AccountManager | API reference | Android Developers

Archived: 2026-04-05 17:26:22 UTC

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

```
public class AccountManager
extends Object
```

This class provides access to a centralized registry of the user's online accounts. The user enters credentials (username and password) once per account, granting applications access to online resources with "one-click" approval.

Different online services have different ways of handling accounts and authentication, so the account manager uses pluggable *authenticator* modules for different *account types*. Authenticators (which may be written by third parties) handle the actual details of validating account credentials and storing account information. For example, Google, Facebook, and Microsoft Exchange each have their own authenticator.

Many servers support some notion of an *authentication token*, which can be used to authenticate a request to the server without sending the user's actual password. (Auth tokens are normally created with a separate request which does include the user's credentials.) AccountManager can generate auth tokens for applications, so the application doesn't need to handle passwords directly. Auth tokens are normally reusable and cached by AccountManager, but must be refreshed periodically. It's the responsibility of applications to *invalidate* auth tokens when they stop working so the AccountManager knows it needs to regenerate them.

Applications accessing a server normally go through these steps:

- Get an instance of AccountManager using `get(Context)`.
- List the available accounts using `getAccountsByType(String)` or `getAccountsByTypeAndFeatures(String, String, AccountManagerCallback, Handler)`. Normally applications will only be interested in accounts with one particular *type*, which identifies the authenticator. Account *features* are used to identify particular account subtypes and capabilities. Both the account type and features are authenticator-specific strings, and must be known by the application in coordination with its preferred authenticators.
- Select one or more of the available accounts, possibly by asking the user for their preference. If no suitable accounts are available, `addAccount(String, String, String, Bundle, Activity, AccountManagerCallback, Handler)` may be called to prompt the user to create an account of the appropriate type.
- **Important:** If the application is using a previously remembered account selection, it must make sure the account is still in the list of accounts returned by `getAccountsByType(String)`. Requesting an auth token for an account no longer on the device results in an undefined failure.
- Request an auth token for the selected account(s) using one of the `getAuthToken(Account, String, Bundle, Activity, AccountManagerCallback, Handler)` methods or related helpers. Refer to the description of each method for exact usage and error handling details.
- Make the request using the auth token. The form of the auth token, the format of the request, and the protocol used are all specific to the service you are accessing. The application may use whatever network and protocol libraries are useful.
- **Important:** If the request fails with an authentication error, it could be that a cached auth token is stale and no longer honored by the server. The application must call `invalidateAuthToken(String, String)` to remove the token from the cache, otherwise requests will continue failing! After invalidating the auth token, immediately go back to the "Request an auth token" step above. If the process fails the second time, then it can be treated as a "genuine" authentication failure and the user notified or other appropriate actions taken.

Some AccountManager methods may need to interact with the user to prompt for credentials, present options, or ask the user to add an account. The caller may choose whether to allow AccountManager to directly launch the necessary user interface

and wait for the user, or to return an Intent which the caller may use to launch the interface, or (in some cases) to install a notification which the user can select at any time to launch the interface. To have AccountManager launch the interface directly, the caller must supply the current foreground `Activity` context.

Many AccountManager methods take `AccountManagerCallback` and `Handler` as parameters. These methods return immediately and run asynchronously. If a callback is provided then `AccountManagerCallback.run` will be invoked on the Handler's thread when the request completes, successfully or not. The result is retrieved by calling `AccountManagerFuture.getResult()` on the `AccountManagerFuture` returned by the method (and also passed to the callback). This method waits for the operation to complete (if necessary) and either returns the result or throws an exception if an error occurred during the operation. To make the request synchronously, call `AccountManagerFuture.getResult()` immediately on receiving the future from the method; no callback need be supplied.

Requests which may block, including `AccountManagerFuture.getResult()`, must never be called on the application's main event thread. These operations throw `IllegalStateException` if they are used on the main thread.

Summary

Constants	
`String`	`ACTION_ACCOUNT_REMOVED` Action sent as a broadcast Intent by the AccountsService when any account is removed or renamed.
`String`	`ACTION_AUTHENTICATOR_INTENT`
`String`	`AUTHENTICATOR_ATTRIBUTES_NAME`
`String`	`AUTHENTICATOR_META_DATA_NAME`
`int`	`ERROR_CODE_BAD_ARGUMENTS`
`int`	`ERROR_CODE_BAD_AUTHENTICATION`
`int`	`ERROR_CODE_BAD_REQUEST`
`int`	`ERROR_CODE_CANCELED`
`int`	`ERROR_CODE_INVALID_RESPONSE`
`int`	`ERROR_CODE_NETWORK_ERROR`
`int`	`ERROR_CODE_REMOTE_EXCEPTION`
`int`	`ERROR_CODE_UNSUPPORTED_OPERATION`
`String`	`KEY_ACCOUNTS`
`String`	`KEY_ACCOUNT_AUTHENTICATOR_RESPONSE`
`String`	`KEY_ACCOUNT_MANAGER_RESPONSE`
`String`	`KEY_ACCOUNT_NAME` Bundle key used for the `String` account name in results from methods which return information about a particular account.

String	KEY_ACCOUNT_SESSION_BUNDLE Bundle key used for a Bundle in result from startAddAccountSession(String, String, String, Bundle, Activity, AccountManagerCallback, Handler) and friends which returns session data for installing an account later.
String	KEY_ACCOUNT_STATUS_TOKEN Bundle key used for the String account status token in result from startAddAccountSession(String, String, String, Bundle, Activity, AccountManagerCallback, Handler) and friends which returns information about a particular account.
String	KEY_ACCOUNT_TYPE Bundle key used for the String account type in results from methods which return information about a particular account.
String	KEY_ANDROID_PACKAGE_NAME The Android package of the caller will be set in the options bundle by the AccountManager and will be passed to the AccountManagerService and to the AccountAuthenticators.
String	KEY_AUTHENTICATOR_TYPES
String	KEY_AUTHTOKEN Bundle key used for the auth token value in results from getAuthToken(Account, String, Bundle, Activity, AccountManagerCallback, Handler) and friends.
String	KEY_AUTH_FAILED_MESSAGE
String	KEY_AUTH_TOKEN_LABEL
String	KEY_BOOLEAN_RESULT
String	KEY_CALLER_PID The process id of caller app.
String	KEY_CALLER_UID The UID of caller app.
String	KEY_ERROR_CODE
String	KEY_ERROR_MESSAGE
String	KEY_INTENT Bundle key used for an Intent in results from methods that may require the caller to interact with the user.
String	KEY_LAST_AUTHENTICATED_TIME Bundle key used to supply the last time the credentials of the account were authenticated successfully.
String	KEY_PASSWORD

	Bundle key used to supply the password directly in options to `confirmCredentials(Account, Bundle, Activity, AccountManagerCallback, Handler)` , rather than prompting the user with the standard password prompt.
`String`	`KEY_USERDATA`
`String`	`LOGIN_ACCOUNTS_CHANGED_ACTION` *This constant was deprecated in API level 26. use* `addOnAccountsUpdatedListener(OnAccountsUpdateListener, Handler, boolean)` *to get account updates in runtime.*
`String`	`PACKAGE_NAME_KEY_LEGACY_NOT_VISIBLE` Key to set default visibility for applications which don't satisfy conditions in `PACKAGE_NAME_KEY_LEGACY_VISIBLE` .
`String`	`PACKAGE_NAME_KEY_LEGACY_VISIBLE` Key to set visibility for applications which satisfy one of the following conditions: • Target API level below `Build.VERSION_CODES.O` and have deprecated `Manifest.permission.GET_ACCOUNTS` permission.
`int`	`VISIBILITY_NOT_VISIBLE` Account is not visible to given application and only authenticator can grant visibility.
`int`	`VISIBILITY_UNDEFINED` Account visibility was not set.
`int`	`VISIBILITY_USER_MANAGED_NOT_VISIBLE` Account is not visible to given application, but user can reveal it, for example, using `newChooseAccountIntent(Account,List,String[],String,String,String[],Bundle)`
`int`	`VISIBILITY_USER_MANAGED_VISIBLE` Account is visible to given application, but user can revoke visibility.
`int`	`VISIBILITY_VISIBLE` Account is always visible to given application and only authenticator can revoke visibility.

