicy#PACKAGE_POLICY_BLOCKLIST` (/reference/android/app/admin/PackagePolicy#PACKAGE_POLICY_BLOCKLIST) policy type when `disabled` is false or a `PackagePolicy#PACKAGE_POLICY_ALLOWLIST` (/reference/android/app/admin/PackagePolicy#PACKAGE_POLICY_ALLOWLIST) policy type when `disabled` is true.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`disabled`	`boolean`: If true caller-Id information in the managed profile is not displayed.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a profile owner.

setCrossProfileContactsSearchDisabled

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCrossProfileContactsSearchDisabled (ComponentName (/reference/android/co
            boolean disabled)
```

This method was deprecated in API level 34.

From `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE) use `setManagedProfileContactsAccessPolicy(android.app.admin.PackagePolicy)` (/reference/android/app/admin/DevicePolicyManager#setManagedProfileContactsAccessPolicy(android.app.admin.PackagePolicy))

Called by a profile owner of a managed profile to set whether contacts search from the managed profile will be shown in the parent profile, for incoming calls.

The calling device admin must be a profile owner. If it is not, a security exception will be thrown. Starting with `Build.VERSION_CODES.UPSIDE_DOWN_CAKE`

(/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), calling this function is similar to calling `setManagedProfileContactsAccessPolicy(android.app.admin.PackagePolicy)` (/reference/android/app/admin/DevicePolicyManager#setManagedProfileContactsAccessPolicy(android.app.admin.PackagePolicy)) with a `PackagePolicy#PACKAGE_POLICY_BLOCKLIST` (/reference/android/app/admin/PackagePolicy#PACKAGE_POLICY_BLOCKLIST) policy type when `disabled` is false or a `PackagePolicy#PACKAGE_POLICY_ALLOWLIST` (/reference/android/app/admin/PackagePolicy#PACKAGE_POLICY_ALLOWLIST) policy type when `disabled` is true.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`disabled`	`boolean`: If true contacts search in the managed profile is not displayed.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a profile owner.

setCrossProfilePackages

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCrossProfilePackages (ComponentName (/reference/android/content/ComponentN
            Set (/reference/java/util/Set)<String (/reference/java/lang/String)> packageNames)
```

Sets the set of admin-allowlisted package names that are allowed to request user consent for cross-profile communication.

Assumes that the caller is a profile owner and is the given `admin`.

Previous calls are overridden by each subsequent call to this method.

Note that other apps may be able to request user consent for cross-profile communication if they have been explicitly allowlisted by the OEM.

When previously-set cross-profile packages are missing from `packageNames`, the app-op for `INTERACT_ACROSS_PROFILES` will be reset for those packages. This will not occur for packages that are allowlisted by the OEM.

Parameters

`admin`	`ComponentName`: the `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with This value cannot be `null`.
`packageNames`	`Set`: the new cross-profile package names This value cannot be `null`.

setDefaultDialerApplication

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setDefaultDialerApplication (String (/reference/java/lang/String) packageName)
```

Must be called by a device owner or a profile owner of an organization-owned managed profile to set the default dialer application for the calling user.

When the profile owner of an organization-owned managed profile calls this method, it sets the default dialer application in the work profile. This is only meaningful when work profile telephony is enabled by `setManagedSubscriptionsPolicy(ManagedSubscriptionsPolicy)` (/reference/android/app/admin/DevicePolicyManager#setManagedSubscriptionsPolicy(android.app.admin.ManagedSubscriptionsPolicy))

.

If the device does not support telephony (`PackageManager#FEATURE_TELEPHONY` (/reference/android/content/pm/PackageManager#FEATURE_TELEPHONY)), calling this method will do nothing.

Parameters

`packageName`	`String`: The name of the package to set as the default dialer application. This value cannot be `null`.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner or a profile owner of an organization-owned managed profile.
`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if the package cannot be set as the default dialer, for example if the package is not installed or does not expose the expected activities or services that a dialer app is required to have.

setDefaultSmsApplication

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setDefaultSmsApplication (ComponentName (/reference/android/content/ComponentName),
                String (/reference/java/lang/String) packageName)
```

Must be called by a device owner or a profile owner of an organization-owned managed profile to set the default SMS application.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance, returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)), where the caller must be the profile owner of an organization-owned managed profile and the package must be a pre-installed system package. If called on the parent instance, then the default SMS application is set on the personal profile.

Starting from Android `Build.VERSION_CODES.UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), the profile owner of an organization-owned managed profile can also call this method directly (not on the parent profile instance) to set the default SMS application in the work profile. This is only meaningful when work profile telephony is enabled by `setManagedSubscriptionsPolicy(ManagedSubscriptionsPolicy)` (/reference/android/app/admin/DevicePolicyManager#setManagedSubscriptionsPolicy(android.app.admin.ManagedSubscriptionsPolicy))

.

Parameters

admin	**ComponentName**: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be **null**.
packageName	**String**: The name of the package to set as the default SMS application. This value cannot be **null**.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device or profile owner or if called on the p owner of an organization-owned managed profile.
IllegalArgumentException (/reference/java/lang/IllegalArgumentException)	if called on the parent profile and the package provided is n
IllegalStateException (/reference/java/lang/IllegalStateException)	while trying to set default sms app on the profile and Manag MANAGED_SUBSCRIPTIONS (/reference/android/app/admin/ManagedSubscriptionsPolicy policy is not set.

setDelegatedScopes

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setDelegatedScopes (ComponentName (/reference/android/content/ComponentName)
                String (/reference/java/lang/String) delegatePackage,
                List (/reference/java/util/List)<String (/reference/java/lang/String)> scopes)
```

Called by a profile owner or device owner to grant access to privileged APIs to another app. Granted APIs are determined by `scopes`, which is a list of the `DELEGATION_*` constants.

A broadcast with the `ACTION_APPLICATION_DELEGATION_SCOPES_CHANGED` (/reference/android/app/admin/DevicePolicyManager#ACTION_APPLICATION_DELEGATION_SCOPES_CHANGED) action will be sent to the `delegatePackage` with its new scopes in an `ArrayList<String>` extra under the `EXTRA_DELEGATION_SCOPES` (/reference/android/app/admin/DevicePolicyManager#EXTRA_DELEGATION_SCOPES) key. The broadcast is sent with the `Intent#FLAG_RECEIVER_REGISTERED_ONLY` (/reference/android/content/Intent#FLAG_RECEIVER_REGISTERED_ONLY) flag.

Delegated scopes are a per-user state. The delegated access is persistent until it is later cleared by calling this method with an empty `scopes` list or uninstalling the `delegatePackage`.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`delegatePackage`	`String`: The package name of the app which will be given access. This value cannot be `null`.
`scopes`	`List`: The groups of privileged APIs whose access should be granted to `delegatedPackage`. This value cannot be `null`.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device or a profile owner.

setDeviceOwnerLockScreenInfo

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setDeviceOwnerLockScreenInfo (ComponentName (/reference/android/content/Compo
                CharSequence (/reference/java/lang/CharSequence) info)
```

Sets the device owner information to be shown on the lock screen.

Device owner information set using this method overrides any owner information manually set by the user and prevents the user from further changing it.

If the device owner information is `null` or empty then the device owner info is cleared and the user owner info is shown on the lock screen if it is set.

If the device owner information contains only whitespaces then the message on the lock screen will be blank and the user will not be allowed to change it.

If the device owner information needs to be localized, it is the responsibility of the `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) to listen to the `Intent#ACTION_LOCALE_CHANGED` (/reference/android/content/Intent#ACTION_LOCALE_CHANGED) broadcast and set a new version of this string accordingly.

May be called by the device owner or the profile owner of an organization-owned device.

Parameters

`admin`	`ComponentName`: The name of the admin component to check. This value cannot be `null`.
`info`	`CharSequence`: Device owner information which will be displayed instead of the user owner info.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not a device owner.

setEndUserSessionMessage

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setEndUserSessionMessage (ComponentName (/reference/android/content/ComponentName) admin,
                CharSequence (/reference/java/lang/CharSequence) endUserSessionMessage)
```

Called by a device owner to specify the user session end message. This may be displayed during a user switch.

The message should be limited to a short statement or it may be truncated.

If the message needs to be localized, it is the responsibility of the `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) to listen to the `Intent#ACTION_LOCALE_CHANGED` (/reference/android/content/Intent#ACTION_LOCALE_CHANGED) broadcast and set a new version of this message accordingly.

Parameters

`admin`	`ComponentName`: which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`endUserSessionMessage`	`CharSequence`: message for ending user session, or `null` to use system default message.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device owner.

setFactoryResetProtectionPolicy

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setFactoryResetProtectionPolicy (ComponentName (/reference/android/content/Co
                FactoryResetProtectionPolicy (/reference/android/app/admin/FactoryResetProte
```

Callable by device owner or profile owner of an organization-owned device, to set a factory reset protection (FRP) policy. When a new policy is set, the system notifies the FRP management agent of a policy change by broadcasting `ACTION_RESET_PROTECTION_POLICY_CHANGED`.

Parameters

`admin`	`ComponentName`: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin This value may be `null`.
`policy`	`FactoryResetProtectionPolicy`: the new FRP policy, or **`null`** to clear the current policy.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device owner or a profile owner of an organization-owned device.

UnsupportedOperationException (/reference/java/lang/UnsupportedOperationException)	if factory reset protection is not supported on the device.

setGlobalPrivateDnsModeOpportunistic

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int setGlobalPrivateDnsModeOpportunistic (ComponentName (/reference/android/conte
```

Sets the global Private DNS mode to opportunistic. May only be called by the device owner.

In this mode, the DNS subsystem will attempt a TLS handshake to the network-supplied resolver prior to attempting name resolution in cleartext.

Note: The device owner won't be able to set the global private DNS mode if there are unaffiliated secondary users or profiles on the device. It's recommended that affiliation ids are set for new users as soon as possible after provisioning via `setAffiliationIds(ComponentName, Set)` (/reference/android/app/admin/DevicePolicyManager#setAffiliationIds(android.content.ComponentName,%20java.util.Set<java.lang.String>)).

Parameters

`admin`	`ComponentName`: which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Returns

`int`	`PRIVATE_DNS_SET_NO_ERROR` if the mode was set successfully, or `PRIVA` `FAILURE_SETTING` if it could not be set. Value is `PRIVATE_DNS_SET_NO_` (/reference/android/app/admin/DevicePolicyManager#PRIVATE_DNS_SET_N `SET_ERROR_HOST_NOT_SERVING` (/reference/android/app/admin/DevicePolicyManager#PRIVATE_DNS_SET_E

, or PRIVATE_DNS_SET_ERROR_FAILURE_SETTING (/reference/android/app/admin/DevicePolicyManager#PRIVATE_DNS_SET_E

Throws

SecurityException (/reference/java/lang/SecurityException)	if the caller is not the device owner.

setGlobalPrivateDnsModeSpecifiedHost

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int setGlobalPrivateDnsModeSpecifiedHost (ComponentName (/reference/android/conte
                String (/reference/java/lang/String) privateDnsHost)
```

Sets the global Private DNS host to be used. May only be called by the device owner.

Note that the method is blocking as it will perform a connectivity check to the resolver, to ensure it is valid. Because of that, the method should not be called on any thread that relates to user interaction, such as the UI thread.

In case a VPN is used in conjunction with Private DNS resolver, the Private DNS resolver must be reachable both from within and outside the VPN. Otherwise, the device may lose the ability to resolve hostnames as system traffic to the resolver may not go through the VPN.

Note: The device owner won't be able to set the global private DNS mode if there are unaffiliated secondary users or profiles on the device. It's recommended that affiliation ids are set for new users as soon as possible after provisioning via setAffiliationIds(ComponentName, Set) (/reference/android/app/admin/DevicePolicyManager#setAffiliationIds(android.content.ComponentName,%20java.util.Set<java.lang.String>)).

This method may take several seconds to complete, so it should only be called from a worker thread.

Parameters

`admin`	`ComponentName`: which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`privateDnsHost`	`String`: The hostname of a server that implements DNS over TLS (RFC7858). This value cannot be `null`.

Returns

`int`	`PRIVATE_DNS_SET_NO_ERROR` if the mode was set successfully, `PRIVATE FAILURE_SETTING` if it could not be set or `PRIVATE_DNS_SET_ERROR_HO` specified host does not implement RFC7858. Value is `PRIVATE_DNS_SET_` (/reference/android/app/admin/DevicePolicyManager#PRIVATE_DNS_SET_ `SET_ERROR_HOST_NOT_SERVING` (/reference/android/app/admin/DevicePolicyManager#PRIVATE_DNS_SET_ , or `PRIVATE_DNS_SET_ERROR_FAILURE_SETTING` (/reference/android/app/admin/DevicePolicyManager#PRIVATE_DNS_SET_

Throws

`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if the `privateDnsHost` is not a valid hostname.
`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not the device owner.

setGlobalSetting

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setGlobalSetting (ComponentName (/reference/android/content/ComponentName) ad
                String (/reference/java/lang/String) setting,
                String (/reference/java/lang/String) value)
```

This method is mostly deprecated. Most of the settings that still have an effect have dedicated setter methods or user restrictions. See individual settings for details.

Called by device owner to update `Settings.Global` (/reference/android/provider/Settings.Global) settings. Validation that the value of the setting is in the correct form for the setting type should be performed by the caller.

The settings that can be updated with this method are:

- `Settings.Global.ADB_ENABLED` (/reference/android/provider/Settings.Global#ADB_ENABLED) : use `UserManager#DISALLOW_DEBUGGING_FEATURES` (/reference/android/os/UserManager#DISALLOW_DEBUGGING_FEATURES) instead to restrict users from enabling debugging features and this setting to turn adb on.
- `Settings.Global.USB_MASS_STORAGE_ENABLED` (/reference/android/provider/Settings.Global#USB_MASS_STORAGE_ENABLED)
- `Settings.Global.STAY_ON_WHILE_PLUGGED_IN` (/reference/android/provider/Settings.Global#STAY_ON_WHILE_PLUGGED_IN) This setting is only available from `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M) onwards and can only be set if `setMaximumTimeToLock(ComponentName, long)` (/reference/android/app/admin/DevicePolicyManager#setMaximumTimeToLock(android.content.ComponentName,%20long)) is not used to set a timeout.
- `Settings.Global.WIFI_DEVICE_OWNER_CONFIGS_LOCKDOWN` (/reference/android/provider/Settings.Global#WIFI_DEVICE_OWNER_CONFIGS_LOCKDOWN)

  This setting is only available from `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M) onwards.

The following settings used to be supported, but can be controlled in other ways:

- `Settings.Global.AUTO_TIME` (/reference/android/provider/Settings.Global#AUTO_TIME) : Use `setAutoTimeEnabled(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setAutoTimeEnabled(android.content.ComponentName,%20boolean))

and `UserManager#DISALLOW_CONFIG_DATE_TIME` (/reference/android/os/UserManager#DISALLOW_CONFIG_DATE_TIME) instead.

- `Settings.Global.AUTO_TIME_ZONE` (/reference/android/provider/Settings.Global#AUTO_TIME_ZONE) : Use `setAutoTimeZoneEnabled(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setAutoTimeZoneEnabled(android.content.ComponentName,%20boolean)) and `UserManager#DISALLOW_CONFIG_DATE_TIME` (/reference/android/os/UserManager#DISALLOW_CONFIG_DATE_TIME) instead.
- `Settings.Global.DATA_ROAMING` (/reference/android/provider/Settings.Global#DATA_ROAMING) : Use `UserManager#DISALLOW_DATA_ROAMING` (/reference/android/os/UserManager#DISALLOW_DATA_ROAMING) instead.

Changing the following settings has no effect as of `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M):

- `Settings.Global.BLUETOOTH_ON` (/reference/android/provider/Settings.Global#BLUETOOTH_ON). Use `BluetoothAdapter.enable()` (/reference/android/bluetooth/BluetoothAdapter#enable()) and `BluetoothAdapter.disable()` (/reference/android/bluetooth/BluetoothAdapter#disable()) instead.
- `Settings.Global.DEVELOPMENT_SETTINGS_ENABLED` (/reference/android/provider/Settings.Global#DEVELOPMENT_SETTINGS_ENABLED)
- `Settings.Global.MODE_RINGER` (/reference/android/provider/Settings.Global#MODE_RINGER). Use `AudioManager.setRingerMode(int)` (/reference/android/media/AudioManager#setRingerMode(int)) instead.
- `Settings.Global.NETWORK_PREFERENCE` (/reference/android/provider/Settings.Global#NETWORK_PREFERENCE)
- `Settings.Global.WIFI_ON` (/reference/android/provider/Settings.Global#WIFI_ON). Use `WifiManager.setWifiEnabled(boolean)` (/reference/android/net/wifi/WifiManager#setWifiEnabled(boolean)) instead.
- `Settings.Global.WIFI_SLEEP_POLICY` (/reference/android/provider/Settings.Global#WIFI_SLEEP_POLICY). No longer has effect.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`setting`	`String`: The name of the setting to update.
`value`	`String`: The value to update the setting to.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device owner.

setKeepUninstalledPackages

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setKeepUninstalledPackages (ComponentName (/reference/android/content/Compone
                List (/reference/java/util/List)<String (/reference/java/lang/String)> packageNames
```

Set a list of apps to keep around as APKs even if no user has currently installed it. This function can be called by a device owner or by a delegate given the `DELEGATION_KEEP_UNINSTALLED_PACKAGES` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_KEEP_UNINSTALLED_PACKAGES) scope via `setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>)).

Please note that setting this policy does not imply that specified apps will be automatically pre-cached.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if the caller is a keep uninstalled packages delegate.
`packageNames`	`List`: List of package names to keep cached. This value cannot be `null`.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device owner.

See also:

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`DELEGATION_KEEP_UNINSTALLED_PACKAGES` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_KEEP_UNINSTALLED_PACKAGES)

setKeyPairCertificate

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean setKeyPairCertificate (ComponentName (/reference/android/content/ComponentName),
                String (/reference/java/lang/String) alias,
                List (/reference/java/util/List)<Certificate (/reference/java/security/cert/Certificate)> certs,
                boolean isUserSelectable)
```

This API can be called by the following to associate certificates with a key pair that was generated using `generateKeyPair(ComponentName, String, KeyGenParameterSpec, int)`

(/reference/android/app/admin/DevicePolicyManager#generateKeyPair(android.content.ComponentName,%20java.lang.String,%20android.security.keystore.KeyGenParameterSpec,%20int))
, and set whether the key is available for the user to choose in the certificate selection prompt:

- Device owner
- Profile owner
- Delegated certificate installer
- Credential management app

From Android `Build.VERSION_CODES.S` (/reference/android/os/Build.VERSION_CODES#S), the credential management app can call this API. If called by the credential management app, the componentName must be `null`. Note, there can only be a credential management app on an unmanaged device.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app.
`alias`	`String`: The private key alias under which to install the certificate. The `ali` This value cannot be `null`.
`certs`	`List`: The certificate chain to install. The chain should start with the leaf ce `getCertificateChain(Context, String)` (/reference/android/securit `null`.
`isUserSelectable`	`boolean`: `true` to indicate that a user can select this key via the certificate `DeviceAdminReceiver.onChoosePrivateKeyAlias(Context, Inte` (/reference/android/app/admin/DeviceAdminReceiver#onChoosePrivateKe .

Returns

`boolean`	`true` if the provided `alias` exists and the certificates has been successfully associated with it, `false` otherwise.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not `null` and not a device or profile owner, or `admin` is null but the calling application is not a delegated certificate installer or credential management app.

setKeyguardDisabled

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean setKeyguardDisabled (ComponentName (/reference/android/content/ComponentNa
                boolean disabled)
```

Called by a device owner or profile owner of secondary users that is affiliated with the device to disable the keyguard altogether.

Setting the keyguard to disabled has the same effect as choosing "None" as the screen lock type. However, this call has no effect if a password, pin or pattern is currently set. If a password, pin or pattern is set after the keyguard was disabled, the keyguard stops being disabled.

As of `Build.VERSION_CODES.P` (/reference/android/os/Build.VERSION_CODES#P), this call also dismisses the keyguard if it is currently shown.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

`disabled`	`boolean`: `true` disables the keyguard, `false` reenables it.

Returns

`boolean`	`false` if attempting to disable the keyguard while a lock password was in place. `true` otherwise.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not the device owner, or a profile owner of secondary user that is affiliated with the device.

See also:

`isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser())

`getSecondaryUsers(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getSecondaryUsers(android.content.ComponentName))

setKeyguardDisabledFeatures

Added in API level 17 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setKeyguardDisabledFeatures (ComponentName (/reference/android/content/Compor
                int which)
```

Called by an application that is administering the device to disable keyguard customizations, such as widgets. After setting this, keyguard features will be disabled according to the provided feature list.

A calling device admin must have requested `DeviceAdminInfo#USES_POLICY_DISABLE_KEYGUARD_FEATURES`

(/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_DISABLE_KEYGUARD_FEATURES) to be able to call this method; if it has not, a security exception will be thrown.

Calling this from a managed profile before version `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M) will throw a security exception. From version `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M) the profile owner of a managed profile can set:

- `KEYGUARD_DISABLE_TRUST_AGENTS` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_TRUST_AGENTS), which affects the parent user, but only if there is no separate challenge set on the managed profile.
- `KEYGUARD_DISABLE_FINGERPRINT` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_FINGERPRINT), `KEYGUARD_DISABLE_FACE` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_FACE) or `KEYGUARD_DISABLE_IRIS` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_IRIS) which affects the managed profile challenge if there is one, or the parent user otherwise.
- `KEYGUARD_DISABLE_UNREDACTED_NOTIFICATIONS` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_UNREDACTED_NOTIFICATIONS) which affects notifications generated by applications in the managed profile.

From version `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) the profile owner of an organization-owned managed profile can set:

- `KEYGUARD_DISABLE_SECURE_CAMERA` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_SECURE_CAMERA) which affects the parent user when called on the parent profile.
- `KEYGUARD_DISABLE_SECURE_NOTIFICATIONS` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_SECURE_NOTIFICATIONS) which affects the parent user when called on the parent profile.

`KEYGUARD_DISABLE_TRUST_AGENTS` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_TRUST_AGENTS), `KEYGUARD_DISABLE_FINGERPRINT` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_FINGERPRINT),

KEYGUARD_DISABLE_FACE (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_FACE), KEYGUARD_DISABLE_IRIS (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_IRIS), KEYGUARD_DISABLE_SECURE_CAMERA (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_SECURE_CAMERA) and KEYGUARD_DISABLE_SECURE_NOTIFICATIONS (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_SECURE_NOTIFICATIONS) can also be set on the DevicePolicyManager (/reference/android/app/admin/DevicePolicyManager) instance returned by getParentProfileInstance(android.content.ComponentName) (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to set restrictions on the parent profile. KEYGUARD_DISABLE_SECURE_CAMERA (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_SECURE_CAMERA) can only be set on the parent profile instance if the calling device admin is the profile owner of an organization-owned managed profile.

Requests to disable other features on a managed profile will be ignored.

The admin can check which features have been disabled by calling getKeyguardDisabledFeatures(android.content.ComponentName) (/reference/android/app/admin/DevicePolicyManager#getKeyguardDisabledFeatures(android.content.ComponentName))

Parameters

admin	ComponentName: Which DeviceAdminReceiver (/reference/android/app request is associated with. Null if the caller is not a device admin This value
which	int: The disabled features flag which can be either KEYGUARD_DISABLE_ (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABL KEYGUARD_DISABLE_FEATURES_ALL (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABL combination of KEYGUARD_DISABLE_WIDGETS_ALL (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABL DISABLE_SECURE_CAMERA (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABL DISABLE_SECURE_NOTIFICATIONS (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABL

KEYGUARD_DISABLE_TRUST_AGENTS (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABL
DISABLE_UNREDACTED_NOTIFICATIONS (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABL
, KEYGUARD_DISABLE_FINGERPRINT (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABL
DISABLE_FACE (/reference/android/app/admin/DevicePolicyManager#KEY
KEYGUARD_DISABLE_IRIS (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABL
SHORTCUTS_ALL (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABL

Throws

SecurityException (/reference/java/lang/SecurityException)	if admin is not an active administrator or does not use DeviceAdmi DISABLE_KEYGUARD_FEATURES (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_DI

setLocationEnabled

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setLocationEnabled (ComponentName (/reference/android/content/ComponentName)
                boolean locationEnabled)
```

Called by device owners to set the user's global location setting.

Parameters

admin	ComponentName: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with This value cannot be null.
locationEnabled	boolean: whether location should be enabled or disabled. Note: on automotive builds

(/reference/android/content/pm/PackageManager#FEATURE_AUTOMOTIVE
, calls to disable will be ignored.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not a device owner.

setLockTaskFeatures

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setLockTaskFeatures (ComponentName (/reference/android/content/ComponentName)
                int flags)
```

Sets which system features are enabled when the device runs in lock task mode. This method doesn't affect the features when lock task mode is inactive. Any system features not included in `flags` are implicitly disabled when calling this method. By default, only `LOCK_TASK_FEATURE_GLOBAL_ACTIONS` (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FEATURE_GLOBAL_ACTIONS) is enabled; all the other features are disabled. To disable the global actions dialog, call this method omitting `LOCK_TASK_FEATURE_GLOBAL_ACTIONS` (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FEATURE_GLOBAL_ACTIONS).

This method can only be called by the device owner, a profile owner of an affiliated user or profile, or the profile owner when no device owner is set or holders of the permission `Manifest.permission.MANAGE_DEVICE_POLICY_LOCK_TASK` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_LOCK_TASK). See `isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser()). Any features set using this method are cleared if the user becomes unaffiliated.

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), after the lock task features policy has been set, `PolicyUpdateReceiver#onPolicySetResult(Context, String, Bundle, TargetUser, PolicyUpdateResult)`

(/reference/android/app/admin/PolicyUpdateReceiver#onPolicySetResult(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult))
will notify the admin on whether the policy was successfully set or not. This callback will contain:

- The policy identifier `DevicePolicyIdentifiers#LOCK_TASK_POLICY` (/reference/android/app/admin/DevicePolicyIdentifiers#LOCK_TASK_POLICY)
- The `TargetUser` (/reference/android/app/admin/TargetUser) that this policy relates to
- The `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult), which will be `PolicyUpdateResult#RESULT_POLICY_SET` (/reference/android/app/admin/PolicyUpdateResult#RESULT_POLICY_SET) if the policy was successfully set or the reason the policy failed to be set (e.g. `PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY` (/reference/android/app/admin/PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY)
)

If there has been a change to the policy, `PolicyUpdateReceiver#onPolicyChanged(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicyChanged(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult))
will notify the admin of this change. This callback will contain the same parameters as PolicyUpdateReceiver#onPolicySetResult and the `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult) will contain the reason why the policy changed.

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), lock task features and lock task packages are bundled as one policy. A failure to apply one will result in a failure to apply the other.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/ request is associated with. Null if the caller is not a device admin. This value

---

`flags`	`int`: The system features enabled during lock task mode. Value is either `0` c FEATURE_NONE (/reference/android/app/admin/DevicePolicyManager#LOC TASK_FEATURE_SYSTEM_INFO (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FEATUI FEATURE_NOTIFICATIONS (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FEATUI FEATURE_HOME (/reference/android/app/admin/DevicePolicyManager#LOC TASK_FEATURE_OVERVIEW (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FEATUI FEATURE_GLOBAL_ACTIONS (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FEATUI FEATURE_KEYGUARD (/reference/android/app/admin/DevicePolicyManager and LOCK_TASK_FEATURE_BLOCK_ACTIVITY_START_IN_TASK (/reference/android/app/admin/DevicePolicyManager#LOCK_TASK_FEATUI

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not the device owner, the profile owner of an affiliated us or the profile owner when no device owner is set or holder of the pe Manifest.permission.MANAGE_DEVICE_POLICY_LOCK_TASK (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC .

See also:

isAffiliatedUser() (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser())

setLockTaskPackages

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setLockTaskPackages (ComponentName (/reference/android/content/ComponentName)
                String[] (/reference/java/lang/String) packages)
```

Sets which packages may enter lock task mode.

Any packages that share uid with an allowed package will also be allowed to activate lock task. From `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M) removing packages from the lock task package list results in locked tasks belonging to those packages to be finished.

This function can only be called by the device owner, a profile owner of an affiliated user or profile, or the profile owner when no device owner is set or holders of the permission `Manifest.permission.MANAGE_DEVICE_POLICY_LOCK_TASK` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_LOCK_TASK). See `isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser()). Any package set via this method will be cleared if the user becomes unaffiliated.

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), after the lock task policy has been set, `PolicyUpdateReceiver#onPolicySetResult(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicySetResult(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin on whether the policy was successfully set or not. This callback will contain:

- The policy identifier `DevicePolicyIdentifiers#LOCK_TASK_POLICY` (/reference/android/app/admin/DevicePolicyIdentifiers#LOCK_TASK_POLICY)
- The `TargetUser` (/reference/android/app/admin/TargetUser) that this policy relates to
- The `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult), which will be `PolicyUpdateResult#RESULT_POLICY_SET` (/reference/android/app/admin/PolicyUpdateResult#RESULT_POLICY_SET) if the policy was successfully set or the reason the policy failed to be set (e.g. `PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY` (/reference/android/app/admin/PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY)))

If there has been a change to the policy, `PolicyUpdateReceiver#onPolicyChanged(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicyChanged(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin of this change. This callback will contain the same parameters as

PolicyUpdateReceiver#onPolicySetResult and the `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult) will contain the reason why the policy changed.

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), lock task features and lock task packages are bundled as one policy. A failure to apply one will result in a failure to apply the other.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.
`packages`	`String`: The list of packages allowed to enter lock task mode This value cannot be `null`.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not the device owner, the profile owner of an affiliated u or the profile owner when no device owner is set or holder of the pe `Manifest.permission.MANAGE_DEVICE_POLICY_LOCK_TASK` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC .

See also:

`isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser())

`Activity.startLockTask()` (/reference/android/app/Activity#startLockTask())

`DeviceAdminReceiver.onLockTaskModeEntering(Context, Intent, String)` (/reference/android/app/admin/DeviceAdminReceiver#onLockTaskModeEntering(android.content.Context,%20android.content.Intent,%20java.lang.String))

`DeviceAdminReceiver.onLockTaskModeExiting(Context, Intent)` (/reference/android/app/admin/DeviceAdminReceiver#onLockTaskModeExiting(android.content.Context,%20android.content.Intent))

`UserManager.DISALLOW_CREATE_WINDOWS` (/reference/android/os/UserManager#DISALLOW_CREATE_WINDOWS)

setLogoutEnabled

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setLogoutEnabled (ComponentName (/reference/android/content/ComponentName) ad
                boolean enabled)
```

Called by a device owner to specify whether logout is enabled for all secondary users. The system may show a logout button that stops the user and switches back to the primary user.

Parameters

`admin`	**`ComponentName`**: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be **`null`**.
`enabled`	**`boolean`**: whether logout should be enabled or not.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if **`admin`** is not a device owner.

setLongSupportMessage

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setLongSupportMessage (ComponentName (/reference/android/content/ComponentNam
                CharSequence (/reference/java/lang/CharSequence) message)
```

Called by a device admin to set the long support message. This will be displayed to the user in the device administrators settings screen. If the message is longer than 20000 characters it may be truncated.

If the long support message needs to be localized, it is the responsibility of the `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) to listen to the `Intent#ACTION_LOCALE_CHANGED` (/reference/android/content/Intent#ACTION_LOCALE_CHANGED) broadcast and set a new version of this string accordingly.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`message`	`CharSequence`: Long message to be displayed to the user in settings or null to clear the existing message.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not an active administrator.

See also:

`setShortSupportMessage(ComponentName, CharSequence)` (/reference/android/app/admin/DevicePolicyManager#setShortSupportMessage(android.content.ComponentName,%20java.lang.CharSequence))

setManagedProfileCallerIdAccessPolicy

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setManagedProfileCallerIdAccessPolicy (PackagePolicy (/reference/android/ap
```

Called by a profile owner of a managed profile to set the packages that are allowed to lookup contacts in the managed profile based on caller id information.

For example, the policy determines if a dialer app in the parent profile resolving an incoming call can search the caller id data, such as phone number, of managed contacts and return managed contacts that match.

The calling device admin must be a profile owner of a managed profile. If it is not, a `SecurityException` (/reference/java/lang/SecurityException) will be thrown.

A `PackagePolicy#PACKAGE_POLICY_ALLOWLIST_AND_SYSTEM` (/reference/android/app/admin/PackagePolicy#PACKAGE_POLICY_ALLOWLIST_AND_SYSTEM) policy type allows access from the OEM default packages for the Sms, Dialer and Contact roles, in addition to the packages specified in `PackagePolicy#getPackageNames()` (/reference/android/app/admin/PackagePolicy#getPackageNames())

Parameters

`policy`	`PackagePolicy`: the policy to set, setting this value to `null` will allow all packages

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if caller is not a profile owner of a managed profile

setManagedProfileContactsAccessPolicy

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setManagedProfileContactsAccessPolicy (PackagePolicy (/reference/android/ap
```

Called by a profile owner of a managed profile to set the packages that are allowed access to the managed profile contacts from the parent user.

For example, the system will enforce the provided policy and determine if contacts in the managed profile are shown when queried by an application in the parent user.

The calling device admin must be a profile owner of a managed profile. If it is not, a `SecurityException` (/reference/java/lang/SecurityException) will be thrown.