Public methods

`AccountManagerFuture<Bundle>`	`addAccount(String accountType, String authTokenType, String[] requiredFeatures, Bundle addAccountOptions, Activity activity, AccountManagerCallback<Bundle> callback, Handler h` Asks the user to add an account of a specified type.
`boolean`	`addAccountExplicitly(Account account, String password, Bundle userdata)` Adds an account directly to the AccountManager.

`boolean`	`addAccountExplicitly(Account account, String password, Bundle extras, Map<String, Integ visibility)` Adds an account directly to the AccountManager.
`void`	`addOnAccountsUpdatedListener(OnAccountsUpdateListener listener, Handler handler, boolea updateImmediately, String[] accountTypes)` Adds an `OnAccountsUpdateListener` to this instance of the `AccountManager` .
`void`	`addOnAccountsUpdatedListener(OnAccountsUpdateListener listener, Handler handler, boolea updateImmediately)` Adds an `OnAccountsUpdateListener` to this instance of the `AccountManager` .
`String`	`blockingGetAuthToken(Account account, String authTokenType, boolean notifyAuthFailure)` This convenience helper synchronously gets an auth token with `getAuthToken(Account,String,boolean,AccountManagerCallback,Handler)` .
`void`	`clearPassword(Account account)` Forgets a saved password.
`AccountManagerFuture<Bundle>`	`confirmCredentials(Account account, Bundle options, Activity activity, AccountManagerCallback<Bundle> callback, Handler handler)` Confirms that the user knows the password for an account to make extra sure they are the owner of account.
`AccountManagerFuture<Bundle>`	`editProperties(String accountType, Activity activity, AccountManagerCallback<Bundle> ca Handler handler)` Offers the user an opportunity to change an authenticator's settings.
`AccountManagerFuture<Bundle>`	`finishSession(Bundle sessionBundle, Activity activity, AccountManagerCallback<Bundle> c Handler handler)` Finishes the session started by `startAddAccountSession(String, String, String, Bundle, Activ AccountManagerCallback, Handler)` or `startUpdateCredentialsSession(Account, String, Bund Activity, AccountManagerCallback, Handler)` .
`static AccountManager`	`get(Context context)` Gets an AccountManager instance associated with a Context.
`int`	`getAccountVisibility(Account account, String packageName)` Get visibility of certain account for given application.
`Account[]`	`getAccounts()` Lists all accounts visible to the caller regardless of type.

`Map<Account, Integer>`	`getAccountsAndVisibilityForPackage(String packageName, String accountType)` Gets all accounts of given type and their visibility for specific package.
`Account[]`	`getAccountsByType(String type)` Lists all accounts of particular type visible to the caller.
`AccountManagerFuture<Account[]>`	`getAccountsByTypeAndFeatures(String type, String[] features, AccountManagerCallback<Acc callback, Handler handler)` Lists all accounts of a type which have certain features.
`Account[]`	`getAccountsByTypeForPackage(String type, String packageName)` Returns the accounts visible to the specified package in an environment where some apps are not a to view all accounts.
`AccountManagerFuture<Bundle>`	`getAuthToken(Account account, String authTokenType, Bundle options, boolean notifyAuthF AccountManagerCallback<Bundle> callback, Handler handler)` Gets an auth token of the specified type for a particular account, optionally raising a notification if must enter credentials.
`AccountManagerFuture<Bundle>`	`getAuthToken(Account account, String authTokenType, Bundle options, Activity activity, AccountManagerCallback<Bundle> callback, Handler handler)` Gets an auth token of the specified type for a particular account, prompting the user for credentials necessary.
`AccountManagerFuture<Bundle>`	`getAuthToken(Account account, String authTokenType, boolean notifyAuthFailure, AccountManagerCallback<Bundle> callback, Handler handler)` *This method was deprecated in API level 15. use* *`getAuthToken(Account,String,android.os.Bundle,boolean,AccountManagerCallback,android.os`* *instead*
`AccountManagerFuture<Bundle>`	`getAuthTokenByFeatures(String accountType, String authTokenType, String[] features, Act activity, Bundle addAccountOptions, Bundle getAuthTokenOptions, AccountManagerCallback<B callback, Handler handler)` This convenience helper combines the functionality of `getAccountsByTypeAndFeatures(String, S AccountManagerCallback, Handler)` , `getAuthToken(Account, String, Bundle, Activity, AccountManagerCallback, Handler)` , and `addAccount(String, String, String, Bundle, Activi AccountManagerCallback, Handler)` .
`AuthenticatorDescription[]`	`getAuthenticatorTypes()` Lists the currently registered authenticators.
`Map<String, Integer>`	`getPackagesAndVisibilityForAccount(Account account)` Returns package names and visibility which were explicitly set for given account.

String	getPassword(Account account) Gets the saved password associated with the account.
String	getPreviousName(Account account) Gets the previous name associated with the account or null, if none.
String	getUserData(Account account, String key) Gets the user data named by "key" associated with the account.
AccountManagerFuture<Boolean>	hasFeatures(Account account, String[] features, AccountManagerCallback<Boolean> callbac Handler handler) Finds out whether a particular account has all the specified features.
void	invalidateAuthToken(String accountType, String authToken) Removes an auth token from the AccountManager's cache.
AccountManagerFuture<Boolean>	isCredentialsUpdateSuggested(Account account, String statusToken, AccountManagerCallback<Boolean> callback, Handler handler) Checks whether updateCredentials(Account, String, Bundle, Activity, AccountManagerCallb Handler) or startUpdateCredentialsSession(Account, String, Bundle, Activity, AccountManagerCallback, Handler) should be called with respect to the specified account.
static Intent	newChooseAccountIntent(Account selectedAccount, List<Account> allowableAccounts, String allowableAccountTypes, String descriptionOverrideText, String addAccountAuthTokenType, S addAccountRequiredFeatures, Bundle addAccountOptions) Returns an intent to an Activity that prompts the user to choose from a list of accounts.
static Intent	newChooseAccountIntent(Account selectedAccount, ArrayList<Account> allowableAccounts, S allowableAccountTypes, boolean alwaysPromptForAccount, String descriptionOverrideText, S addAccountAuthTokenType, String[] addAccountRequiredFeatures, Bundle addAccountOptions) Deprecated in favor of newChooseAccountIntent(Account,List,String[],String,String,String[],Bundle).
boolean	notifyAccountAuthenticated(Account account) Notifies the system that the account has just been authenticated.
String	peekAuthToken(Account account, String authTokenType) Gets an auth token from the AccountManager's cache.
AccountManagerFuture<Bundle>	removeAccount(Account account, Activity activity, AccountManagerCallback<Bundle> callba Handler handler) Removes an account from the AccountManager.
AccountManagerFuture<Boolean>	removeAccount(Account account, AccountManagerCallback<Boolean> callback, Handler handle