A `PackagePolicy#PACKAGE_POLICY_ALLOWLIST_AND_SYSTEM` (/reference/android/app/admin/PackagePolicy#PACKAGE_POLICY_ALLOWLIST_AND_SYSTEM) policy type allows access from the OEM default packages for the Sms, Dialer and Contact roles, in addition to the packages specified in `PackagePolicy#getPackageNames()` (/reference/android/app/admin/PackagePolicy#getPackageNames())

Parameters

`policy`	`PackagePolicy`: the policy to set, setting this value to `null` will allow all packages

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if caller is not a profile owner of a managed profile

setManagedProfileMaximumTimeOff

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setManagedProfileMaximumTimeOff (ComponentName (/reference/android/content/Co
            long timeoutMillis)
```

Called by a profile owner of an organization-owned managed profile to set maximum time the profile is allowed to be turned off. If the profile is turned off for longer, personal apps are suspended on the device.

When personal apps are suspended, an ongoing notification about that is shown to the user. When the user taps the notification, system invokes `ACTION_CHECK_POLICY_COMPLIANCE` (/reference/android/app/admin/DevicePolicyManager#ACTION_CHECK_POLICY_COMPLIANCE) in the profile owner package. Profile owner implementation that uses personal apps suspension must handle this intent.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with This value cannot be `null`.
`timeoutMillis`	`long`: Maximum time the profile is allowed to be off in milliseconds or 0 if not limited. The minimum non-zero value corresponds to 72 hours. If an admin sets a smaller non-zero vaulue, 72 hours will be set instead.

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if the profile owner doesn't have an activity that handles `ACTION_COMPLIANCE` (/reference/android/app/admin/DevicePolicyManager#ACTION_C

See also:

`setPersonalAppsSuspended(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setPersonalAppsSuspended(android.content.ComponentName,%20boolean))

setManagedSubscriptionsPolicy

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setManagedSubscriptionsPolicy (ManagedSubscriptionsPolicy (/reference/andr
```

Called by a profile owner of an organization-owned device to specify `ManagedSubscriptionsPolicy` (/reference/android/app/admin/ManagedSubscriptionsPolicy)

Managed subscriptions policy controls how SIMs would be associated with the managed profile. For example a policy of type `ManagedSubscriptionsPolicy#TYPE_ALL_MANAGED_SUBSCRIPTIONS` (/reference/android/app/admin/ManagedSubscriptionsPolicy#TYPE_ALL_MANAGED_SUBSCRIPTIONS) assigns all SIM-based subscriptions to the managed profile. In this case OEM default dialer and messages app are automatically installed in the managed profile and all incoming and outgoing calls and text messages are handled by them.

This API can only be called during device setup.

Parameters

`policy`	`ManagedSubscriptionsPolicy`: `ManagedSubscriptionsPolicy` (/reference/android/app/admin/ManagedSubscriptionsPolicy) policy, passing null for this resets the policy to be the default.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not a profile owner on an organization-owned managed profile.
`IllegalStateException` (/reference/java/lang/IllegalStateException)	if called after the device setup has been completed.
`UnsupportedOperationException` (/reference/java/lang/UnsupportedOperationException)	if managed subscriptions policy is not explicitly enabled by the device policy management role holder during device setup.

See also:

`ManagedSubscriptionsPolicy` (/reference/android/app/admin/ManagedSubscriptionsPolicy)

setMasterVolumeMuted

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setMasterVolumeMuted (ComponentName (/reference/android/content/ComponentName
                boolean on)
```

Called by profile or device owners to set the global volume mute on or off. This has no effect when set on a managed profile.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`on`	`boolean`: `true` to mute global volume, `false` to turn mute off.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner.

setMaximumFailedPasswordsForWipe

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setMaximumFailedPasswordsForWipe (ComponentName (/reference/android/content/C
                int num)
```

Setting this to a value greater than zero enables a policy that will perform a device or profile wipe after too many incorrect device-unlock passwords have been entered. This policy combines watching for failed passwords and wiping the device, and requires that calling Device Admins request both `DeviceAdminInfo#USES_POLICY_WATCH_LOGIN` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_WATCH_LOGIN) and `DeviceAdminInfo#USES_POLICY_WIPE_DATA` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_WIPE_DATA)}.

When this policy is set on the system or the main user, the device will be factory reset after too many incorrect password attempts. When set on any other user, only the corresponding user or profile will be wiped.

To implement any other policy (e.g. wiping data for a particular application only, erasing or revoking credentials, or reporting the failure to a server), you should implement `DeviceAdminReceiver#onPasswordFailed(Context, android.content.Intent)` (/reference/android/app/admin/DeviceAdminReceiver#onPasswordFailed(android.content.Context,%20android.content.Intent)) instead. Do not use this API, because if the maximum count is reached, the device or profile will be wiped immediately, and your callback will not be invoked.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to set a value on the parent profile. This allows a profile wipe after too many incorrect device-unlock password have been entered on the parent profile even if each profile has a separate challenge.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always empty and this method has no effect - i.e. the policy is not set.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.
`num`	`int`: The number of failed password attempts at which point the device or profile will be wiped.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not null, and `admin` is not an active administrator or doe both `DeviceAdminInfo#USES_POLICY_WATCH_LOGIN` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_W and `DeviceAdminInfo#USES_POLICY_WIPE_DATA` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_WI if `admin` is null and the caller does not have permission to wipe the

setMaximumTimeToLock

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setMaximumTimeToLock (ComponentName (/reference/android/content/ComponentName
                long timeMs)
```

Called by an application that is administering the device to set the maximum time for user activity until the device will lock. This limits the length that the user can set. It takes effect immediately.

A calling device admin must have requested `DeviceAdminInfo#USES_POLICY_FORCE_LOCK` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_FORCE_LOCK) to be able to call this method; if it has not, a security exception will be thrown.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)`

(/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName))

in order to set restrictions on the parent profile.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin This value may be `null`.
`timeMs`	`long`: The new desired maximum time to lock in milliseconds. A value of 0 means there is no restriction.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not an active administrator or it does not use `DeviceAdminInfo#USES_POLICY_FORCE_LOCK` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_FC

setMeteredDataDisabledPackages

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public List (/reference/java/util/List)<String (/reference/java/lang/String)> setMeteredDataDisable
                List (/reference/java/util/List)<String (/reference/java/lang/String)> packageNames
```

Called by a device or profile owner to restrict packages from using metered data.

Parameters

`admin`	`ComponentName`: which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is

	associated with. This value cannot be **null**.
packageNames	**List**: the list of package names to be restricted. This value cannot be **null**.

Returns

List (/reference/java/util/List) <**String** (/reference/java/lang/String)>	a list of package names which could not be restricted. This value cannot be **null**.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device or profile owner.

setMinimumRequiredWifiSecurityLevel

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setMinimumRequiredWifiSecurityLevel (int level)
```

Called by device owner or profile owner of an organization-owned managed profile to specify the minimum security level required for Wi-Fi networks. The device may not connect to networks that do not meet the minimum security level. If the current network does not meet the minimum security level set, it will be disconnected. The following shows the Wi-Fi security levels from the lowest to the highest security level: **WIFI_SECURITY_OPEN** (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_OPEN) **WIFI_SECURITY_PERSONAL** (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_PERSONAL) **WIFI_SECURITY_ENTERPRISE_EAP** (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_ENTERPRISE_EAP)

`WIFI_SECURITY_ENTERPRISE_192` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_ENTERPRISE_192)

Parameters

`level`	`int`: minimum security level Value is `WIFI_SECURITY_OPEN` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_OPE `SECURITY_PERSONAL` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_PER `WIFI_SECURITY_ENTERPRISE_EAP` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_ENT , or `WIFI_SECURITY_ENTERPRISE_192` (/reference/android/app/admin/DevicePolicyManager#WIFI_SECURITY_ENT

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not permitted to set this policy

setMtePolicy

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setMtePolicy (int policy)
```

Called by a device owner, profile owner of an organization-owned device, to set the Memory Tagging Extension (MTE) policy. MTE is a CPU extension that allows to protect against certain classes of security problems at a small runtime performance cost overhead.

The MTE policy can only be set to `MTE_DISABLED` (/reference/android/app/admin/DevicePolicyManager#MTE_DISABLED) if called by a device owner. Otherwise a `SecurityException` (/reference/java/lang/SecurityException) will be thrown.

The device needs to be rebooted to apply changes to the MTE policy.

Parameters

`policy`	`int`: the MTE policy to be set Value is MTE_ENABLED (/reference/android/app/admin/DevicePolicyManager#MTE_ENABLED), MT (/reference/android/app/admin/DevicePolicyManager#MTE_DISABLED), or CONTROLLED_BY_POLICY (/reference/android/app/admin/DevicePolicyManager#MTE_NOT_CONTRO

Throws

SecurityException (/reference/java/lang/SecurityException)	if caller is not permitted to set Mte policy
UnsupportedOperationException (/reference/java/lang/UnsupportedOperationException)	if the device does not support MTE

setNearbyAppStreamingPolicy

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setNearbyAppStreamingPolicy (int policy)
```

Called by a device/profile owner to set nearby app streaming policy. App streaming is when the device starts an app on a virtual display and sends a video stream of the app to nearby devices.

Parameters

`policy`	`int`: One of the `NearbyStreamingPolicy` constants. Value is NEARBY_S POLICY (/reference/android/app/admin/DevicePolicyManager#NEARBY_STREAMIN NEARBY_STREAMING_DISABLED (/reference/android/app/admin/DevicePolicyManager#NEARBY_STREAMIN ENABLED (/reference/android/app/admin/DevicePolicyManager#NEARBY_S

STREAMING_SAME_MANAGED_ACCOUNT_ONLY
(/reference/android/app/admin/DevicePolicyManager#NEARBY_STREAMIN

Throws

SecurityException (/reference/java/lang/SecurityException)	if caller is not a device or profile owner.

setNearbyNotificationStreamingPolicy

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setNearbyNotificationStreamingPolicy (int policy)
```

Called by a device/profile owner to set nearby notification streaming policy. Notification streaming is sending notification data from pre-installed apps to nearby devices.

Parameters

policy	int: One of the NearbyStreamingPolicy constants. Value is NEARBY_S POLICY (/reference/android/app/admin/DevicePolicyManager#NEARBY_STREAMIN NEARBY_STREAMING_DISABLED (/reference/android/app/admin/DevicePolicyManager#NEARBY_STREAMIN ENABLED (/reference/android/app/admin/DevicePolicyManager#NEARBY_S STREAMING_SAME_MANAGED_ACCOUNT_ONLY (/reference/android/app/admin/DevicePolicyManager#NEARBY_STREAMIN

Throws

SecurityException (/reference/java/lang/SecurityException)	if caller is not a device or profile owner

setNetworkLoggingEnabled

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setNetworkLoggingEnabled (ComponentName (/reference/android/content/Component
                boolean enabled)
```

Called by a device owner, profile owner of a managed profile or delegated app with `DELEGATION_NETWORK_LOGGING` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_NETWORK_LOGGING) to control the network logging feature.

Supported for a device owner from Android 8 and a delegated app granted by a device owner from Android 10. Supported for a profile owner of a managed profile and a delegated app granted by a profile owner from Android 12. When network logging is enabled by a profile owner, the network logs will only include work profile network activity, not activity on the personal profile.

Network logs contain DNS lookup and connect() library call events. The following library functions are recorded while network logging is active:

- `getaddrinfo()`
- `gethostbyname()`
- `connect()`

Network logging is a low-overhead tool for forensics but it is not guaranteed to use full system call logging; event reporting is enabled by default for all processes but not strongly enforced. Events from applications using alternative implementations of libc, making direct kernel calls, or deliberately obfuscating traffic may not be recorded.

Some common network events may not be reported. For example:

- Applications may hardcode IP addresses to reduce the number of DNS lookups, or use an alternative system for name resolution, and so avoid calling `getaddrinfo()` or `gethostbyname`.
- Applications may use datagram sockets for performance reasons, for example for a game client. Calling `connect()` is unnecessary for this kind of socket, so it will not trigger a network event.

It is possible to directly intercept layer 3 traffic leaving the device using an always-on VPN service. See `setAlwaysOnVpnPackage(android.content.ComponentName, java.lang.String, boolean)` (/reference/android/app/admin/DevicePolicyManager#setAlwaysOnVpnPackage(android.content.ComponentName,%20java.lang.String,%20boolean)) and `VpnService` (/reference/android/net/VpnService) for details.

Note: The device owner won't be able to retrieve network logs if there are unaffiliated secondary users or profiles on the device, regardless of whether the feature is enabled. Logs will be discarded if the internal buffer fills up while waiting for all users to become affiliated. Therefore it's recommended that affiliation ids are set for new users as soon as possible after provisioning via `setAffiliationIds(ComponentName, Set)` (/reference/android/app/admin/DevicePolicyManager#setAffiliationIds(android.content.ComponentName,%20java.util.Set<java.lang.String>)) .

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if called by a delegated app.
`enabled`	`boolean`: whether network logging should be enabled or not.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device owner or profile owner.

See also:

`setAffiliationIds(ComponentName, Set)` (/reference/android/app/admin/DevicePolicyManager#setAffiliationIds(android.content.ComponentName,%20java.util.Set<java.lang.String>))

`retrieveNetworkLogs(ComponentName, long)` (/reference/android/app/admin/DevicePolicyManager#retrieveNetworkLogs(android.content.ComponentName,%20long))

setOrganizationColor

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setOrganizationColor (ComponentName (/reference/android/content/ComponentName
                int color)
```

This method was deprecated in API level 31.
From `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R), the organization color is never used as the background color of the confirm credentials screen.

Called by a profile owner of a managed profile to set the color used for customization. This color is used as background color of the confirm credentials screen for that user. The default color is teal (#00796B).

The confirm credentials screen can be created using `KeyguardManager.createConfirmDeviceCredentialIntent(CharSequence, CharSequence)` (/reference/android/app/KeyguardManager#createConfirmDeviceCredentialIntent(java.lang.CharSequence,%20java.lang.CharSequence))
.

Starting from Android R, the organization color will no longer be used as the background color of the confirm credentials screen.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`color`	`int`: The 24bit (0xRRGGBB) representation of the color to be used.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a profile owner.

setOrganizationId

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setOrganizationId (String (/reference/java/lang/String) enterpriseId)
```

Sets the Enterprise ID for the work profile or managed device. This is a requirement for generating an enrollment-specific ID for the device, see `getEnrollmentSpecificId()` (/reference/android/app/admin/DevicePolicyManager#getEnrollmentSpecificId()). It is recommended that the Enterprise ID is at least 6 characters long, and no more than 64 characters.

Parameters

`enterpriseId`	`String`: An identifier of the organization this work profile or device is enrolled into. This value cannot be **null**.

setOrganizationName

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setOrganizationName (ComponentName (/reference/android/content/ComponentName)
                CharSequence (/reference/java/lang/CharSequence) title)
```

Called by the device owner (since API 26) or profile owner (since API 24) to set the name of the organization under management.

If the organization name needs to be localized, it is the responsibility of the caller to listen to the `Intent#ACTION_LOCALE_CHANGED` (/reference/android/content/Intent#ACTION_LOCALE_CHANGED) broadcast and set a new version of this string accordingly.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.
`title`	`CharSequence`: The organization name or `null` to clear a previously set name.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner.

setOverrideApnsEnabled

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setOverrideApnsEnabled (ComponentName (/reference/android/content/ComponentNa
                boolean enabled)
```

Called by device owner to set if override APNs should be enabled.

Override APNs are separated from other APNs on the device, and can only be inserted or modified by the device owner. When enabled, only override APNs are in use, any other APNs are ignored.

Note: Enterprise APNs added by managed profile owners do not need to be enabled by this API. They are part of the preferential network service config and is controlled by `setPreferentialNetworkServiceConfigs(List)` (/reference/android/app/admin/DevicePolicyManager#setPreferentialNetworkServiceConfigs(java.util.List<android.app.admin.PreferentialNetworkServiceConfig>)) .

Parameters

`admin`	`ComponentName`: which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with This value cannot be `null`.
`enabled`	`boolean`: `true` if override APNs should be enabled, `false` otherwise

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device owner.

setPackagesSuspended

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public String[] (/reference/java/lang/String) setPackagesSuspended (ComponentName (/reference
                String[] (/reference/java/lang/String) packageNames,
                boolean suspended)
```

Called by device or profile owners to suspend packages for this user. This function can be called by a device owner, profile owner, or by a delegate given the `DELEGATION_PACKAGE_ACCESS` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PACKAGE_ACCESS) scope via `setDelegatedScopes(ComponentName, String, List)`

(/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

.

A suspended package will not be able to start activities. Its notifications will be hidden, it will not show up in recents, will not be able to show toasts or dialogs or ring the device.

The package must already be installed. If the package is uninstalled while suspended the package will no longer be suspended. The admin can block this by using `setUninstallBlocked(ComponentName, String, boolean)` (/reference/android/app/admin/DevicePolicyManager#setUninstallBlocked(android.content.ComponentName,%20java.lang.String,%20boolean))

.

Some apps cannot be suspended, such as device admins, the active launcher, the required package installer, the required package uninstaller, the required package verifier, the default dialer, and the permission controller.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.
`packageNames`	`String`: The package names to suspend or unsuspend. This value cannot be `null`.
`suspended`	`boolean`: If set to `true` than the packages will be suspended, if set to `false` the packages will be unsuspended.

Returns

`String[]` (/reference/java/lang/String)	an array of package names for which the suspended status is not set as requested in this method. This value cannot be `null`.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device or profile owner.

See also:

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`DELEGATION_PACKAGE_ACCESS` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PACKAGE_ACCESS)

setPasswordExpirationTimeout

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPasswordExpirationTimeout (ComponentName (/reference/android/content/Compo
                long timeout)
```

Called by a device admin to set the password expiration timeout. Calling this method will restart the countdown for password expiration for the given admin, as will changing the device password (for all admins).

The provided timeout is the time delta in ms and will be added to the current time. For example, to have the password expire 5 days from now, timeout would be 5 * 86400 * 1000 = 432000000 ms for timeout.

To disable password expiration, a value of 0 may be used for timeout.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password expiration is always disabled.

A calling device admin must have requested `DeviceAdminInfo#USES_POLICY_EXPIRE_PASSWORD`

(/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_EXPIRE_PASSWORD) to be able to call this method; if it has not, a security exception will be thrown.

Note that setting the password will automatically reset the expiration time for all active admins. Active admins do not need to explicitly call this method in that case.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to set restrictions on the parent profile.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin This value may be `null`.
`timeout`	`long`: The limit (in ms) that a password can remain in effect. A value of 0 means there is no restriction (unlimited).

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not an active administrator or `admin` does not use `DeviceAdminInfo#USES_POLICY_EXPIRE_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_EX

setPasswordHistoryLength

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPasswordHistoryLength (ComponentName (/reference/android/content/Component
                int length)
```

Called by an application that is administering the device to set the length of the password history. After setting this, the user will not be able to enter a new password that is the same as any password in the history. Note that the current password will remain until the user has set a new one, so the change does not take place immediately. To prompt the user for a new password, use `ACTION_SET_NEW_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PASSWORD) or `ACTION_SET_NEW_PARENT_PROFILE_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PARENT_PROFILE_PASSWORD) after setting this value.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password history length is always 0.

The calling device admin must have requested `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD) to be able to call this method; if it has not, a security exception will be thrown.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to set restrictions on the parent profile.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`length`	`int`: The new desired length of password history. A value of 0 means there is no restriction.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not an active administrator or `admin` does not use `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_LI

setPasswordMinimumLength

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPasswordMinimumLength (ComponentName (/reference/android/content/Component
                int length)
```

This method was deprecated in API level 31.

see `setPasswordQuality(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) for details.

Called by an application that is administering the device to set the minimum allowed password length. After setting this, the user will not be able to enter a new password that is not at least as restrictive as what has been set. Note that the current password will remain until the user has set a new one, so the change does not take place immediately. To prompt the user for a new password, use `ACTION_SET_NEW_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PASSWORD) or

`ACTION_SET_NEW_PARENT_PROFILE_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PARENT_PROFILE_PASSWORD) after setting this value. This constraint is only imposed if the administrator has also requested either `PASSWORD_QUALITY_NUMERIC` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_NUMERIC) , `PASSWORD_QUALITY_NUMERIC_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_NUMERIC_COMPLEX), `PASSWORD_QUALITY_ALPHABETIC` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_ALPHABETIC), `PASSWORD_QUALITY_ALPHANUMERIC` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_ALPHANUMERIC), or `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX) with `setPasswordQuality(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) . If an app targeting SDK level `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) and above enforces this constraint without settings password quality to one of these values first, this method will throw `IllegalStateException` (/reference/java/lang/IllegalStateException).

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty.

The calling device admin must have requested `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD) to be able to call this method; if it has not, a security exception will be thrown.

Apps targeting `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) and below can call this method on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to set restrictions on the parent profile.

Note: this method is ignored on {PackageManager#FEATURE_AUTOMOTIVE automotive builds}.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`length`	`int`: The new desired minimum password length. A value of 0 means there is no restriction.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not an active administrator or `admin` does not use `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_
`IllegalStateException` (/reference/java/lang/IllegalStateException)	if the calling app is targeting SDK level `Build.VERSION_CODES.` (/reference/android/os/Build.VERSION_CODES#R) and above and sufficient password quality requirement prior to calling this metho

setPasswordMinimumLetters

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPasswordMinimumLetters (ComponentName (/reference/android/content/Componer
                int length)
```

This method was deprecated in API level 31.

see `setPasswordQuality(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int))

for details.

Called by an application that is administering the device to set the minimum number of letters required in the password. After setting this, the user will not be able to enter a new password that is not at least as restrictive as what has been set. Note that the current password will remain until the user has set a new one, so the change does not take place immediately. To prompt the user for a new password, use `ACTION_SET_NEW_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PASSWORD) or `ACTION_SET_NEW_PARENT_PROFILE_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PARENT_PROFILE_PASSWORD) after setting this value. This constraint is only imposed if the administrator has also requested `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX) with `setPasswordQuality(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) . If an app targeting SDK level `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) and above enforces this constraint without settings password quality to `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX) first, this method will throw `IllegalStateException` (/reference/java/lang/IllegalStateException). The default value is 1.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty.

The calling device admin must have requested `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD) to be able to call this method; if it has not, a security exception will be thrown.

Apps targeting `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) and below can call this method on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to set restrictions on the parent profile.

Note: this method is ignored on {PackageManager#FEATURE_AUTOMOTIVE automotive builds}.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`length`	`int`: The new desired minimum number of letters required in the password. A value of 0 means there is no restriction.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not an active administrator or `admin` does not use `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_
`IllegalStateException` (/reference/java/lang/IllegalStateException)	if the calling app is targeting SDK level `Build.VERSION_CODES.` (/reference/android/os/Build.VERSION_CODES#R) and above and sufficient password quality requirement prior to calling this metho

setPasswordMinimumLowerCase

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPasswordMinimumLowerCase (ComponentName (/reference/android/content/Compor
                int length)
```

This method was deprecated in API level 31.

see `setPasswordQuality(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int))

for details.

Called by an application that is administering the device to set the minimum number of lower case letters required in the password. After setting this, the user will not be able to enter a new password that is not at least as restrictive as what has been set. Note that the current password will remain until the user has set a new one, so the change does not take place immediately. To prompt the user for a new password, use `ACTION_SET_NEW_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PASSWORD) or `ACTION_SET_NEW_PARENT_PROFILE_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PARENT_PROFILE_PASSWORD) after setting this value. This constraint is only imposed if the administrator has also requested `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX) with `setPasswordQuality(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) . If an app targeting SDK level `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) and above enforces this constraint without settings password quality to `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX) first, this method will throw `IllegalStateException` (/reference/java/lang/IllegalStateException). The default value is 0.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty.

The calling device admin must have requested `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD) to be able to call this method; if it has not, a security exception will be thrown.

Apps targeting `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) and below can call this method on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to set restrictions on the parent profile.

Note: this method is ignored on {PackageManager#FEATURE_AUTOMOTIVE automotive builds}.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`length`	`int`: The new desired minimum number of lower case letters required in the password. A value of 0 means there is no restriction.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not an active administrator or `admin` does not use `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_
`IllegalStateException` (/reference/java/lang/IllegalStateException)	if the calling app is targeting SDK level `Build.VERSION_CODES.` (/reference/android/os/Build.VERSION_CODES#R) and above and sufficient password quality requirement prior to calling this metho

setPasswordMinimumNonLetter

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPasswordMinimumNonLetter (ComponentName (/reference/android/content/Compor
                int length)
```

This method was deprecated in API level 31.

see **`setPasswordQuality(android.content.ComponentName, int)`** (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int))

for details.

Called by an application that is administering the device to set the minimum number of non-letter characters (numerical digits or symbols) required in the password. After setting this, the user will not be able to enter a new password that is not at least as restrictive as what has been set. Note that the current password will remain until the user has set a new one, so the change does not take place immediately. To prompt the user for a new password, use `ACTION_SET_NEW_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PASSWORD) or `ACTION_SET_NEW_PARENT_PROFILE_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PARENT_PROFILE_PASSWORD) after setting this value. This constraint is only imposed if the administrator has also requested `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX) with `setPasswordQuality(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) . If an app targeting SDK level `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) and above enforces this constraint without settings password quality to `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX) first, this method will throw `IllegalStateException` (/reference/java/lang/IllegalStateException). The default value is 0.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty.

The calling device admin must have requested `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD) to be able to call this method; if it has not, a security exception will be thrown.

Apps targeting `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) and below can call this method on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to set restrictions on the parent profile.

Note: this method is ignored on {PackageManager#FEATURE_AUTOMOTIVE automotive builds}.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`length`	`int`: The new desired minimum number of letters required in the password. A value of 0 means there is no restriction.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not an active administrator or `admin` does not use `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_
`IllegalStateException` (/reference/java/lang/IllegalStateException)	if the calling app is targeting SDK level `Build.VERSION_CODES.` (/reference/android/os/Build.VERSION_CODES#R) and above and sufficient password quality requirement prior to calling this meth

setPasswordMinimumNumeric

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPasswordMinimumNumeric (ComponentName (/reference/android/content/Componer
                int length)
```

This method was deprecated in API level 31.

see `setPasswordQuality(android.content.ComponentName, int)`

(/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int))
for details.

Called by an application that is administering the device to set the minimum number of numerical digits required in the password. After setting this, the user will not be able to enter a new password that is not at least as restrictive as what has been set. Note that the current password will remain until the user has set a new one, so the change does not take place immediately. To prompt the user for a new password, use `ACTION_SET_NEW_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PASSWORD) or `ACTION_SET_NEW_PARENT_PROFILE_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PARENT_PROFILE_PASSWORD) after setting this value. This constraint is only imposed if the administrator has also requested `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX) with `setPasswordQuality(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)). If an app targeting SDK level `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) and above enforces this constraint without settings password quality to `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX) first, this method will throw `IllegalStateException` (/reference/java/lang/IllegalStateException). The default value is 1.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty.

The calling device admin must have requested `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD) to be able to call this method; if it has not, a security exception will be thrown.

Apps targeting `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) and below can call this method on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)`

(/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName))

in order to set restrictions on the parent profile.

Note: this method is ignored on {PackageManager#FEATURE_AUTOMOTIVE automotive builds}.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`length`	`int`: The new desired minimum number of numerical digits required in the password. A value of 0 means there is no restriction.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not an active administrator or `admin` does not use `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_
`IllegalStateException` (/reference/java/lang/IllegalStateException)	if the calling app is targeting SDK level `Build.VERSION_CODES.` (/reference/android/os/Build.VERSION_CODES#R) and above and sufficient password quality requirement prior to calling this meth

setPasswordMinimumSymbols

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPasswordMinimumSymbols (ComponentName (/reference/android/content/Componer
                int length)
```

This method was deprecated in API level 31.

see `setPasswordQuality(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int))
for details.

Called by an application that is administering the device to set the minimum number of symbols required in the password. After setting this, the user will not be able to enter a new password that is not at least as restrictive as what has been set. Note that the current password will remain until the user has set a new one, so the change does not take place immediately. To prompt the user for a new password, use `ACTION_SET_NEW_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PASSWORD) or `ACTION_SET_NEW_PARENT_PROFILE_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PARENT_PROFILE_PASSWORD) after setting this value. This constraint is only imposed if the administrator has also requested `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX) with `setPasswordQuality(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) . If an app targeting SDK level `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) and above enforces this constraint without settings password quality to `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX) first, this method will throw `IllegalStateException` (/reference/java/lang/IllegalStateException). The default value is 1.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty.

The calling device admin must have requested `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD) to be able to call this method; if it has not, a security exception will be thrown.

Apps targeting `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) and below can call this method on the `DevicePolicyManager`

(/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to set restrictions on the parent profile.

Note: this method is ignored on {PackageManager#FEATURE_AUTOMOTIVE automotive builds}.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`length`	`int`: The new desired minimum number of symbols required in the password. A value of 0 means there is no restriction.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not an active administrator or `admin` does not use `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_
`IllegalStateException` (/reference/java/lang/IllegalStateException)	if the calling app is targeting SDK level `Build.VERSION_CODES.` (/reference/android/os/Build.VERSION_CODES#R) and above and sufficient password quality requirement prior to calling this metho

setPasswordMinimumUpperCase

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPasswordMinimumUpperCase (ComponentName (/reference/android/content/Compor
                int length)
```

This method was deprecated in API level 31.

see `setPasswordQuality(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int))
for details.

Called by an application that is administering the device to set the minimum number of upper case letters required in the password. After setting this, the user will not be able to enter a new password that is not at least as restrictive as what has been set. Note that the current password will remain until the user has set a new one, so the change does not take place immediately. To prompt the user for a new password, use `ACTION_SET_NEW_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PASSWORD) or `ACTION_SET_NEW_PARENT_PROFILE_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PARENT_PROFILE_PASSWORD) after setting this value. This constraint is only imposed if the administrator has also requested `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX) with `setPasswordQuality(ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) . If an app targeting SDK level `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) and above enforces this constraint without settings password quality to `PASSWORD_QUALITY_COMPLEX` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_COMPLEX) first, this method will throw `IllegalStateException` (/reference/java/lang/IllegalStateException). The default value is 0.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty.

The calling device admin must have requested `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD) to be able to call this method; if it has not, a security exception will be thrown.

Apps targeting `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) and below can call this method on the `DevicePolicyManager`

(/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to set restrictions on the parent profile.

Note: this method is ignored on {PackageManager#FEATURE_AUTOMOTIVE automotive builds}.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`length`	`int`: The new desired minimum number of upper case letters required in the password. A value of 0 means there is no restriction.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not an active administrator or `admin` does not use `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_
`IllegalStateException` (/reference/java/lang/IllegalStateException)	if the calling app is targeting SDK level `Build.VERSION_CODES.` (/reference/android/os/Build.VERSION_CODES#R) and above and sufficient password quality requirement prior to calling this metho

setPasswordQuality

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

Deprecated in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPasswordQuality (ComponentName (/reference/android/content/ComponentName)
                int quality)
```

This method was deprecated in API level 31.
Prefer using **`setRequiredPasswordComplexity(int)`** (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswordComplexity(int)), to require a password that satisfies a complexity level defined by the platform, rather than specifying custom password requirement. Setting custom, overly-complicated password requirements leads to passwords that are hard for users to remember and may not provide any security benefits given as Android uses hardware-backed throttling to thwart online and offline brute-forcing of the device's screen lock. Company-owned devices (fully-managed and organization-owned managed profile devices) are able to continue using this method, though it is recommended that **`setRequiredPasswordComplexity(int)`** (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswordComplexity(int)) should be used instead.

Called by an application that is administering the device to set the password restrictions it is imposing. After setting this, the user will not be able to enter a new password that is not at least as restrictive as what has been set. Note that the current password will remain until the user has set a new one, so the change does not take place immediately. To prompt the user for a new password, use `ACTION_SET_NEW_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PASSWORD) or `ACTION_SET_NEW_PARENT_PROFILE_PASSWORD` (/reference/android/app/admin/DevicePolicyManager#ACTION_SET_NEW_PARENT_PROFILE_PASSWORD) after calling this method.

Quality constants are ordered so that higher values are more restrictive; thus the highest requested quality constant (between the policy set here, the user's preference, and any other considerations) is the one that is in effect.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the password is always treated as empty.

The calling device admin must have requested `DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_LIMIT_PASSWORD) to be able to call this method; if it has not, a security exception will be thrown.

Apps targeting `Build.VERSION_CODES.R` (/reference/android/os/Build.VERSION_CODES#R) and below can call this method on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by

`getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to set restrictions on the parent profile. Apps targeting `Build.VERSION_CODES.S` (/reference/android/os/Build.VERSION_CODES#S) and above, with the exception of a profile owner on an organization-owned device (as can be identified by `isOrganizationOwnedDeviceWithManagedProfile()` (/reference/android/app/admin/DevicePolicyManager#isOrganizationOwnedDeviceWithManagedProfile())), will get a `IllegalArgumentException` when calling this method on the parent `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance.

Note: Specifying password requirements using this method clears the password complexity requirements set using `setRequiredPasswordComplexity(int)` (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswordComplexity(int)). If this method is called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)), then password complexity requirements set on the primary `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) must be cleared first by calling `setRequiredPasswordComplexity(int)` (/reference/android/app/admin/DevicePolicyManager#setRequiredPasswordComplexity(int)) with {@link #PASSWORD_COMPLEXITY_NONE) first.