	This method was deprecated in API level 22. use `removeAccount(Account,Activity,AccountManagerCallback,Handler)` *instead*
`boolean`	`removeAccountExplicitly(Account account)` Removes an account directly.
`void`	`removeOnAccountsUpdatedListener(OnAccountsUpdateListener listener)` Removes an `OnAccountsUpdateListener` previously registered with `addOnAccountsUpdatedListener(OnAccountsUpdateListener, Handler, boolean)` .
`AccountManagerFuture<Account>`	`renameAccount(Account account, String newName, AccountManagerCallback<Account> callback handler)` Rename the specified `Account` .
`boolean`	`setAccountVisibility(Account account, String packageName, int visibility)` Set visibility value of given account to certain package.
`void`	`setAuthToken(Account account, String authTokenType, String authToken)` Adds an auth token to the AccountManager cache for an account.
`void`	`setPassword(Account account, String password)` Sets or forgets a saved password.
`void`	`setUserData(Account account, String key, String value)` Sets one userdata key for an account.
`AccountManagerFuture<Bundle>`	`startAddAccountSession(String accountType, String authTokenType, String[] requiredFeatu Bundle options, Activity activity, AccountManagerCallback<Bundle> callback, Handler hand` Asks the user to authenticate with an account of a specified type.
`AccountManagerFuture<Bundle>`	`startUpdateCredentialsSession(Account account, String authTokenType, Bundle options, Ac activity, AccountManagerCallback<Bundle> callback, Handler handler)` Asks the user to enter a new password for the account but not updating the saved credentials for the until `finishSession(Bundle, Activity, AccountManagerCallback, Handler)` is called.
`AccountManagerFuture<Bundle>`	`updateCredentials(Account account, String authTokenType, Bundle options, Activity activ AccountManagerCallback<Bundle> callback, Handler handler)` Asks the user to enter a new password for an account, updating the saved credentials for the accour

Inherited methods

From class `java.lang.Object`

`Object`	`clone()` Creates and returns a copy of this object.
`boolean`	`equals(Object obj)` Indicates whether some other object is "equal to" this one.
`void`	`finalize()` Called by the garbage collector on an object when garbage collection determines that there are no more references to the object.
`final Class<?>`	`getClass()` Returns the runtime class of this `Object`.
`int`	`hashCode()` Returns a hash code value for the object.
`final void`	`notify()` Wakes up a single thread that is waiting on this object's monitor.
`final void`	`notifyAll()` Wakes up all threads that are waiting on this object's monitor.
`String`	`toString()` Returns a string representation of the object.
`final void`	`wait(long timeoutMillis, int nanos)` Causes the current thread to wait until it is awakened, typically by being *notified* or *interrupted*, or until a certain amount of real time has elapsed.
`final void`	`wait(long timeoutMillis)` Causes the current thread to wait until it is awakened, typically by being *notified* or *interrupted*, or until a certain amount of real time has elapsed.
`final void`	`wait()` Causes the current thread to wait until it is awakened, typically by being *notified* or *interrupted*.

Constants

ACTION_ACCOUNT_REMOVED

```
public static final String ACTION_ACCOUNT_REMOVED
```

Action sent as a broadcast Intent by the AccountsService when any account is removed or renamed. Only applications which were able to see the account will receive the intent. Intent extra will include the following fields:

- `KEY_ACCOUNT_NAME` - the name of the removed account
- `KEY_ACCOUNT_TYPE` - the type of the account

Constant Value: "android.accounts.action.ACCOUNT_REMOVED"

ACTION_AUTHENTICATOR_INTENT

```
public static final String ACTION_AUTHENTICATOR_INTENT
```

Constant Value: "android.accounts.AccountAuthenticator"

AUTHENTICATOR_ATTRIBUTES_NAME

```
public static final String AUTHENTICATOR_ATTRIBUTES_NAME
```

Constant Value: "account-authenticator"

AUTHENTICATOR_META_DATA_NAME

```
public static final String AUTHENTICATOR_META_DATA_NAME
```

Constant Value: "android.accounts.AccountAuthenticator"

ERROR_CODE_BAD_ARGUMENTS

```
public static final int ERROR_CODE_BAD_ARGUMENTS
```

Constant Value: 7 (0x00000007)

ERROR_CODE_BAD_AUTHENTICATION

```
public static final int ERROR_CODE_BAD_AUTHENTICATION
```

Constant Value: 9 (0x00000009)

ERROR_CODE_BAD_REQUEST

```
public static final int ERROR_CODE_BAD_REQUEST
```

Constant Value: 8 (0x00000008)

ERROR_CODE_CANCELED

```
public static final int ERROR_CODE_CANCELED
```

Constant Value: 4 (0x00000004)

ERROR_CODE_INVALID_RESPONSE

```
public static final int ERROR_CODE_INVALID_RESPONSE
```

Constant Value: 5 (0x00000005)

ERROR_CODE_NETWORK_ERROR

```
public static final int ERROR_CODE_NETWORK_ERROR
```

Constant Value: 3 (0x00000003)

ERROR_CODE_REMOTE_EXCEPTION

```
public static final int ERROR_CODE_REMOTE_EXCEPTION
```

Constant Value: 1 (0x00000001)

ERROR_CODE_UNSUPPORTED_OPERATION

```
public static final int ERROR_CODE_UNSUPPORTED_OPERATION
```

Constant Value: 6 (0x00000006)

KEY_ACCOUNTS

```
public static final String KEY_ACCOUNTS
```

Constant Value: "accounts"

KEY_ACCOUNT_AUTHENTICATOR_RESPONSE

```
public static final String KEY_ACCOUNT_AUTHENTICATOR_RESPONSE
```

Constant Value: "accountAuthenticatorResponse"

KEY_ACCOUNT_MANAGER_RESPONSE

```
public static final String KEY_ACCOUNT_MANAGER_RESPONSE
```

Constant Value: "accountManagerResponse"

KEY_ACCOUNT_NAME

```
public static final String KEY_ACCOUNT_NAME
```

Bundle key used for the `String` account name in results from methods which return information about a particular account.

Constant Value: "authAccount"

KEY_ACCOUNT_TYPE

```
public static final String KEY_ACCOUNT_TYPE
```

Bundle key used for the `String` account type in results from methods which return information about a particular account.

Constant Value: "accountType"

KEY_ANDROID_PACKAGE_NAME

```
public static final String KEY_ANDROID_PACKAGE_NAME
```

The Android package of the caller will be set in the options bundle by the `AccountManager` and will be passed to the AccountManagerService and to the AccountAuthenticators. The uid of the caller will be known by the AccountManagerService as well as the AccountAuthenticators so they will be able to verify that the package is consistent with the uid (a uid might be shared by many packages).