Note: this method is ignored on {PackageManager#FEATURE_AUTOMOTIVE automotive builds}.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is assoc
`quality`	`int`: The new desired quality. One of `PASSWORD_QUALITY_UNSPECIFIED` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALIT `PASSWORD_QUALITY_BIOMETRIC_WEAK` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALIT `PASSWORD_QUALITY_SOMETHING` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALIT `PASSWORD_QUALITY_NUMERIC`

(/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALIT
PASSWORD_QUALITY_NUMERIC_COMPLEX
(/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALIT
, PASSWORD_QUALITY_ALPHABETIC
(/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALIT
PASSWORD_QUALITY_ALPHANUMERIC
(/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALIT
PASSWORD_QUALITY_COMPLEX
(/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALIT

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not an active administrator, if **admin** does not use De PASSWORD (/reference/android/app/admin/DeviceAdminInfo#USE is targeting Build.VERSION_CODES.S (/reference/android/os/B the method the DevicePolicyManager (/reference/android/ap by getParentProfileInstance(ComponentName) (/reference/android/app/admin/DevicePolicyManager#getParent .
IllegalStateException (/reference/java/lang/IllegalStateException)	if the caller is trying to set password quality on the parent Devic (/reference/android/app/admin/DevicePolicyManager) instance v DevicePolicyManager (/reference/android/app/admin/DeviceP

setPermissionGrantState

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean setPermissionGrantState (ComponentName (/reference/android/content/Compone
                String (/reference/java/lang/String) packageName,
                String (/reference/java/lang/String) permission,
                int grantState)
```

Sets the grant state of a runtime permission for a specific application. The state can be `default` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRANT_STATE_DEFAULT) in which a user can manage it through the UI, `denied` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRANT_STATE_DENIED), in which the

permission is denied and the user cannot manage it through the UI, and `granted` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRANT_STATE_GRANTED) in which the permission is granted and the user cannot manage it through the UI. This method can only be called by a profile owner, device owner, or a delegate given the `DELEGATION_PERMISSION_GRANT` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PERMISSION_GRANT) scope via `setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))
.

Note that user cannot manage other permissions in the affected group through the UI either and their granted state will be kept as the current value. Thus, it's recommended that you set the grant state of all the permissions in the affected group.

Setting the grant state to `default` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRANT_STATE_DEFAULT) does not revoke the permission. It retains the previous grant, if any.

Device admins with a `targetSdkVersion` < `Build.VERSION_CODES.Q` (/reference/android/os/Build.VERSION_CODES#Q) cannot grant and revoke permissions for applications built with a `targetSdkVersion` < `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M).

Admins with a `targetSdkVersion` ≥ `Build.VERSION_CODES.Q` (/reference/android/os/Build.VERSION_CODES#Q) can grant and revoke permissions of all apps. Similar to the user revoking a permission from a application built with a `targetSdkVersion` < `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M) the app-op matching the permission is set to `AppOpsManager.MODE_IGNORED` (/reference/android/app/AppOpsManager#MODE_IGNORED), but the permission stays granted.

NOTE: On devices running `Build.VERSION_CODES.S` (/reference/android/os/Build.VERSION_CODES#S) and above, control over the following, sensors-related, permissions is restricted:

- Manifest.permission.ACCESS_FINE_LOCATION
- Manifest.permission.ACCESS_BACKGROUND_LOCATION
- Manifest.permission.ACCESS_COARSE_LOCATION
- Manifest.permission.CAMERA

- Manifest.permission.RECORD_AUDIO
- Manifest.permission.RECORD_BACKGROUND_AUDIO
- Manifest.permission.ACTIVITY_RECOGNITION
- Manifest.permission.BODY_SENSORS

A profile owner may not grant these permissions (i.e. call this method with any of the permissions listed above and `grantState` of `#PERMISSION_GRANT_STATE_GRANTED`), but may deny them.

A device owner, by default, may continue granting these permissions. However, for increased user control, the admin may opt out of controlling grants for these permissions by including `EXTRA_PROVISIONING_SENSORS_PERMISSION_GRANT_OPT_OUT` (/reference/android/app/admin/DevicePolicyManager#EXTRA_PROVISIONING_SENSORS_PERMISSION_GRANT_OPT_OUT) in the provisioning parameters. In that case the device owner's control will be limited to denying these permissions.

NOTE: On devices running `Build.VERSION_CODES.S` (/reference/android/os/Build.VERSION_CODES#S) and above, control over the following permissions are restricted for managed profile owners:

- Manifest.permission.READ_SMS

A managed profile owner may not grant these permissions (i.e. call this method with any of the permissions listed above and `grantState` of `#PERMISSION_GRANT_STATE_GRANTED`), but may deny them.

Attempts by the admin to grant these permissions, when the admin is restricted from doing so, will be silently ignored (no exception will be thrown). Control over the following permissions are restricted for managed profile owners:

- Manifest.permission.READ_SMS

A managed profile owner may not grant these permissions (i.e. call this method with any of the permissions listed above and `grantState` of `#PERMISSION_GRANT_STATE_GRANTED`), but may deny them.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is assoc the caller is not a device admin. This value may be `null`.
`packageName`	`String`: The application to grant or revoke a permission to. This value cann
`permission`	`String`: The permission to grant or revoke. This value cannot be `null`.
`grantState`	`int`: The permission grant state which is one of `PERMISSION_GRANT_STA` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRAN `PERMISSION_GRANT_STATE_DEFAULT` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRAN `PERMISSION_GRANT_STATE_GRANTED` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRAN , Value is `PERMISSION_GRANT_STATE_DEFAULT` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRAN `PERMISSION_GRANT_STATE_GRANTED` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRAN , or `PERMISSION_GRANT_STATE_DENIED` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRAN

Returns

`boolean`	whether the permission was successfully granted or revoked.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner.

See also:

`PERMISSION_GRANT_STATE_DENIED` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRANT_STATE_DENIED)

`PERMISSION_GRANT_STATE_DEFAULT` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRANT_STATE_DEFAULT)

`PERMISSION_GRANT_STATE_GRANTED` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_GRANT_STATE_GRANTED)

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`DELEGATION_PERMISSION_GRANT` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PERMISSION_GRANT)

setPermissionPolicy

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPermissionPolicy (ComponentName (/reference/android/content/ComponentName)
                int policy)
```

Set the default response for future runtime permission requests by applications. This function can be called by a device owner, profile owner, or by a delegate given the `DELEGATION_PERMISSION_GRANT` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PERMISSION_GRANT) scope via `setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>)). The policy can allow for normal operation which prompts the user to grant a permission, or can allow automatic granting or denying of runtime permission requests by an application. This also applies to new permissions declared by app updates. When a permission is denied or granted this way, the effect is equivalent to setting the permission * grant state via `setPermissionGrantState(ComponentName, String, String, int)` (/reference/android/app/admin/DevicePolicyManager#setPermissionGrantState(android.content.ComponentName,%20java.lang.String,%20java.lang.String,%20int)).

As this policy only acts on runtime permission requests, it only applies to applications built with a `targetSdkVersion` of `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M)

or later.

NOTE: On devices running `Build.VERSION_CODES.S` (/reference/android/os/Build.VERSION_CODES#S) and above, an auto-grant policy will not apply to certain sensors-related permissions on some configurations. See `setPermissionGrantState(android.content.ComponentName, java.lang.String, java.lang.String, int)` (/reference/android/app/admin/DevicePolicyManager#setPermissionGrantState(android.content.ComponentName,%20java.lang.String,%20java.lang.String,%20int)) for the list of permissions affected, and the behavior change for managed profiles and fully-managed devices.

Parameters

`admin`	**`ComponentName`**: Which profile or device owner this request is associated cannot be **`null`**.
`policy`	**`int`**: One of the policy constants `PERMISSION_POLICY_PROMPT` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_POLIC `PERMISSION_POLICY_AUTO_GRANT` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_POLIC and `PERMISSION_POLICY_AUTO_DENY` (/reference/android/app/admin/DevicePolicyManager#PERMISSION_POLIC

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if **`admin`** is not a device or profile owner.

See also:

`setPermissionGrantState(ComponentName, String, String, int)` (/reference/android/app/admin/DevicePolicyManager#setPermissionGrantState(android.content.ComponentName,%20java.lang.String,%20java.lang.String,%20int))

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`DELEGATION_PERMISSION_GRANT` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_PERMISSION_GRANT)

setPermittedAccessibilityServices

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean setPermittedAccessibilityServices (ComponentName (/reference/android/cont
                List (/reference/java/util/List)<String (/reference/java/lang/String)> packageNames
```

Called by a profile or device owner to set the permitted `AccessibilityService` (/reference/android/accessibilityservice/AccessibilityService). When set by a device owner or profile owner the restriction applies to all profiles of the user the device owner or profile owner is an admin for. By default, the user can use any accessibility service. When zero or more packages have been added, accessibility services that are not in the list and not part of the system can not be enabled by the user.

Calling with a `null` value for the list disables the restriction so that all services can be used, calling with an empty list only allows the built-in system services. Any non-system accessibility service that's currently enabled must be included in the list.

System accessibility services are always available to the user and this method can't disable them.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`packageNames`	`List`: List of accessibility service package names.

Returns

`boolean`	`true` if the operation succeeded, or `false` if the list didn't contain every enabled non-system accessibility service.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner.

setPermittedCrossProfileNotificationListeners

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean setPermittedCrossProfileNotificationListeners (ComponentName (/referer
                List (/reference/java/util/List)<String (/reference/java/lang/String)> packageList)
```

Called by a profile owner of a managed profile to set the packages that are allowed to use a `NotificationListenerService` (/reference/android/service/notification/NotificationListenerService) in the primary user to see notifications from the managed profile. By default all packages are permitted by this policy. When zero or more packages have been added, notification listeners installed on the primary user that are not in the list and are not part of the system won't receive events for managed profile notifications.

Calling with a `null` value for the list disables the restriction so that all notification listener services be used. Calling with an empty list disables all but the system's own notification listeners. System notification listener services are always available to the user.

If a device or profile owner want to stop notification listeners in their user from seeing that user's notifications they should prevent that service from running instead (e.g. via `setApplicationHidden(android.content.ComponentName, java.lang.String, boolean)` (/reference/android/app/admin/DevicePolicyManager#setApplicationHidden(android.content.ComponentName,%20java.lang.String,%20boolean))
)

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`packageList`	`List`: List of package names to allowlist This value may be `null`.

Returns

`boolean`	true if setting the restriction succeeded. It will fail if called outside a managed profile

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a profile owner.

See also:

`NotificationListenerService` (/reference/android/service/notification/NotificationListenerService)

setPermittedInputMethods

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean setPermittedInputMethods (ComponentName (/reference/android/content/Compor
                List (/reference/java/util/List)<String (/reference/java/lang/String)> packageNames
```

Called by a profile or device owner or holder of the `Manifest.permission.MANAGE_DEVICE_POLICY_INPUT_METHODS` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_INPUT_METHODS) permission to

set the permitted input methods services for this user. By default, the user can use any input method.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance, returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) , where the caller must be a profile owner of an organization-owned device.

If called on the parent instance:

- The permitted input methods will be applied on the personal profile
- Can only permit all input methods (calling this method with a `null` package list) or only permit system input methods (calling this method with an empty package list). This is to prevent the caller from learning which packages are installed on the personal side

When zero or more packages have been added, input method that are not in the list and not part of the system can not be enabled by the user. This method will fail if it is called for a admin that is not for the foreground user or a profile of the foreground user. Any non-system input method service that's currently enabled must be included in the list.

Calling with a null value for the list disables the restriction so that all input methods can be used, calling with an empty list disables all but the system's own input methods.

System input methods are always available to the user - this method can't modify this.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin This value may be `null`.
`packageNames`	`List`: List of input method package names.

Returns

`boolean`	`true` if the operation succeeded, or `false` if the list didn't contain every enabled non-system input method service.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner and does not hold t permission.MANAGE_DEVICE_POLICY_INPUT_METHODS (/reference/android/Manifest.permission#MANAGE_DEVICE permission, or if called on the parent profile and the `admin` i organization-owned managed profile.
IllegalArgumentException (/reference/java/lang/IllegalArgumentException)	if called on the parent profile, the `admin` is a profile owner o managed profile and the list of permitted input method pack empty.

setPersonalAppsSuspended

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPersonalAppsSuspended (ComponentName (/reference/android/content/Component
                boolean suspended)
```

Called by a profile owner of an organization-owned managed profile to suspend personal apps on the device. When personal apps are suspended the device can only be used for calls.

Parameters

`admin`	`ComponentName`: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with This value cannot be `null`.

`suspended`	**`boolean`**: Whether personal apps should be suspended.

Throws

`IllegalStateException` (/reference/java/lang/IllegalStateException)	if the profile owner doesn't have an activity that handles `ACTION_COMPLIANCE` (/reference/android/app/admin/DevicePolicyManager#ACTION_C

setPreferentialNetworkServiceConfigs

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPreferentialNetworkServiceConfigs (List (/reference/java/util/List)<Preferer
```

Sets preferential network configurations. An example of a supported preferential network service is the Enterprise slice on 5G networks. For devices on 4G networks, the profile owner needs to additionally configure enterprise APN to set up data call for the preferential network service. These APNs can be added using `addOverrideApn(ComponentName, ApnSetting)` (/reference/android/app/admin/DevicePolicyManager#addOverrideApn(android.content.ComponentName, %20android.telephony.data.ApnSetting)) . By default, preferential network service is disabled on the work profile and fully managed devices, on supported carriers and devices. Admins can explicitly enable it with this API. If admin wants to have multiple enterprise slices, it can be configured by passing list of `PreferentialNetworkServiceConfig` (/reference/android/app/admin/PreferentialNetworkServiceConfig) objects.

Parameters

`preferentialNetworkServiceConfigs`	`List`: list of preferential network configurations. This value cannot be `null`.

Throws

SecurityException (/reference/java/lang/SecurityException)	if the caller is not the profile owner or device owner.

See also:

`PreferentialNetworkServiceConfig` (/reference/android/app/admin/PreferentialNetworkServiceConfig)

setPreferentialNetworkServiceEnabled

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPreferentialNetworkServiceEnabled (boolean enabled)
```

Sets whether preferential network service is enabled. For example, an organization can have a deal/agreement with a carrier that all of the work data from its employees\u2019 devices will be sent via a network service dedicated for enterprise use. An example of a supported preferential network service is the Enterprise slice on 5G networks. For devices on 4G networks, the profile owner needs to additionally configure enterprise APN to set up data call for the preferential network service. These APNs can be added using `addOverrideApn(ComponentName, ApnSetting)` (/reference/android/app/admin/DevicePolicyManager#addOverrideApn(android.content.ComponentName, %20android.telephony.data.ApnSetting)) . By default, preferential network service is disabled on the work profile and fully managed devices, on supported carriers and devices. Admins can explicitly enable it with this API.

This method enables preferential network service with a default configuration. To fine-tune the configuration, use {@link #setPreferentialNetworkServiceConfigs) instead.

Before Android version {@link android.os.Build.VERSION_CODES#TIRAMISU}: this method can be called by the profile owner of a managed profile.

Starting from Android version {@link android.os.Build.VERSION_CODES#TIRAMISU}: This method can be called by the profile owner of a managed profile or device owner.

Parameters

`enabled`	`boolean`: whether preferential network service should be enabled.

Throws

SecurityException (/reference/java/lang/SecurityException)	if the caller is not the profile owner or device owner.

setProfileEnabled

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setProfileEnabled (ComponentName (/reference/android/content/ComponentName) a
```

Sets the enabled state of the profile. A profile should be enabled only once it is ready to be used. Only the profile owner can call this.

Parameters

`admin`	`ComponentName`: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not a profile owner.

See also:

`isProfileOwnerApp(String)`
(/reference/android/app/admin/DevicePolicyManager#isProfileOwnerApp(java.lang.String))

setProfileName

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setProfileName (ComponentName (/reference/android/content/ComponentName) admi
                String (/reference/java/lang/String) profileName)
```

Sets the name of the profile. In the device owner case it sets the name of the user which it is called from. Only a profile owner or device owner can call this. If this is never called by the profile or device owner, the name will be set to default values.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associate with. This value cannot be `null`.
`profileName`	`String`: The name of the profile. If the name is longer than 200 characters it will be truncated.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner.

See also:

`isProfileOwnerApp(String)`
(/reference/android/app/admin/DevicePolicyManager#isProfileOwnerApp(java.lang.String))

`isDeviceOwnerApp(String)` (/reference/android/app/admin/DevicePolicyManager#isDeviceOwnerApp(java.lang.String))

setRecommendedGlobalProxy

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setRecommendedGlobalProxy (ComponentName (/reference/android/content/Componer
                ProxyInfo (/reference/android/net/ProxyInfo) proxyInfo)
```

Set a network-independent global HTTP proxy. This is not normally what you want for typical HTTP proxies - they are generally network dependent. However if you're doing something unusual like general internal filtering this may be useful. On a private network where the proxy is not accessible, you may break HTTP using this.

This method requires the caller to be the device owner.

This proxy is only a recommendation and it is possible that some apps will ignore it.

Note: The device owner won't be able to set a global HTTP proxy if there are unaffiliated secondary users or profiles on the device. It's recommended that affiliation ids are set for new users as soon as possible after provisioning via `setAffiliationIds(ComponentName, Set)` (/reference/android/app/admin/DevicePolicyManager#setAffiliationIds(android.content.ComponentName,%20java.util.Set<java.lang.String>)) .

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`proxyInfo`	`ProxyInfo`: The a `ProxyInfo` (/reference/android/net/ProxyInfo) object defining the new global HTTP proxy. A `null` value will clear the global HTTP proxy.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not the device owner.

See also:

`ProxyInfo` (/reference/android/net/ProxyInfo)

setRequiredPasswordComplexity

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setRequiredPasswordComplexity (int passwordComplexity)
```

Sets a minimum password complexity requirement for the user's screen lock. The complexity level is one of the pre-defined levels, and the user is unable to set a password with a lower complexity level.

Note that when called on a profile which uses an unified challenge with its parent, the complexity would apply to the unified challenge.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to set restrictions on the parent profile.

Note: Specifying password requirements using this method clears any password requirements set using the obsolete `setPasswordQuality(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) and any of its associated methods. Additionally, if there are password requirements set using the obsolete `setPasswordQuality(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) on the parent `DevicePolicyManager` instance, they must be cleared by calling

`setPasswordQuality(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setPasswordQuality(android.content.ComponentName,%20int)) with `PASSWORD_QUALITY_UNSPECIFIED` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_QUALITY_UNSPECIFIED) on that instance prior to setting complexity requirement for the managed profile.

Parameters

`passwordComplexity`	`int`: Value is `PASSWORD_COMPLEXITY_NONE` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPL `PASSWORD_COMPLEXITY_LOW` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPL `PASSWORD_COMPLEXITY_MEDIUM` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPL , or `PASSWORD_COMPLEXITY_HIGH` (/reference/android/app/admin/DevicePolicyManager#PASSWORD_COMPL

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the calling application is not a device owner or a profile ow
`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if the complexity level is not one of the four above.
`IllegalStateException` (/reference/java/lang/IllegalStateException)	if the caller is trying to set password complexity while there `setPasswordQuality(android.content.ComponentNa` (/reference/android/app/admin/DevicePolicyManager#setPa on the parent `DevicePolicyManager` instance.

setRequiredStrongAuthTimeout

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setRequiredStrongAuthTimeout (ComponentName (/reference/android/content/Compo
                long timeoutMs)
```

Called by a device/profile owner to set the timeout after which unlocking with secondary, non strong auth (e.g. fingerprint, face, trust agents) times out, i.e. the user has to use a strong authentication method like password, pin or pattern.

This timeout is used internally to reset the timer to require strong auth again after specified timeout each time it has been successfully used.

Fingerprint can also be disabled altogether using `KEYGUARD_DISABLE_FINGERPRINT` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_FINGERPRINT).

Trust agents can also be disabled altogether using `KEYGUARD_DISABLE_TRUST_AGENTS` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_TRUST_AGENTS).

A calling device admin can verify the value it has set by calling `getRequiredStrongAuthTimeout(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getRequiredStrongAuthTimeout(android.content.ComponentName))
and passing in its instance.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName))
in order to set restrictions on the parent profile.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, calling this methods has no effect - i.e. the timeout is not set.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is assoc the caller is not a device admin This value may be `null`.
`timeoutMs`	`long`: The new timeout in milliseconds, after which the user will have to unl authentication method. A value of 0 means the admin is not participating ir timeout. The minimum and maximum timeouts are platform-defined and ar and 72 hours, respectively. Though discouraged, the admin may choose to auth at all times using `KEYGUARD_DISABLE_FINGERPRINT` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABL and/or `KEYGUARD_DISABLE_TRUST_AGENTS` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABL .

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner.

setResetPasswordToken

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean setResetPasswordToken (ComponentName (/reference/android/content/Component
                byte[] token)
```

Called by a profile or device owner to provision a token which can later be used to reset the device lockscreen password (if called by device owner), or managed profile challenge (if called by profile owner), via `resetPasswordWithToken(ComponentName, String, byte, int)` (/reference/android/app/admin/DevicePolicyManager#resetPasswordWithToken(android.content.ComponentName,%20java.lang.String,%20byte[],%20int))

.

If the user currently has a lockscreen password, the provisioned token will not be immediately usable; it only becomes active after the user performs a confirm credential operation, which

can be triggered by `KeyguardManager#createConfirmDeviceCredentialIntent` (/reference/android/app/KeyguardManager#createConfirmDeviceCredentialIntent(java.lang.CharSequence, %20java.lang.CharSequence)). If the user has no lockscreen password, the token is activated immediately. In all cases, the active state of the current token can be checked by `isResetPasswordTokenActive(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#isResetPasswordTokenActive(android.content.ComponentName)). For security reasons, un-activated tokens are only stored in memory and will be lost once the device reboots. In this case a new token needs to be provisioned again.

Once provisioned and activated, the token will remain effective even if the user changes or clears the lockscreen password.

This token is highly sensitive and should be treated at the same level as user credentials. In particular, NEVER store this token on device in plaintext. Do not store the plaintext token in device-encrypted storage if it will be needed to reset password on file-based encryption devices before user unlocks. Consider carefully how any password token will be stored on your server and who will need access to them. Tokens may be the subject of legal access requests.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, the reset token is not set and this method returns false.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.
`token`	`byte`: a secure token a least 32-byte long, which must be generated by a cryptographically strong random number generator.

Returns

`boolean`	true if the operation is successful, false otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if admin is not a device or profile owner.
`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if the supplied token is invalid.

setRestrictionsProvider Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setRestrictionsProvider (ComponentName (/reference/android/content/ComponentN
                ComponentName (/reference/android/content/ComponentName) provider)
```

Designates a specific service component as the provider for making permission requests of a local or remote administrator of the user.

Only a device owner or profile owner can designate the restrictions provider.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`provider`	`ComponentName`: The component name of the service that implements `RestrictionsReceiver` (/reference/android/service/restrictions/RestrictionsReceiver). If this param is null, it removes the restrictions provider previously assigned.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner.

setScreenCaptureDisabled

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setScreenCaptureDisabled (ComponentName (/reference/android/content/ComponentName) admin,
                boolean disabled)
```

Called by a device/profile owner to set whether the screen capture is disabled. Disabling screen capture also prevents the content from being shown on display devices that do not have a secure video output. See `Display.FLAG_SECURE` (/reference/android/view/Display#FLAG_SECURE) for more details about secure surfaces and secure displays.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance, returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)), where the calling device admin must be the profile owner of an organization-owned managed profile. If it is not, a security exception will be thrown.

If the caller is device owner or called on the parent instance by a profile owner of an organization-owned managed profile, then the restriction will be applied to all users.

From version `Build.VERSION_CODES.M` (/reference/android/os/Build.VERSION_CODES#M) disabling screen capture also blocks assist requests for all activities of the relevant user.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.
`disabled`	`boolean`: Whether screen capture is disabled or not.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not permitted to control screen capture policy.

setSecureSetting

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setSecureSetting (ComponentName (/reference/android/content/ComponentName) ad
                String (/reference/java/lang/String) setting,
                String (/reference/java/lang/String) value)
```

This method is mostly deprecated. Most of the settings that still have an effect have dedicated setter methods (e.g. `setLocationEnabled(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setLocationEnabled(android.content.ComponentName,%20boolean))) or user restrictions.

Called by profile or device owners to update `Settings.Secure` (/reference/android/provider/Settings.Secure) settings. Validation that the value of the setting is in the correct form for the setting type should be performed by the caller.

The settings that can be updated by a profile or device owner with this method are:

- `Settings.Secure.DEFAULT_INPUT_METHOD` (/reference/android/provider/Settings.Secure#DEFAULT_INPUT_METHOD)

- `Settings.Secure.SKIP_FIRST_USE_HINTS` (/reference/android/provider/Settings.Secure#SKIP_FIRST_USE_HINTS)

A device owner can additionally update the following settings:

- `Settings.Secure.LOCATION_MODE` (/reference/android/provider/Settings.Secure#LOCATION_MODE), but see note below.

Note: Starting from Android O, apps should no longer call this method with the setting `Settings.Secure.INSTALL_NON_MARKET_APPS` (/reference/android/provider/Settings.Secure#INSTALL_NON_MARKET_APPS), which is deprecated. Instead, device owners or profile owners should use the restriction `UserManager#DISALLOW_INSTALL_UNKNOWN_SOURCES` (/reference/android/os/UserManager#DISALLOW_INSTALL_UNKNOWN_SOURCES). If any app targeting `Build.VERSION_CODES.O` (/reference/android/os/Build.VERSION_CODES#O) or higher calls this method with `Settings.Secure.INSTALL_NON_MARKET_APPS` (/reference/android/provider/Settings.Secure#INSTALL_NON_MARKET_APPS), an `UnsupportedOperationException` (/reference/java/lang/UnsupportedOperationException) is thrown. Starting from Android Q, the device and profile owner can also call `UserManager#DISALLOW_INSTALL_UNKNOWN_SOURCES_GLOBALLY` (/reference/android/os/UserManager#DISALLOW_INSTALL_UNKNOWN_SOURCES_GLOBALLY) to restrict unknown sources for all users. Note: Starting from Android R, apps should no longer call this method with the setting `Settings.Secure.LOCATION_MODE` (/reference/android/provider/Settings.Secure#LOCATION_MODE), which is deprecated. Instead, device owners should call `setLocationEnabled(android.content.ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setLocationEnabled(android.content.ComponentName,%20boolean)). This will be enforced for all apps targeting Android R or above.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`setting`	`String`: The name of the setting to update.

`value`	`String`: The value to update the setting to.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner.

setSecurityLoggingEnabled

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setSecurityLoggingEnabled (ComponentName (/reference/android/content/Componer
                boolean enabled)
```

Called by device owner or a profile owner of an organization-owned managed profile to control the security logging feature.

Security logs contain various information intended for security auditing purposes. When security logging is enabled by any app other than the device owner, certain security logs are not visible (for example personal app launch events) or they will be redacted (for example, details of the physical volume mount events). Please see `SecurityEvent` (/reference/android/app/admin/SecurityLog.SecurityEvent) for details.

Note: The device owner won't be able to retrieve security logs if there are unaffiliated secondary users or profiles on the device, regardless of whether the feature is enabled. Logs will be discarded if the internal buffer fills up while waiting for all users to become affiliated. Therefore it's recommended that affiliation ids are set for new users as soon as possible after provisioning via `setAffiliationIds(ComponentName, Set)` (/reference/android/app/admin/DevicePolicyManager#setAffiliationIds(android.content.ComponentName,%20java.util.Set<java.lang.String>)) . Non device owners are not subject to this restriction since all privacy-sensitive events happening outside the managed profile would have been redacted already. Starting from `ERROR(Build.VERSION_CODES#VANILLA_ICE_CREAM/android.os.Build.VERSION_CODES#VANILLA_ICE_CREAM Build.VERSION_CODES#VANILLA_ICE_CREAM)` (/), after the security logging policy has been set, `PolicyUpdateReceiver#onPolicySetResult(Context, String,`

`Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicySetResult(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin on whether the policy was successfully set or not. This callback will contain:

- The policy identifier `ERROR(DevicePolicyIdentifiers#SECURITY_LOGGING_POLICY/android.app.admin.DevicePolicyIdentifiers#SECURITY_LOGGING_POLICY DevicePolicyIdentifiers#SECURITY_LOGGING_POLICY)` (/)
- The `TargetUser` (/reference/android/app/admin/TargetUser) that this policy relates to
- The `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult), which will be `PolicyUpdateResult#RESULT_POLICY_SET` (/reference/android/app/admin/PolicyUpdateResult#RESULT_POLICY_SET) if the policy was successfully set or the reason the policy failed to be set e.g. `PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY` (/reference/android/app/admin/PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY))

If there has been a change to the policy, `PolicyUpdateReceiver#onPolicyChanged(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicyChanged(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin of this change. This callback will contain the same parameters as PolicyUpdateReceiver#onPolicySetResult and the `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult) will contain the reason why the policy changed.

Parameters

`admin`	`ComponentName`: Which device admin this request is associated with, or `null` if called by a delegated app.
`enabled`	`boolean`: whether security logging should be enabled or not.

Throws

SecurityException (/reference/java/lang/SecurityException)	if the caller is not permitted to control security logging.

See also:

setAffiliationIds(ComponentName, Set) (/reference/android/app/admin/DevicePolicyManager#setAffiliationIds(android.content.ComponentName,%20java.util.Set<java.lang.String>))

retrieveSecurityLogs(ComponentName) (/reference/android/app/admin/DevicePolicyManager#retrieveSecurityLogs(android.content.ComponentName))

setShortSupportMessage added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setShortSupportMessage (ComponentName (/reference/android/content/ComponentNa
                CharSequence (/reference/java/lang/CharSequence) message)
```

Called by a device admin to set the short support message. This will be displayed to the user in settings screens where functionality has been disabled by the admin. The message should be limited to a short statement such as "This setting is disabled by your administrator. Contact someone@example.com for support." If the message is longer than 200 characters it may be truncated.

If the short support message needs to be localized, it is the responsibility of the DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) to listen to the Intent#ACTION_LOCALE_CHANGED (/reference/android/content/Intent#ACTION_LOCALE_CHANGED) broadcast and set a new version of this string accordingly.

Parameters

admin	ComponentName: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is

	associated with. Null if the caller is not a device admin. This value may be `null`.
`message`	`CharSequence`: Short message to be displayed to the user in settings or null to clear the existing message.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not an active administrator.

See also:

`setLongSupportMessage(ComponentName, CharSequence)` (/reference/android/app/admin/DevicePolicyManager#setLongSupportMessage(android.content.ComponentName,%20java.lang.CharSequence))

setStartUserSessionMessage

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setStartUserSessionMessage (ComponentName (/reference/android/content/Compone
                CharSequence (/reference/java/lang/CharSequence) startUserSessionMessage)
```

Called by a device owner to specify the user session start message. This may be displayed during a user switch.

The message should be limited to a short statement or it may be truncated.

If the message needs to be localized, it is the responsibility of the `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) to listen to the `Intent#ACTION_LOCALE_CHANGED` (/reference/android/content/Intent#ACTION_LOCALE_CHANGED) broadcast and set a new version of this message accordingly.

Parameters

`admin`	`ComponentName`: which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`startUserSessionMessage`	`CharSequence`: message for starting user session, or `null` to use system default message.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device owner.

setStatusBarDisabled Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean setStatusBarDisabled (ComponentName (/reference/android/content/ComponentN
                boolean disabled)
```

Called by device owner or profile owner of secondary users that is affiliated with the device to disable the status bar. Disabling the status bar blocks notifications and quick settings.

Note: This method has no effect for LockTask mode. The behavior of the status bar in LockTask mode can be configured with `setLockTaskFeatures(android.content.ComponentName, int)` (/reference/android/app/admin/DevicePolicyManager#setLockTaskFeatures(android.content.ComponentName,%20int)) . Calls to this method when the device is in LockTask mode will be registered, but will only take effect when the device leaves LockTask mode.

This policy does not have any effect while on the lock screen, where the status bar will not be disabled. Using LockTask instead of this method is recommended.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.
`disabled`	`boolean`: `true` disables the status bar, `false` reenables it.

Returns

`boolean`	`false` if attempting to disable the status bar failed. `true` otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not the device owner, or a profile owner of secondary user that is affiliated with the device.

See also:

`isAffiliatedUser()` (/reference/android/app/admin/DevicePolicyManager#isAffiliatedUser())

`getSecondaryUsers(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getSecondaryUsers(android.content.ComponentName))

setStorageEncryption

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int setStorageEncryption (ComponentName (/reference/android/content/ComponentName)
                boolean encrypt)
```

This method was deprecated in API level 30.
This method does not actually modify the storage encryption of the device. It has never affected the encryption status of a device. Called by an application that is administering the device to request that the storage system be encrypted. Does nothing if the caller is on a secondary user or a managed profile.

When multiple device administrators attempt to control device encryption, the most secure, supported setting will always be used. If any device administrator requests device encryption, it will be enabled; Conversely, if a device administrator attempts to disable device encryption while another device administrator has enabled it, the call to disable will fail (most commonly returning `ENCRYPTION_STATUS_ACTIVE` (/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATUS_ACTIVE)).