Constant Value: "androidPackageName"

KEY_AUTHENTICATOR_TYPES

```
public static final String KEY_AUTHENTICATOR_TYPES
```

Constant Value: "authenticator_types"

KEY_AUTH_FAILED_MESSAGE

```
public static final String KEY_AUTH_FAILED_MESSAGE
```

Constant Value: "authFailedMessage"

KEY_AUTH_TOKEN_LABEL

```
public static final String KEY_AUTH_TOKEN_LABEL
```

Constant Value: "authTokenLabelKey"

KEY_BOOLEAN_RESULT

```
public static final String KEY_BOOLEAN_RESULT
```

Constant Value: "booleanResult"

KEY_CALLER_PID

```
public static final String KEY_CALLER_PID
```

The process id of caller app.

Constant Value: "callerPid"

KEY_CALLER_UID

```
public static final String KEY_CALLER_UID
```

The UID of caller app.

Constant Value: "callerUid"

KEY_ERROR_CODE

```
public static final String KEY_ERROR_CODE
```

Constant Value: "errorCode"

KEY_ERROR_MESSAGE

```
public static final String KEY_ERROR_MESSAGE
```

Constant Value: "errorMessage"

KEY_INTENT

```
public static final String KEY_INTENT
```

Bundle key used for an `Intent` in results from methods that may require the caller to interact with the user. The Intent can be used to start the corresponding user interface activity.

Constant Value: "intent"

KEY_LAST_AUTHENTICATED_TIME

```
public static final String KEY_LAST_AUTHENTICATED_TIME
```

Bundle key used to supply the last time the credentials of the account were authenticated successfully. Time is specified in milliseconds since epoch. Associated time is updated on successful authentication of account on adding account, confirming credentials, or updating credentials.

Constant Value: "lastAuthenticatedTime"

KEY_USERDATA

```
public static final String KEY_USERDATA
```

Constant Value: "userdata"

VISIBILITY_NOT_VISIBLE

```
public static final int VISIBILITY_NOT_VISIBLE
```

Account is not visible to given application and only authenticator can grant visibility.

Constant Value: 3 (0x00000003)

VISIBILITY_USER_MANAGED_VISIBLE

```
public static final int VISIBILITY_USER_MANAGED_VISIBLE
```

Account is visible to given application, but user can revoke visibility.

Constant Value: 2 (0x00000002)

VISIBILITY_VISIBLE

```
public static final int VISIBILITY_VISIBLE
```

Account is always visible to given application and only authenticator can revoke visibility.

Constant Value: 1 (0x00000001)

Public methods

addAccount

```
public AccountManagerFuture<Bundle> addAccount (String accountType,
                String authTokenType,
                String[] requiredFeatures,
                Bundle addAccountOptions,
                Activity activity,
                AccountManagerCallback<Bundle> callback,
                Handler handler)
```

Asks the user to add an account of a specified type. The authenticator for this account type processes this request with the appropriate user interface. If the user does elect to create a new account, the account name is returned.

This method may be called from any thread, but the returned `AccountManagerFuture` must not be used on the main thread.

NOTE: If targeting your app to work on API level 22 and before, MANAGE_ACCOUNTS permission is needed for those platforms. See docs for this function in API level 22.
Requires android.Manifest.permission.MANAGE_ACCOUNTS

Parameters	
`accountType`	`String` : The type of account to add; must not be null
`authTokenType`	`String` : The type of auth token (see `getAuthToken(Account, String, Bundle, Activity, AccountManagerCallback, Handler)`) this account will need to be able to generate, null for none
`requiredFeatures`	`String` : The features (see `hasFeatures(Account, String, AccountManagerCallback, Handler)`) this account must have, null for none
`addAccountOptions`	`Bundle` : Authenticator-specific options for the request, may be null or empty
`activity`	`Activity` : The `Activity` context to use for launching a new authenticator-defined sub-Activity to prompt the user to create an account; used only to call startActivity(); if null, the prompt will not be launched directly, but the necessary `Intent` will be returned to the caller instead
`callback`	`AccountManagerCallback` : Callback to invoke when the request completes, null for no callback
`handler`	`Handler` : `Handler` identifying the callback thread, null for the main thread

addAccountExplicitly

```
public boolean addAccountExplicitly (Account account,
                String password,
                Bundle userdata)
```

Adds an account directly to the AccountManager. Normally used by sign-up wizards associated with authenticators, not directly by applications.

Calling this method does not update the last authenticated timestamp, referred by `KEY_LAST_AUTHENTICATED_TIME` . To update it, call `notifyAccountAuthenticated(Account)` after getting success. However, if this method is called when it is triggered by addAccount() or addAccountAsUser() or similar functions, then there is no need to update timestamp manually as it is updated automatically by framework on successful completion of the mentioned functions.

It is safe to call this method from the main thread.

This method requires the caller to have a signature match with the authenticator that owns the specified account.

NOTE: If targeting your app to work on API level 22 and before, AUTHENTICATE_ACCOUNTS permission is needed for those platforms. See docs for this function in API level 22.
Requires android.Manifest.permission.AUTHENTICATE_ACCOUNTS

Parameters	
`account`	`Account` : The `Account` to add
`password`	`String` : The password to associate with the account, null for none
`userdata`	`Bundle` : String values to use for the account's userdata, null for none
Returns	
`boolean`	True if the account was successfully added, false if the account already exists, the account is null, the user is locked, or another error occurs.

addAccountExplicitly

```
public boolean addAccountExplicitly (Account account,
                String password,
                Bundle extras,
                Map<String, Integer> visibility)
```

Adds an account directly to the AccountManager. Additionally it specifies Account visibility for given list of packages.

Normally used by sign-up wizards associated with authenticators, not directly by applications.

Calling this method does not update the last authenticated timestamp, referred by `KEY_LAST_AUTHENTICATED_TIME`. To update it, call `notifyAccountAuthenticated(Account)` after getting success.

It is safe to call this method from the main thread.

This method requires the caller to have a signature match with the authenticator that owns the specified account.

Parameters	
`account`	`Account` : The `Account` to add
`password`	`String` : The password to associate with the account, null for none
`extras`	`Bundle` : String values to use for the account's userdata, null for none
`visibility`	`Map` : Map from packageName to visibility values which will be set before account is added. See `getAccountVisibility(Account, String)` for possible values. Declaring package visibility needs for package names in the map is needed, if the caller is targeting API level 34 and above.
Returns	
`boolean`	True if the account was successfully added, false if the account already exists, the account is null, or another error occurs.

addOnAccountsUpdatedListener

```
public void addOnAccountsUpdatedListener (OnAccountsUpdateListener listener,
                Handler handler,
                boolean updateImmediately,
                String[] accountTypes)
```

Adds an `OnAccountsUpdateListener` to this instance of the `AccountManager`. This listener will be notified whenever user or AbstractAccountAuthenticator made changes to accounts of given types related to the caller - either list of accounts

returned by `getAccounts()` was changed, or new account was added for which user can grant access to the caller.

As long as this listener is present, the AccountManager instance will not be garbage-collected, and neither will the `Context` used to retrieve it, which may be a large Activity instance. To avoid memory leaks, you must remove this listener before then. Normally listeners are added in an Activity or Service's `Activity.onCreate` and removed in `Activity.onDestroy`.

It is safe to call this method from the main thread.