This policy controls encryption of the secure (application data) storage area. Data written to other storage areas may or may not be encrypted, and this policy does not require or control the encryption of any other storage areas. There is one exception: If `Environment.isExternalStorageEmulated()` (/reference/android/os/Environment#isExternalStorageEmulated()) is `true`, then the directory returned by `Environment.getExternalStorageDirectory()` (/reference/android/os/Environment#getExternalStorageDirectory()) must be written to disk within the encrypted storage area.

Important Note: On some devices, it is possible to encrypt storage without requiring the user to create a device PIN or Password. In this case, the storage is encrypted, but the encryption key may not be fully secured. For maximum security, the administrator should also require (and check for) a pattern, PIN, or password.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`encrypt`	`boolean`: true to request encryption, false to release any previous request

Returns

`int`	the new total request status (for all active admins), or **ENCRYPTION_STATU** (/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATU if called for a non-system user. Will be one of **ENCRYPTION_STATUS_UNSU** (/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATU , **ENCRYPTION_STATUS_INACTIVE** (/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATU **ENCRYPTION_STATUS_ACTIVE** (/reference/android/app/admin/DevicePolicyManager#ENCRYPTION_STATU the value of the requests; use **getStorageEncryptionStatus()** (/reference/android/app/admin/DevicePolicyManager#getStorageEncryptio query the actual device state.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not an active administrator or does not use **DeviceAdminInfo#USES_ENCRYPTED_STORAGE** (/reference/android/app/admin/DeviceAdminInfo#USES_ENCRYPTE

setSystemSetting

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setSystemSetting (ComponentName (/reference/android/content/ComponentName) ad
                String (/reference/java/lang/String) setting,
                String (/reference/java/lang/String) value)
```

Called by a device or profile owner to update `Settings.System` (/reference/android/provider/Settings.System) settings. Validation that the value of the setting is in the correct form for the setting type should be performed by the caller.

The settings that can be updated by a device owner or profile owner of secondary user with this method are:

- `Settings.System.SCREEN_BRIGHTNESS` (/reference/android/provider/Settings.System#SCREEN_BRIGHTNESS)
- `Settings.System.SCREEN_BRIGHTNESS_MODE` (/reference/android/provider/Settings.System#SCREEN_BRIGHTNESS_MODE)
- `Settings.System.SCREEN_OFF_TIMEOUT` (/reference/android/provider/Settings.System#SCREEN_OFF_TIMEOUT)

Starting from Android `ERROR(/android.os.Build.VERSION_CODES#VANILLA_ICE_CREAM)` (/), a profile owner on an organization-owned device can call this method on the parent `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) to set system settings on the parent user.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`setting`	`String`: The name of the setting to update. This value cannot be `null`. Val is `Settings.System.SCREEN_BRIGHTNESS_MODE` (/reference/android/provider/Settings.System#SCREEN_BRIGHTNESS_MOD , `Settings.System.SCREEN_BRIGHTNESS` (/reference/android/provider/Settings.System#SCREEN_BRIGHTNESS), android.provider.Settings.System.SCREEN_BRIGHTNESS_FLOAT, or `Settings.System.SCREEN_OFF_TIMEOUT` (/reference/android/provider/Settings.System#SCREEN_OFF_TIMEOUT)
`value`	`String`: The value to update the setting to.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device or profile owner.

See also:

Settings.System.SCREEN_OFF_TIMEOUT (/reference/android/provider/Settings.System#SCREEN_OFF_TIMEOUT)

setSystemUpdatePolicy

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setSystemUpdatePolicy (ComponentName (/reference/android/content/ComponentNam
                SystemUpdatePolicy (/reference/android/app/admin/SystemUpdatePolicy) policy)
```

Called by device owners or profile owners of an organization-owned managed profile to set a local system update policy. When a new policy is set, ACTION_SYSTEM_UPDATE_POLICY_CHANGED (/reference/android/app/admin/DevicePolicyManager#ACTION_SYSTEM_UPDATE_POLICY_CHANGED) is broadcast.

If the supplied system update policy has freeze periods set but the freeze periods do not meet 90-day maximum length or 60-day minimum separation requirement set out in SystemUpdatePolicy#setFreezePeriods (/reference/android/app/admin/SystemUpdatePolicy#setFreezePeriods(java.util.List<android.app.admin.FreezePeriod>)), SystemUpdatePolicy.ValidationFailedException (/reference/android/app/admin/SystemUpdatePolicy.ValidationFailedException) will the thrown. Note that the system keeps a record of freeze periods the device experienced previously, and combines them with the new freeze periods to be set when checking the maximum freeze length and minimum freeze separation constraints. As a result, freeze periods that passed validation during SystemUpdatePolicy#setFreezePeriods (/reference/android/app/admin/SystemUpdatePolicy#setFreezePeriods(java.util.List<android.app.admin.FreezePeriod>)) might fail the additional checks here due to the freeze period history. If this is causing issues during development, `adb shell dpm clear-freeze-period-record` can be used to clear the record.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. All components in the package can set system update policies and the most recent policy takes effect. This should be null if the caller is not a device admin.
`policy`	`SystemUpdatePolicy`: the new policy, or `null` to clear the current policy.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device owne a profile owner of an organization-owned managec profile, or the caller is not permitted to set this policy
`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if the policy type or maintena window is not valid.
`SystemUpdatePolicy.ValidationFailedException` (/reference/android/app/admin/SystemUpdatePolicy.ValidationFailedException)	if the policy's freeze period d not meet the requirement.

See also:

`SystemUpdatePolicy` (/reference/android/app/admin/SystemUpdatePolicy)

`SystemUpdatePolicy.setFreezePeriods(List)` (/reference/android/app/admin/SystemUpdatePolicy#setFreezePeriods(java.util.List<android.app.admin.FreezePeriod>))

setTime

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean setTime (ComponentName (/reference/android/content/ComponentName) admin,
                long millis)
```

Called by a device owner or a profile owner of an organization-owned managed profile to set the system wall clock time. This only takes effect if called when `Settings.Global.AUTO_TIME` (/reference/android/provider/Settings.Global#AUTO_TIME) is 0, otherwise `false` will be returned.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.
`millis`	`long`: time in milliseconds since the Epoch

Returns

`boolean`	`true` if set time succeeded, `false` otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device owner or a profile owner of an organization-owned managed profile.

setTimeZone

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean setTimeZone (ComponentName (/reference/android/content/ComponentName) admi
```

String (/reference/java/lang/String) timeZone)

Called by a device owner or a profile owner of an organization-owned managed profile to set the system's persistent default time zone. This only takes effect if called when Settings.Global.AUTO_TIME_ZONE (/reference/android/provider/Settings.Global#AUTO_TIME_ZONE) is 0, otherwise `false` will be returned.

Parameters

admin	ComponentName: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be null.
timeZone	String: one of the Olson ids from the list returned by TimeZone.getAvailableIDs() (/reference/java/util/TimeZone#getAvailableIDs())

Returns

boolean	true if set timezone succeeded, false otherwise.

Throws

SecurityException (/reference/java/lang/SecurityException)	if admin is not a device owner or a profile owner of an organization-owned managed profile.

See also:

AlarmManager.setTimeZone(String) (/reference/android/app/AlarmManager#setTimeZone(java.lang.String))

setTrustAgentConfiguration

added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTrustAgentConfiguration (ComponentName (/reference/android/content/Compone
                ComponentName (/reference/android/content/ComponentName) target,
                PersistableBundle (/reference/android/os/PersistableBundle) configuration)
```

Sets a list of configuration features to enable for a trust agent component. This is meant to be used in conjunction with `KEYGUARD_DISABLE_TRUST_AGENTS` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_TRUST_AGENTS), which disables all trust agents but those enabled by this function call. If flag `KEYGUARD_DISABLE_TRUST_AGENTS` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_TRUST_AGENTS) is not set, then this call has no effect.

For any specific trust agent, whether it is disabled or not depends on the aggregated state of each admin's `KEYGUARD_DISABLE_TRUST_AGENTS` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_TRUST_AGENTS) setting and its trust agent configuration as set by this function call. In particular: if any admin sets `KEYGUARD_DISABLE_TRUST_AGENTS` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_TRUST_AGENTS) and does not additionally set any trust agent configuration, the trust agent is disabled completely. Otherwise, the trust agent will receive the list of configurations from all admins who set `KEYGUARD_DISABLE_TRUST_AGENTS` (/reference/android/app/admin/DevicePolicyManager#KEYGUARD_DISABLE_TRUST_AGENTS) and aggregate the configurations to determine its behavior. The exact meaning of aggregation is trust-agent-specific.

A calling device admin must have requested `DeviceAdminInfo#USES_POLICY_DISABLE_KEYGUARD_FEATURES` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_DISABLE_KEYGUARD_FEATURES) to be able to call this method; if not, a security exception will be thrown.

This method can be called on the `DevicePolicyManager` (/reference/android/app/admin/DevicePolicyManager) instance returned by `getParentProfileInstance(android.content.ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getParentProfileInstance(android.content.ComponentName)) in order to set the configuration for the parent profile.

On devices not supporting `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature, calling this method has no effect - no trust agent configuration will be set.
Requires the `PackageManager#FEATURE_SECURE_LOCK_SCREEN` (/reference/android/content/pm/PackageManager#FEATURE_SECURE_LOCK_SCREEN) feature which can be detected using `PackageManager.hasSystemFeature(String)` (/reference/android/content/pm/PackageManager#hasSystemFeature(java.lang.String)).

Parameters

`admin`	**`ComponentName`**: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin This value may be **`null`**.
`target`	**`ComponentName`**: Component name of the agent to be configured. This value cannot be **`null`**.
`configuration`	**`PersistableBundle`**: Trust-agent-specific feature configuration bundle. Please consult documentation of the specific trust agent to determine the interpretation of this bundle.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if **`admin`** is not an active administrator or does not use `DeviceAdmi` `DISABLE_KEYGUARD_FEATURES` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_DI

setUninstallBlocked

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setUninstallBlocked (ComponentName (/reference/android/content/ComponentName)
                String (/reference/java/lang/String) packageName,
```

```
            boolean uninstallBlocked)
```

Change whether a user can uninstall a package. This function can be called by a device owner, profile owner, or by a delegate given the `DELEGATION_BLOCK_UNINSTALL` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_BLOCK_UNINSTALL) scope via `setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>)) or holders of the permission `Manifest.permission.MANAGE_DEVICE_POLICY_APPS_CONTROL` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_APPS_CONTROL).

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), after the set uninstall blocked policy has been set, `PolicyUpdateReceiver#onPolicySetResult(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicySetResult(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin on whether the policy was successfully set or not. This callback will contain:

- The policy identifier `DevicePolicyIdentifiers#PACKAGE_UNINSTALL_BLOCKED_POLICY` (/reference/android/app/admin/DevicePolicyIdentifiers#PACKAGE_UNINSTALL_BLOCKED_POLICY)
- The additional policy params bundle, which contains `PolicyUpdateReceiver#EXTRA_PACKAGE_NAME` (/reference/android/app/admin/PolicyUpdateReceiver#EXTRA_PACKAGE_NAME) the package name the policy applies to
- The `TargetUser` (/reference/android/app/admin/TargetUser) that this policy relates to
- The `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult), which will be `PolicyUpdateResult#RESULT_POLICY_SET` (/reference/android/app/admin/PolicyUpdateResult#RESULT_POLICY_SET) if the policy was successfully set or the reason the policy failed to be set (e.g. `PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY` (/reference/android/app/admin/PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY))
)

If there has been a change to the policy, `PolicyUpdateReceiver#onPolicyChanged(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicyChanged(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin of this change. This callback will contain the same parameters as PolicyUpdateReceiver#onPolicySetResult and the `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult) will contain the reason why the policy changed.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.
`packageName`	`String`: package to change.
`uninstallBlocked`	`boolean`: true if the user shouldn't be able to uninstall the package.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device or profile owner or holder of the permission `permission.MANAGE_DEVICE_POLICY_APPS_CONTROL` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC.

See also:

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`DELEGATION_BLOCK_UNINSTALL` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_BLOCK_UNINSTALL)

setUsbDataSignalingEnabled

Added in API level 31 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setUsbDataSignalingEnabled (boolean enabled)
```

Called by a device owner or profile owner of an organization-owned managed profile to enable or disable USB data signaling for the device. When disabled, USB data connections (except from charging functions) are prohibited.

This API is not supported on all devices, the caller should call `canUsbDataSignalingBeDisabled()` (/reference/android/app/admin/DevicePolicyManager#canUsbDataSignalingBeDisabled()) to check whether enabling or disabling USB data signaling is supported on the device. Starting from Android 15, after the USB data signaling policy has been set, `PolicyUpdateReceiver#onPolicySetResult(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicySetResult(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin on whether the policy was successfully set or not. This callback will contain:

- The `TargetUser` (/reference/android/app/admin/TargetUser) that this policy relates to
- The `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult), which will be `PolicyUpdateResult#RESULT_POLICY_SET` (/reference/android/app/admin/PolicyUpdateResult#RESULT_POLICY_SET) if the policy was successfully set or the reason the policy failed to be set e.g. `PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY` (/reference/android/app/admin/PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY))

If there has been a change to the policy, `PolicyUpdateReceiver#onPolicyChanged(Context, String, Bundle, TargetUser, PolicyUpdateResult)`

(/reference/android/app/admin/PolicyUpdateReceiver#onPolicyChanged(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult))
will notify the admin of this change. This callback will contain the same parameters as PolicyUpdateReceiver#onPolicySetResult and the `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult) will contain the reason why the policy changed.

Parameters

`enabled`	**`boolean`**: whether USB data signaling should be enabled or not.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the caller is not permitted to set this policy
`IllegalStateException` (/reference/java/lang/IllegalStateException)	if disabling USB data signaling is not supported or if USB data signaling fails to be enabled/disabled.

setUserControlDisabledPackages

Added in API level 30 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setUserControlDisabledPackages (ComponentName (/reference/android/content/Com
            List (/reference/java/util/List)<String (/reference/java/lang/String)> packages)
```

Called by a device owner or a profile owner or holder of the permission `Manifest.permission.MANAGE_DEVICE_POLICY_APPS_CONTROL` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLICY_APPS_CONTROL) to disable user control over apps. User will not be able to clear app data or force-stop packages. When called by a device owner, applies to all users on the device. Packages with user control disabled are exempted from App Standby Buckets.

Starting from `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE), after the user control disabled packages policy has been set, `PolicyUpdateReceiver#onPolicySetResult(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicySetResult(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin on whether the policy was successfully set or not. This callback will contain:

- The policy identifier `DevicePolicyIdentifiers#USER_CONTROL_DISABLED_PACKAGES_POLICY` (/reference/android/app/admin/DevicePolicyIdentifiers#USER_CONTROL_DISABLED_PACKAGES_POLICY)
- The `TargetUser` (/reference/android/app/admin/TargetUser) that this policy relates to
- The `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult), which will be `PolicyUpdateResult#RESULT_POLICY_SET` (/reference/android/app/admin/PolicyUpdateResult#RESULT_POLICY_SET) if the policy was successfully set or the reason the policy failed to be set (e.g. `PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY` (/reference/android/app/admin/PolicyUpdateResult#RESULT_FAILURE_CONFLICTING_ADMIN_POLICY)))

If there has been a change to the policy, `PolicyUpdateReceiver#onPolicyChanged(Context, String, Bundle, TargetUser, PolicyUpdateResult)` (/reference/android/app/admin/PolicyUpdateReceiver#onPolicyChanged(android.content.Context,%20java.lang.String,%20android.os.Bundle,%20android.app.admin.TargetUser,%20android.app.admin.PolicyUpdateResult)) will notify the admin of this change. This callback will contain the same parameters as PolicyUpdateReceiver#onPolicySetResult and the `PolicyUpdateResult` (/reference/android/app/admin/PolicyUpdateResult) will contain the reason why the policy changed.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. Null if the caller is not a device admin. This value may be `null`.

`packages`	`List`: The package names for the apps. This value cannot be `null`.

Throws

SecurityException (/reference/java/lang/SecurityException)	if `admin` is not a device owner or a profile owner or holder of the pe Manifest.permission.MANAGE_DEVICE_POLICY_APPS_CONTR (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC .

setUserIcon

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setUserIcon (ComponentName (/reference/android/content/ComponentName) admin,
                Bitmap (/reference/android/graphics/Bitmap) icon)
```

Called by profile or device owners to set the user's photo.

Parameters

`admin`	`ComponentName`: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`icon`	`Bitmap`: the bitmap to set as the photo.

Throws

SecurityException	if `admin` is not a device or profile owner.

(/reference/java/lang/SecurityException)

setWifiSsidPolicy

Added in API level 33 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setWifiSsidPolicy (WifiSsidPolicy (/reference/android/app/admin/WifiSsidPolicy) p
```

Called by device owner or profile owner of an organization-owned managed profile to specify the Wi-Fi SSID policy (`WifiSsidPolicy` (/reference/android/app/admin/WifiSsidPolicy)). Wi-Fi SSID policy specifies the SSID restriction the network must satisfy in order to be eligible for a connection. Providing a null policy results in the deactivation of the SSID restriction

Parameters

`policy`	`WifiSsidPolicy`: Wi-Fi SSID policy This value may be `null`.

Throws

SecurityException (/reference/java/lang/SecurityException)	if the caller is not permitted to manage wifi policy

startUserInBackground

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int startUserInBackground (ComponentName (/reference/android/content/ComponentName
                UserHandle (/reference/android/os/UserHandle) userHandle)
```

Called by a device owner to start the specified secondary user in background.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.
`userHandle`	`UserHandle`: the user to be started in background. This value cannot be `null`.

Returns

`int`	one of the following result codes: `UserManager#USER_OPERATION_ERRO` (/reference/android/os/UserManager#USER_OPERATION_ERROR_UNKNOV `UserManager#USER_OPERATION_SUCCESS` (/reference/android/os/UserManager#USER_OPERATION_SUCCESS), `UserManager#USER_OPERATION_ERROR_MANAGED_PROFILE` (/reference/android/os/UserManager#USER_OPERATION_ERROR_MANAGE `UserManager#USER_OPERATION_ERROR_MAX_RUNNING_USERS` (/reference/android/os/UserManager#USER_OPERATION_ERROR_MAX_RU , Value is `UserManager.USER_OPERATION_SUCCESS` (/reference/android/os/UserManager#USER_OPERATION_SUCCESS), `User USER_OPERATION_ERROR_UNKNOWN` (/reference/android/os/UserManager#USER_OPERATION_ERROR_UNKNOV `UserManager.USER_OPERATION_ERROR_MANAGED_PROFILE` (/reference/android/os/UserManager#USER_OPERATION_ERROR_MANAGE `UserManager.USER_OPERATION_ERROR_MAX_RUNNING_USERS` (/reference/android/os/UserManager#USER_OPERATION_ERROR_MAX_RU , `UserManager.USER_OPERATION_ERROR_CURRENT_USER` (/reference/android/os/UserManager#USER_OPERATION_ERROR_CURREN `UserManager.USER_OPERATION_ERROR_LOW_STORAGE` (/reference/android/os/UserManager#USER_OPERATION_ERROR_LOW_ST `UserManager.USER_OPERATION_ERROR_MAX_USERS` (/reference/android/os/UserManager#USER_OPERATION_ERROR_MAX_US android.os.UserManager.USER_OPERATION_ERROR_USER_ACCOUNT_ALRE android.os.UserManager.USER_OPERATION_ERROR_DISABLED_USER, or android.os.UserManager.USER_OPERATION_ERROR_PRIVATE_PROFILE

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device owner.

See also:

`getSecondaryUsers(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getSecondaryUsers(android.content.ComponentName))

stopUser

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int stopUser (ComponentName (/reference/android/content/ComponentName) admin,
                UserHandle (/reference/android/os/UserHandle) userHandle)
```

Called by a device owner to stop the specified secondary user.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be **`null`**.
`userHandle`	`UserHandle`: the user to be stopped. This value cannot be **`null`**.

Returns

`int`	one of the following result codes: `UserManager#USER_OPERATION_ERRO` (/reference/android/os/UserManager#USER_OPERATION_ERROR_UNKNOW `UserManager#USER_OPERATION_SUCCESS`

(/reference/android/os/UserManager#USER_OPERATION_SUCCESS),
UserManager#USER_OPERATION_ERROR_MANAGED_PROFILE
(/reference/android/os/UserManager#USER_OPERATION_ERROR_MANAGE
UserManager#USER_OPERATION_ERROR_CURRENT_USER
(/reference/android/os/UserManager#USER_OPERATION_ERROR_CURREN
is UserManager.USER_OPERATION_SUCCESS
(/reference/android/os/UserManager#USER_OPERATION_SUCCESS), User
USER_OPERATION_ERROR_UNKNOWN
(/reference/android/os/UserManager#USER_OPERATION_ERROR_UNKNOV
UserManager.USER_OPERATION_ERROR_MANAGED_PROFILE
(/reference/android/os/UserManager#USER_OPERATION_ERROR_MANAGE
UserManager.USER_OPERATION_ERROR_MAX_RUNNING_USERS
(/reference/android/os/UserManager#USER_OPERATION_ERROR_MAX_RU
, UserManager.USER_OPERATION_ERROR_CURRENT_USER
(/reference/android/os/UserManager#USER_OPERATION_ERROR_CURREN
UserManager.USER_OPERATION_ERROR_LOW_STORAGE
(/reference/android/os/UserManager#USER_OPERATION_ERROR_LOW_ST
UserManager.USER_OPERATION_ERROR_MAX_USERS
(/reference/android/os/UserManager#USER_OPERATION_ERROR_MAX_US
android.os.UserManager.USER_OPERATION_ERROR_USER_ACCOUNT_ALRE
android.os.UserManager.USER_OPERATION_ERROR_DISABLED_USER, or
android.os.UserManager.USER_OPERATION_ERROR_PRIVATE_PROFILE

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not a device owner.

See also:

`getSecondaryUsers(ComponentName)`
(/reference/android/app/admin/DevicePolicyManager#getSecondaryUsers(android.content.ComponentNam
e))

switchUser

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean switchUser (ComponentName (/reference/android/content/ComponentName) admin
                UserHandle (/reference/android/os/UserHandle) userHandle)
```

Called by a device owner to switch the specified secondary user to the foreground.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be **`null`**.
`userHandle`	`UserHandle`: the user to switch to; null will switch to primary.

Returns

`boolean`	**`true`** if the switch was successful, **`false`** otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if **`admin`** is not a device owner.

See also:

`Intent.ACTION_USER_FOREGROUND` (/reference/android/content/Intent#ACTION_USER_FOREGROUND)

`getSecondaryUsers(ComponentName)` (/reference/android/app/admin/DevicePolicyManager#getSecondaryUsers(android.content.ComponentName))

transferOwnership

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void transferOwnership (ComponentName (/reference/android/content/ComponentName) a
                ComponentName (/reference/android/content/ComponentName) target,
                PersistableBundle (/reference/android/os/PersistableBundle) bundle)
```

Changes the current administrator to another one. All policies from the current administrator are migrated to the new administrator. The whole operation is atomic - the transfer is either complete or not done at all.

Depending on the current administrator (device owner, profile owner), you have the following expected behaviour:

- A device owner can only be transferred to a new device owner
- A profile owner can only be transferred to a new profile owner

Use the `bundle` parameter to pass data to the new administrator. The data will be received in the `DeviceAdminReceiver#onTransferOwnershipComplete(Context, PersistableBundle)` (/reference/android/app/admin/DeviceAdminReceiver#onTransferOwnershipComplete(android.content.Context,%20android.os.PersistableBundle)) callback of the new administrator.

The transfer has failed if the original administrator is still the corresponding owner after calling this method.

The incoming target administrator must have the `<support-transfer-ownership />` tag inside the `<device-admin></device-admin>` tags in the xml file referenced by `DeviceAdminReceiver#DEVICE_ADMIN_META_DATA` (/reference/android/app/admin/DeviceAdminReceiver#DEVICE_ADMIN_META_DATA). Otherwise an `IllegalArgumentException` (/reference/java/lang/IllegalArgumentException) will be thrown.

Parameters

`admin`	`ComponentName`: which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with. This value cannot be `null`.

`target`	`ComponentName`: which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) we want the new administrator to be. This value cannot be `null`.
`bundle`	`PersistableBundle`: data to be sent to the new administrator. This value may be `null`.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if `admin` is not a device owner nor a profile owner.
`IllegalArgumentException` (/reference/java/lang/IllegalArgumentException)	if `admin` or `target` is `null`, they are components in the same package or `target` is not an active admin.

uninstallAllUserCaCerts

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void uninstallAllUserCaCerts (ComponentName (/reference/android/content/ComponentN
```

Uninstalls all custom trusted CA certificates from the profile. Certificates installed by means other than device policy will also be removed, except for system CA certificates.

Parameters

`admin`	`ComponentName`: Which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or `null` if calling from a delegated certificate installer.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not **null** and not a device or profile owner.

uninstallCaCert

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void uninstallCaCert (ComponentName (/reference/android/content/ComponentName) adm
                byte[] certBuffer)
```

Uninstalls the given certificate from trusted user CAs, if present. The caller must be a profile or device owner on that user, or a delegate package given the `DELEGATION_CERT_INSTALL` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_INSTALL) scope via `setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>)) ; otherwise a security exception will be thrown.

Parameters

admin	**ComponentName**: Which DeviceAdminReceiver (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with, or **null** if calling from a delegated certificate installer.
certBuffer	**byte**: encoded form of the certificate to remove.

Throws

SecurityException (/reference/java/lang/SecurityException)	if **admin** is not **null** and not a device or profile owner.

See also:

`setDelegatedScopes(ComponentName, String, List)` (/reference/android/app/admin/DevicePolicyManager#setDelegatedScopes(android.content.ComponentName,%20java.lang.String,%20java.util.List<java.lang.String>))

`DELEGATION_CERT_INSTALL` (/reference/android/app/admin/DevicePolicyManager#DELEGATION_CERT_INSTALL)

updateOverrideApn

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean updateOverrideApn (ComponentName (/reference/android/content/ComponentName
                int apnId,
                ApnSetting (/reference/android/telephony/data/ApnSetting) apnSetting)
```

Called by device owner or managed profile owner to update an override APN.

This method may returns `false` if there is no override APN with the given `apnId`.

This method may also returns `false` if `apnSetting` conflicts with an existing override APN. Update the existing conflicted APN instead.

See `addOverrideApn(ComponentName, ApnSetting)` (/reference/android/app/admin/DevicePolicyManager#addOverrideApn(android.content.ComponentName,%20android.telephony.data.ApnSetting)) for the definition of conflict.

Before Android version `Build.VERSION_CODES.TIRAMISU` (/reference/android/os/Build.VERSION_CODES#TIRAMISU): Only device owners can update APNs.

Starting from Android version `Build.VERSION_CODES.TIRAMISU` (/reference/android/os/Build.VERSION_CODES#TIRAMISU): Both device owners and managed profile owners can update enterprise APNs (`ApnSetting#TYPE_ENTERPRISE` (/reference/android/telephony/data/ApnSetting#TYPE_ENTERPRISE)), while only device owners can update other type of APNs.

Parameters

`admin`	`ComponentName`: which `DeviceAdminReceiver` (/reference/android/app/admin/DeviceAdminReceiver) this request is associated with This value cannot be `null`.
`apnId`	`int`: the `id` of the override APN to update
`apnSetting`	`ApnSetting`: the override APN to update This value cannot be `null`.

Returns

`boolean`	`true` if the required override APN is successfully updated, `false` otherwise.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	If request is for enterprise APN `admin` is either device owner or profile owner and in all other types of APN if `admin` is not a device owner.

See also:

`setOverrideApnsEnabled(ComponentName, boolean)` (/reference/android/app/admin/DevicePolicyManager#setOverrideApnsEnabled(android.content.ComponentName,%20boolean))

wipeData

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void wipeData (int flags,
                CharSequence (/reference/java/lang/CharSequence) reason)
```

Ask that all user data be wiped.

If called as a secondary user or managed profile, the user itself and its associated user data will be wiped. In particular, If the caller is a profile owner of an organization-owned managed profile, calling this method will relinquish the device for personal use, removing the managed profile and all policies set by the profile owner.

Calling this method from the primary user will only work if the calling app is targeting SDK level `Build.VERSION_CODES#TIRAMISU` (/reference/android/os/Build.VERSION_CODES#TIRAMISU) or below, in which case it will cause the device to reboot, erasing all device data - including all the secondary users and their data - while booting up. If an app targeting SDK level `Build.VERSION_CODES#UPSIDE_DOWN_CAKE` (/reference/android/os/Build.VERSION_CODES#UPSIDE_DOWN_CAKE) and above is calling this method from the primary user or last full user, `IllegalStateException` (/reference/java/lang/IllegalStateException) will be thrown.

If an app wants to wipe the entire device irrespective of which user they are from, they should use `wipeDevice(int)` (/reference/android/app/admin/DevicePolicyManager#wipeDevice(int)) instead.

Parameters

`flags`	`int`: Bit mask of additional options: currently supported flags are `WIPE_EXTERNAL_STORAGE` (/reference/android/app/admin/DevicePolicyManager#WIPE_EXTERNAL_ST `WIPE_RESET_PROTECTION_DATA` (/reference/android/app/admin/DevicePolicyManager#WIPE_RESET_PROTE and `WIPE_EUICC` (/reference/android/app/admin/DevicePolicyManager#WI
`reason`	`CharSequence`: a string that contains the reason for wiping data, which ca presented to the user. This value cannot be `null`.

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the calling application does not own an active administrato `DeviceAdminInfo#USES_POLICY_WIPE_DATA` (/reference/android/app/admin/DeviceAdminInfo#USES_POI

	is not granted the Manifest.permission.MASTER_CLEAR (/reference/android/Manifest.permission#MASTER_CLEAR) permission.MANAGE_DEVICE_POLICY_WIPE_DATA (/reference/android/Manifest.permission#MANAGE_DEVICE permissions.
IllegalArgumentException (/reference/java/lang/IllegalArgumentException)	if the input reason string is null or empty, or if WIPE_SILENT (/reference/android/app/admin/DevicePolicyManager#WIPE
IllegalStateException (/reference/java/lang/IllegalStateException)	if called on last full-user or system-user

See also:

wipeDevice(int) (/reference/android/app/admin/DevicePolicyManager#wipeDevice(int))

wipeData(int) (/reference/android/app/admin/DevicePolicyManager#wipeData(int))

wipeData

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void wipeData (int flags)
```

See wipeData(int, java.lang.CharSequence) (/reference/android/app/admin/DevicePolicyManager#wipeData(int,%20java.lang.CharSequence))

Parameters

flags	int: Bit mask of additional options: currently supported flags are WIPE_EXTERNAL_STORAGE (/reference/android/app/admin/DevicePolicyManager#WIPE_EXTERNAL_ST WIPE_RESET_PROTECTION_DATA (/reference/android/app/admin/DevicePolicyManager#WIPE_RESET_PROTE , WIPE_EUICC (/reference/android/app/admin/DevicePolicyManager#WIPE_ WIPE_SILENTLY (/reference/android/app/admin/DevicePolicyManager#WI .

Throws

SecurityException (/reference/java/lang/SecurityException)	if the calling application does not own an active administrator tha DeviceAdminInfo#USES_POLICY_WIPE_DATA (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_ is not granted the Manifest.permission.MASTER_CLEAR (/reference/android/Manifest.permission#MASTER_CLEAR) or Ma permission.MANAGE_DEVICE_POLICY_WIPE_DATA (/reference/android/Manifest.permission#MANAGE_DEVICE_POI permissions.
IllegalStateException (/reference/java/lang/IllegalStateException)	if called on last full-user or system-user

See also:

`wipeDevice(int)` (/reference/android/app/admin/DevicePolicyManager#wipeDevice(int))

`wipeData(int, CharSequence)` (/reference/android/app/admin/DevicePolicyManager#wipeData(int,%20java.lang.CharSequence))

wipeDevice

Added in API level 34 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void wipeDevice (int flags)
```

Ask that the device be wiped and factory reset.

The calling Device Owner or Organization Owned Profile Owner must have requested `DeviceAdminInfo#USES_POLICY_WIPE_DATA` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_WIPE_DATA) to be able to call this method; if it has not, a security exception will be thrown.

Parameters

`flags`	`int`: Bit mask of additional options: currently supported flags are `WIPE_EXTERNAL_STORAGE` (/reference/android/app/admin/DevicePolicyManager#WIPE_EXTERNAL_ST `WIPE_RESET_PROTECTION_DATA` (/reference/android/app/admin/DevicePolicyManager#WIPE_RESET_PROTE , `WIPE_EUICC` (/reference/android/app/admin/DevicePolicyManager#WIPE_ `WIPE_SILENTLY` (/reference/android/app/admin/DevicePolicyManager#WI .

Throws

`SecurityException` (/reference/java/lang/SecurityException)	if the calling application does not own an active administrator that u `DeviceAdminInfo#USES_POLICY_WIPE_DATA` (/reference/android/app/admin/DeviceAdminInfo#USES_POLICY_WI not granted the `Manifest.permission.MASTER_CLEAR` (/reference/android/Manifest.permission#MASTER_CLEAR) or both `permission.MANAGE_DEVICE_POLICY_WIPE_DATA` (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC `Manifest.permission.MANAGE_DEVICE_POLICY_ACROSS_USE (/reference/android/Manifest.permission#MANAGE_DEVICE_POLIC permissions.

See also:

`wipeData(int)` (/reference/android/app/admin/DevicePolicyManager#wipeData(int))

`wipeData(int, CharSequence)` (/reference/android/app/admin/DevicePolicyManager#wipeData(int,%20java.lang.CharSequence))