Parameters	
`listener`	`OnAccountsUpdateListener` : The listener to send notifications to
`handler`	`Handler` : `Handler` identifying the thread to use for notifications, null for the main thread
`updateImmediately`	`boolean` : If true, the listener will be invoked (on the handler thread) right away with the current account list
`accountTypes`	`String` : If set, only changes to accounts of given types will be reported.

addOnAccountsUpdatedListener

```
public void addOnAccountsUpdatedListener (OnAccountsUpdateListener listener,
                Handler handler,
                boolean updateImmediately)
```

Adds an `OnAccountsUpdateListener` to this instance of the `AccountManager`. This listener will be notified whenever user or AbstractAccountAuthenticator made changes to accounts of any type related to the caller. This method is equivalent to addOnAccountsUpdatedListener(listener, handler, updateImmediately, null).
Requires `Manifest.permission.GET_ACCOUNTS`

Parameters	
`listener`	`OnAccountsUpdateListener`
`handler`	`Handler`
`updateImmediately`	`boolean`

blockingGetAuthToken

```
public String blockingGetAuthToken (Account account,
                String authTokenType,
                boolean notifyAuthFailure)
```

This convenience helper synchronously gets an auth token with `getAuthToken(Account,String,boolean,AccountManagerCallback,Handler)`.

This method may block while a network request completes, and must never be made from the main thread.

NOTE: If targeting your app to work on API level 22 and before, USE_CREDENTIALS permission is needed for those platforms. See docs for this function in API level 22.
Requires android.Manifest.permission.USE_CREDENTIALS

Parameters	
`account`	`Account` : The account to fetch an auth token for
`authTokenType`	`String` : The auth token type, see `getAuthToken()`

`notifyAuthFailure`	`boolean` : If true, display a notification and return null if authentication fails; if false, prompt and wait for the user to re-enter correct credentials before returning

Returns	
`String`	An auth token of the specified type for this account, or null if authentication fails or none can be fetched.

Throws	
`AuthenticatorException`	if the authenticator failed to respond
`OperationCanceledException`	if the request was canceled for any reason, including the user canceling a credential request
`IOException`	if the authenticator experienced an I/O problem creating a new auth token, usually because of network trouble

clearPassword

```
public void clearPassword (Account account)
```

Forgets a saved password. This erases the local copy of the password; it does not change the user's account password on the server. Has the same effect as setPassword(account, null) but requires fewer permissions, and may be used by applications or management interfaces to "sign out" from an account.

This method only successfully clear the account's password when the caller has the same signature as the authenticator that owns the specified account. Otherwise, this method will silently fail.

It is safe to call this method from the main thread.

NOTE: If targeting your app to work on API level 22 and before, MANAGE_ACCOUNTS permission is needed for those platforms. See docs for this function in API level 22.
Requires android.Manifest.permission.MANAGE_ACCOUNTS

Parameters	
`account`	`Account` : The account whose password to clear

confirmCredentials

```
public AccountManagerFuture<Bundle> confirmCredentials (Account account,
                Bundle options,
                Activity activity,
                AccountManagerCallback<Bundle> callback,
                Handler handler)
```

Confirms that the user knows the password for an account to make extra sure they are the owner of the account. The user-entered password can be supplied directly, otherwise the authenticator for this account type prompts the user with the appropriate interface. This method is intended for applications which want extra assurance; for example, the phone lock screen uses this to let the user unlock the phone with an account password if they forget the lock pattern.

If the user-entered password matches a saved password for this account, the request is considered valid; otherwise the authenticator verifies the password (usually by contacting the server).

This method may be called from any thread, but the returned `AccountManagerFuture` must not be used on the main thread.

NOTE: If targeting your app to work on API level 22 and before, MANAGE_ACCOUNTS permission is needed for those platforms. See docs for this function in API level 22.
Requires android.Manifest.permission.MANAGE_ACCOUNTS

Parameters	
`account`	`Account` : The account to confirm password knowledge for
`options`	`Bundle` : Authenticator-specific options for the request; if the `KEY_PASSWORD` string field is present, the authenticator may use it directly rather than prompting the user; may be null or empty
`activity`	`Activity` : The `Activity` context to use for launching a new authenticator-defined sub-Activity to prompt the user to enter a password; used only to call startActivity(); if null, the prompt will not be launched directly, but the necessary `Intent` will be returned to the caller instead
`callback`	`AccountManagerCallback` : Callback to invoke when the request completes, null for no callback
`handler`	`Handler` : `Handler` identifying the callback thread, null for the main thread

editProperties

```
public AccountManagerFuture<Bundle> editProperties (String accountType,
                Activity activity,
                AccountManagerCallback<Bundle> callback,
                Handler handler)
```

Offers the user an opportunity to change an authenticator's settings. These properties are for the authenticator in general, not a particular account. Not all authenticators support this method.

This method may be called from any thread, but the returned `AccountManagerFuture` must not be used on the main thread.

This method requires the caller to have the same signature as the authenticator associated with the specified account type.

NOTE: If targeting your app to work on API level 22 and before, MANAGE_ACCOUNTS permission is needed for those platforms. See docs for this function in API level 22.
Requires android.Manifest.permission.MANAGE_ACCOUNTS

Parameters	
`accountType`	`String` : The account type associated with the authenticator to adjust
`activity`	`Activity` : The `Activity` context to use for launching a new authenticator-defined sub-Activity to adjust authenticator settings; used only to call startActivity(); if null, the settings dialog will not be launched directly, but the necessary `Intent` will be returned to the caller instead
`callback`	`AccountManagerCallback` : Callback to invoke when the request completes, null for no callback
`handler`	`Handler` : `Handler` identifying the callback thread, null for the main thread

finishSession

```
public AccountManagerFuture<Bundle> finishSession (Bundle sessionBundle,
                Activity activity,
                AccountManagerCallback<Bundle> callback,
                Handler handler)
```

Finishes the session started by `startAddAccountSession(String, String, String, Bundle, Activity, AccountManagerCallback, Handler)` or `startUpdateCredentialsSession(Account, String, Bundle, Activity, AccountManagerCallback, Handler)` . This will either add the account to AccountManager or update the local credentials stored.

This method may be called from any thread, but the returned `AccountManagerFuture` must not be used on the main thread.

Parameters

`sessionBundle`	`Bundle` : a `Bundle` created by `startAddAccountSession(String, String, String, Bundle, Activity, AccountManagerCallback, Handler)` or `startUpdateCredentialsSession(Account, String, Bundle, Activity, AccountManagerCallback, Handler)`
`activity`	`Activity` : The `Activity` context to use for launching a new authenticator-defined sub-Activity to prompt the user to create an account or reauthenticate existing account; used only to call startActivity(); if null, the prompt will not be launched directly, but the necessary `Intent` will be returned to the caller instead
`callback`	`AccountManagerCallback` : Callback to invoke when the request completes, null for no callback
`handler`	`Handler` : `Handler` identifying the callback thread, null for the main thread

Returns	
`AccountManagerFuture<Bundle>`	An `AccountManagerFuture` which resolves to a Bundle with these fields if an activity was supplied and an account was added to device or local credentials were updated:: • `KEY_ACCOUNT_NAME` - the name of the account created • `KEY_ACCOUNT_TYPE` - the type of the account • `KEY_ACCOUNT_STATUS_TOKEN` - optional, token to check status of the account If no activity was specified and additional information is needed from user, the returned Bundle may only contain `KEY_INTENT` with the `Intent` needed to launch the actual account creation process. If an error occurred, `AccountManagerFuture.getResult()` throws: • `AuthenticatorException` if no authenticator was registered for this account type or the authenticator failed to respond • `OperationCanceledException` if the operation was canceled for any reason, including the user canceling the creation process or adding accounts (of this type) has been disabled by policy • `IOException` if the authenticator experienced an I/O problem creating a new account, usually because of network trouble

getAccounts

```
public Account[] getAccounts ()
```

Lists all accounts visible to the caller regardless of type. Equivalent to getAccountsByType(null). These accounts may be visible because the user granted access to the account, or the AbstractAccountAuthenticator managing the account did so or because the client shares a signature with the managing AbstractAccountAuthenticator.

Caution: This method returns personal and sensitive user data. If your app accesses, collects, uses, or shares personal and sensitive data, you must clearly disclose that fact to users. For apps published on Google Play, policies protecting user data require that you do the following:

- Disclose to the user how your app accesses, collects, uses, or shares personal and sensitive data. Learn more about acceptable disclosure and consent.
- Provide a privacy policy that describes your use of this data on- and off-device.

To learn more, visit the Google Play Policy regarding user data.

It is safe to call this method from the main thread.
Requires `Manifest.permission.GET_ACCOUNTS`

Returns

`Account[]`	An array of `Account`, one for each account. Empty (never null) if no accounts have been added.

getAccountsAndVisibilityForPackage

```
public Map<Account, Integer> getAccountsAndVisibilityForPackage (String packageName,
                String accountType)
```

Gets all accounts of given type and their visibility for specific package. This method requires the caller to have a signature match with the authenticator that manages accountType. It is a helper method which combines calls to `getAccountsByType(String)` by authenticator and `getAccountVisibility(Account, String)` for every returned account.

Parameters	
`packageName`	`String` : Package name
`accountType`	`String` : `Account` type

getAccountsByType

```
public Account[] getAccountsByType (String type)
```

Lists all accounts of particular type visible to the caller. These accounts may be visible because the user granted access to the account, or the AbstractAccountAuthenticator managing the account did so or because the client shares a signature with the managing AbstractAccountAuthenticator.

The account type is a string token corresponding to the authenticator and useful domain of the account. For example, there are types corresponding to Google and Facebook. The exact string token to use will be published somewhere associated with the authenticator in question.

Caution: This method returns personal and sensitive user data. If your app accesses, collects, uses, or shares personal and sensitive data, you must clearly disclose that fact to users. For apps published on Google Play, policies protecting user data require that you do the following:

- Disclose to the user how your app accesses, collects, uses, or shares personal and sensitive data. Learn more about acceptable disclosure and consent.
- Provide a privacy policy that describes your use of this data on- and off-device.

To learn more, visit the Google Play Policy regarding user data.

It is safe to call this method from the main thread.

Caller targeting API level `Build.VERSION_CODES.O` and above, will get list of accounts made visible to it by user (see `newChooseAccountIntent(Account,List,String[],String,String,String[],Bundle)`) or AbstractAccountAuthenticator using `setAccountVisibility(Account, String, int)` . `Manifest.permission.GET_ACCOUNTS` permission is not used.

Caller targeting API level below `Build.VERSION_CODES.O` that have not been granted the `Manifest.permission.GET_ACCOUNTS` permission, will only see those accounts managed by AbstractAccountAuthenticators whose signature matches the client.

NOTE: If targeting your app to work on API level `Build.VERSION_CODES.LOLLIPOP_MR1` and before, `Manifest.permission.GET_ACCOUNTS` permission is needed for those platforms, irrespective of uid or signature match. See docs for this function in API level `Build.VERSION_CODES.LOLLIPOP_MR1` .
Requires `Manifest.permission.GET_ACCOUNTS`

Parameters	
`type`	`String` : The type of accounts to return, null to retrieve all accounts

Returns	
`Account[]`	An array of `Account` , one per matching account. Empty (never null) if no accounts of the specified type have been added.

getAccountsByTypeAndFeatures

```
public AccountManagerFuture<Account[]> getAccountsByTypeAndFeatures (String type,
                String[] features,
                AccountManagerCallback<Account[]> callback,
                Handler handler)
```

Lists all accounts of a type which have certain features. The account type identifies the authenticator (see `getAccountsByType(String)`). Account features are authenticator-specific string tokens identifying boolean account properties (see `hasFeatures(Account, String, AccountManagerCallback, Handler)`).

Unlike `getAccountsByType(String)` , this method calls the authenticator, which may contact the server or do other work to check account features, so the method returns an `AccountManagerFuture` .

This method may be called from any thread, but the returned `AccountManagerFuture` must not be used on the main thread.

Caller targeting API level `Build.VERSION_CODES.O` and above, will get list of accounts made visible to it by user (see `newChooseAccountIntent(Account,List,String[],String,String,String[],Bundle)`) or AbstractAccountAuthenticator using `setAccountVisibility(Account, String, int)` . `Manifest.permission.GET_ACCOUNTS` permission is not used.

Caller targeting API level below `Build.VERSION_CODES.O` that have not been granted the `Manifest.permission.GET_ACCOUNTS` permission, will only see those accounts managed by AbstractAccountAuthenticators whose signature matches the client.

NOTE: If targeting your app to work on API level `Build.VERSION_CODES.LOLLIPOP_MR1` and before, `Manifest.permission.GET_ACCOUNTS` permission is needed for those platforms, irrespective of uid or signature match. See docs for this function in API level `Build.VERSION_CODES.LOLLIPOP_MR1` .

Parameters	
`type`	`String` : The type of accounts to return, must not be null
`features`	`String` : An array of the account features to require, may be null or empty *
`callback`	`AccountManagerCallback` : Callback to invoke when the request completes, null for no callback
`handler`	`Handler` : `Handler` identifying the callback thread, null for the main thread

getAccountsByTypeForPackage

```
public Account[] getAccountsByTypeForPackage (String type,
                String packageName)
```

Returns the accounts visible to the specified package in an environment where some apps are not authorized to view all accounts. This method can only be called by system apps and authenticators managing the type. Beginning API level `Build.VERSION_CODES.O` it also return accounts which user can make visible to the application (see `VISIBILITY_USER_MANAGED_VISIBLE`).

Parameters	
`type`	`String` : The type of accounts to return, null to retrieve all accounts
`packageName`	`String` : The package name of the app for which the accounts are to be returned

Returns	
`Account[]`	An array of `Account`, one per matching account. Empty (never null) if no accounts of the specified type can be accessed by the package.

getAuthToken

```
public AccountManagerFuture<Bundle> getAuthToken (Account account,
                String authTokenType,
                Bundle options,
                boolean notifyAuthFailure,
                AccountManagerCallback<Bundle> callback,
                Handler handler)
```

Gets an auth token of the specified type for a particular account, optionally raising a notification if the user must enter credentials. This method is intended for background tasks and services where the user should not be immediately interrupted with a password prompt.

If a previously generated auth token is cached for this account and type, then it is returned. Otherwise, if a saved password is available, it is sent to the server to generate a new auth token. Otherwise, an `Intent` is returned which, when started, will prompt the user for a password. If the notifyAuthFailure parameter is set, a status bar notification is also created with the same Intent, alerting the user that they need to enter a password at some point.

In that case, you may need to wait until the user responds, which could take hours or days or forever. When the user does respond and supply a new password, the account manager will broadcast the `LOGIN_ACCOUNTS_CHANGED_ACTION` Intent and notify `OnAccountsUpdateListener` which applications can use to try again.

If notifyAuthFailure is not set, it is the application's responsibility to launch the returned Intent at some point. Either way, the result from this call will not wait for user action.

Some authenticators have auth token *types*, whose value is authenticator-dependent. Some services use different token types to access different functionality -- for example, Google uses different auth tokens to access Gmail and Google Calendar for the same account.

This method may be called from any thread, but the returned `AccountManagerFuture` must not be used on the main thread.

NOTE: If targeting your app to work on API level 22 and before, USE_CREDENTIALS permission is needed for those platforms. See docs for this function in API level 22.
Requires android.Manifest.permission.USE_CREDENTIALS

Parameters	
`account`	`Account`: The account to fetch an auth token for
`authTokenType`	`String`: The auth token type, an authenticator-dependent string token, must not be null
`options`	`Bundle`: Authenticator-specific options for the request, may be null or empty
`notifyAuthFailure`	`boolean`: True to add a notification to prompt the user for a password if necessary, false to leave that to the caller
`callback`	`AccountManagerCallback`: Callback to invoke when the request completes, null for no callback
`handler`	`Handler`: `Handler` identifying the callback thread, null for the main thread

Returns	
`AccountManagerFuture<Bundle>`	An `AccountManagerFuture` which resolves to a Bundle with at least the following fields on success:

	• `KEY_ACCOUNT_NAME` - the name of the account you supplied • `KEY_ACCOUNT_TYPE` - the type of the account • `KEY_AUTHTOKEN` - the auth token you wanted (Other authenticator-specific values may be returned.) If the user must enter credentials, the returned Bundle contains only `KEY_INTENT` with the `Intent` needed to launch a prompt. If an error occurred, `AccountManagerFuture.getResult()` throws: • `AuthenticatorException` if the authenticator failed to respond • `OperationCanceledException` if the operation is canceled for any reason, incluidng the user canceling a credential request • `IOException` if the authenticator experienced an I/O problem creating a new auth token, usually because of network trouble If the account is no longer present on the device, the return value is authenticator-dependent. The caller should verify the validity of the account before requesting an auth token.

getAuthToken

```
public AccountManagerFuture<Bundle> getAuthToken (Account account,
                String authTokenType,
                Bundle options,
                Activity activity,
                AccountManagerCallback<Bundle> callback,
                Handler handler)
```

Gets an auth token of the specified type for a particular account, prompting the user for credentials if necessary. This method is intended for applications running in the foreground where it makes sense to ask the user directly for a password.

If a previously generated auth token is cached for this account and type, then it is returned. Otherwise, if a saved password is available, it is sent to the server to generate a new auth token. Otherwise, the user is prompted to enter a password.

Some authenticators have auth token *types*, whose value is authenticator-dependent. Some services use different token types to access different functionality -- for example, Google uses different auth tokens to access Gmail and Google Calendar for the same account.

NOTE: If targeting your app to work on API level 22 and before, USE_CREDENTIALS permission is needed for those platforms. See docs for this function in API level 22.

This method may be called from any thread, but the returned `AccountManagerFuture` must not be used on the main thread. Requires android.Manifest.permission.USE_CREDENTIALS

Parameters	
`account`	`Account` : The account to fetch an auth token for
`authTokenType`	`String` : The auth token type, an authenticator-dependent string token, must not be null
`options`	`Bundle` : Authenticator-specific options for the request, may be null or empty
`activity`	`Activity` : The `Activity` context to use for launching a new authenticator-defined sub-Activity to prompt the user for a password if necessary; used only to call startActivity(); must not be null.
`callback`	`AccountManagerCallback` : Callback to invoke when the request completes, null for no callback
`handler`	`Handler` : `Handler` identifying the callback thread, null for the main thread

getAuthToken

```
public AccountManagerFuture<Bundle> getAuthToken (Account account,
                String authTokenType,
                boolean notifyAuthFailure,
                AccountManagerCallback<Bundle> callback,
                Handler handler)
```

This method was deprecated in API level 15.
use `getAuthToken(Account,String,android.os.Bundle,boolean,AccountManagerCallback,android.os.Handler)` instead

Gets an auth token of the specified type for a particular account, optionally raising a notification if the user must enter credentials. This method is intended for background tasks and services where the user should not be immediately interrupted with a password prompt.

If a previously generated auth token is cached for this account and type, then it is returned. Otherwise, if a saved password is available, it is sent to the server to generate a new auth token. Otherwise, an `Intent` is returned which, when started, will prompt the user for a password. If the notifyAuthFailure parameter is set, a status bar notification is also created with the same Intent, alerting the user that they need to enter a password at some point.

In that case, you may need to wait until the user responds, which could take hours or days or forever. When the user does respond and supply a new password, the account manager will broadcast the `LOGIN_ACCOUNTS_CHANGED_ACTION` Intent and notify `OnAccountsUpdateListener` which applications can use to try again.

If notifyAuthFailure is not set, it is the application's responsibility to launch the returned Intent at some point. Either way, the result from this call will not wait for user action.

Some authenticators have auth token *types*, whose value is authenticator-dependent. Some services use different token types to access different functionality -- for example, Google uses different auth tokens to access Gmail and Google Calendar for the same account.

This method may be called from any thread, but the returned `AccountManagerFuture` must not be used on the main thread. Requires android.Manifest.permission.USE_CREDENTIALS

Parameters	
`account`	`Account` : The account to fetch an auth token for
`authTokenType`	`String` : The auth token type, an authenticator-dependent string token, must not be null
`notifyAuthFailure`	`boolean` : True to add a notification to prompt the user for a password if necessary, false to leave that to the caller
`callback`	`AccountManagerCallback` : Callback to invoke when the request completes, null for no callback
`handler`	`Handler` : `Handler` identifying the callback thread, null for the main thread
Returns	
`AccountManagerFuture<Bundle>`	An `AccountManagerFuture` which resolves to a Bundle with at least the following fields on success: • `KEY_ACCOUNT_NAME` - the name of the account you supplied • `KEY_ACCOUNT_TYPE` - the type of the account • `KEY_AUTHTOKEN` - the auth token you wanted (Other authenticator-specific values may be returned.) If the user must enter credentials, the returned Bundle contains only `KEY_INTENT` with the `Intent` needed to launch a prompt. If an error occurred, `AccountManagerFuture.getResult()` throws: • `AuthenticatorException` if the authenticator failed to respond

	• `OperationCanceledException` if the operation is canceled for any reason, incluidng the user canceling a credential request • `IOException` if the authenticator experienced an I/O problem creating a new auth token, usually because of network trouble If the account is no longer present on the device, the return value is authenticator-dependent. The caller should verify the validity of the account before requesting an auth token.

getAuthTokenByFeatures

```
public AccountManagerFuture<Bundle> getAuthTokenByFeatures (String accountType,
                String authTokenType,
                String[] features,
                Activity activity,
                Bundle addAccountOptions,
                Bundle getAuthTokenOptions,
                AccountManagerCallback<Bundle> callback,
                Handler handler)
```

This convenience helper combines the functionality of `getAccountsByTypeAndFeatures(String, String, AccountManagerCallback, Handler)` , `getAuthToken(Account, String, Bundle, Activity, AccountManagerCallback, Handler)` , and `addAccount(String, String, String, Bundle, Activity, AccountManagerCallback, Handler)` .

This method gets a list of the accounts matching specific type and feature set which are visible to the caller (see `getAccountsByType(String)` for details); if there is exactly one already visible account, it is used; if there are some accounts for which user grant visibility, the user is prompted to pick one; if there are none, the user is prompted to add one. Finally, an auth token is acquired for the chosen account.

This method may be called from any thread, but the returned `AccountManagerFuture` must not be used on the main thread.

NOTE: If targeting your app to work on API level 22 and before, MANAGE_ACCOUNTS permission is needed for those platforms. See docs for this function in API level 22.
Requires android.Manifest.permission.MANAGE_ACCOUNTS

Parameters	
`accountType`	`String` : The account type required (see `getAccountsByType(String)`), must not be null
`authTokenType`	`String` : The desired auth token type (see `getAuthToken(Account, String, Bundle, Activity, AccountManagerCallback, Handler)`), must not be null
`features`	`String` : Required features for the account (see `getAccountsByTypeAndFeatures(String, String, AccountManagerCallback, Handler)`), may be null or empty
`activity`	`Activity` : The `Activity` context to use for launching new sub-Activities to prompt to add an account, select an account, and/or enter a password, as necessary; used only to call startActivity(); should not be null
`addAccountOptions`	`Bundle` : Authenticator-specific options to use for adding new accounts; may be null or empty
`getAuthTokenOptions`	`Bundle` : Authenticator-specific options to use for getting auth tokens; may be null or empty
`callback`	`AccountManagerCallback` : Callback to invoke when the request completes, null for no callback
`handler`	`Handler` : `Handler` identifying the callback thread, null for the main thread

getAuthenticatorTypes

```
public AuthenticatorDescription[] getAuthenticatorTypes ()
```

Lists the currently registered authenticators.

It is safe to call this method from the main thread.

No permission is required to call this method.

Caller targeting API level 34 and above, the results are filtered by the rules of package visibility.

getPackagesAndVisibilityForAccount

```
public Map<String, Integer> getPackagesAndVisibilityForAccount (Account account)
```

Returns package names and visibility which were explicitly set for given account.

This method requires the caller to have a signature match with the authenticator that owns the specified account.

Parameters	
`account`	`Account` : The account for which visibility data should be returned
Returns	
`Map<String, Integer>`	Map from package names to visibility for given account

getPassword

```
public String getPassword (Account account)
```

Gets the saved password associated with the account. This is intended for authenticators and related code; applications should get an auth token instead.

It is safe to call this method from the main thread.

This method requires the caller to have a signature match with the authenticator that owns the specified account.

NOTE: If targeting your app to work on API level `Build.VERSION_CODES.LOLLIPOP_MR1` and before, AUTHENTICATE_ACCOUNTS permission is needed for those platforms. See docs for this function in API level `Build.VERSION_CODES.LOLLIPOP_MR1` .
Requires android.Manifest.permission.AUTHENTICATE_ACCOUNTS

Parameters	
`account`	`Account` : The account to query for a password. Must not be `null` .
Returns	
`String`	The account's password, null if none or if the account doesn't exist

getPreviousName

```
public String getPreviousName (Account account)
```

Gets the previous name associated with the account or `null` , if none. This is intended so that clients of `OnAccountsUpdateListener` can determine if an authenticator has renamed an account.

It is safe to call this method from the main thread.

Parameters	
`account`	`Account` : The account to query for a previous name.
Returns	
`String`	The account's previous name, null if the account has never been renamed.

getUserData

```
public String getUserData (Account account,
                String key)
```

Gets the user data named by "key" associated with the account. This is intended for authenticators and related code to store arbitrary metadata along with accounts. The meaning of the keys and values is up to the authenticator for the account.

It is safe to call this method from the main thread.

This method requires the caller to have a signature match with the authenticator that owns the specified account.

NOTE: If targeting your app to work on API level `Build.VERSION_CODES.LOLLIPOP_MR1` and before, AUTHENTICATE_ACCOUNTS permission is needed for those platforms. See docs for this function in API level `Build.VERSION_CODES.LOLLIPOP_MR1` .
Requires android.Manifest.permission.AUTHENTICATE_ACCOUNTS

Parameters	
`account`	`Account` : The account to query for user data
`key`	`String`
Returns	
`String`	The user data, null if the account, key doesn't exist, or the user is locked

hasFeatures

```
public AccountManagerFuture<Boolean> hasFeatures (Account account,
                String[] features,
                AccountManagerCallback<Boolean> callback,
                Handler handler)
```

Finds out whether a particular account has all the specified features. Account features are authenticator-specific string tokens identifying boolean account properties. For example, features are used to tell whether Google accounts have a particular service (such as Google Calendar or Google Talk) enabled. The feature names and their meanings are published somewhere associated with the authenticator in question.

This method may be called from any thread, but the returned `AccountManagerFuture` must not be used on the main thread.

If caller target API level is below `Build.VERSION_CODES.O` , it is required to hold the permission `Manifest.permission.GET_ACCOUNTS` or have a signature match with the AbstractAccountAuthenticator that manages the account.

Parameters	
`account`	`Account` : The `Account` to test
`features`	`String` : An array of the account features to check
`callback`	`AccountManagerCallback` : Callback to invoke when the request completes, null for no callback

`handler`	`Handler` : `Handler` identifying the callback thread, null for the main thread

invalidateAuthToken

```
public void invalidateAuthToken (String accountType,
                String authToken)
```

Removes an auth token from the AccountManager's cache. Does nothing if the auth token is not currently in the cache. Applications must call this method when the auth token is found to have expired or otherwise become invalid for authenticating requests. The AccountManager does not validate or expire cached auth tokens otherwise.

It is safe to call this method from the main thread.

NOTE: If targeting your app to work on API level 22 and before, MANAGE_ACCOUNTS or USE_CREDENTIALS permission is needed for those platforms. See docs for this function in API level 22.
Requires android.Manifest.permission.MANAGE_ACCOUNTS or android.Manifest.permission.USE_CREDENTIALS

Parameters	
`accountType`	`String` : The account type of the auth token to invalidate, must not be null
`authToken`	`String` : The auth token to invalidate, may be null

isCredentialsUpdateSuggested

```
public AccountManagerFuture<Boolean> isCredentialsUpdateSuggested (Account account,
                String statusToken,
                AccountManagerCallback<Boolean> callback,
                Handler handler)
```

Checks whether `updateCredentials(Account, String, Bundle, Activity, AccountManagerCallback, Handler)` or `startUpdateCredentialsSession(Account, String, Bundle, Activity, AccountManagerCallback, Handler)` should be called with respect to the specified account.

This method may be called from any thread, but the returned `AccountManagerFuture` must not be used on the main thread.

Parameters	
`account`	`Account` : The `Account` to be checked whether `updateCredentials(Account, String, Bundle, Activity, AccountManagerCallback, Handler)` or `startUpdateCredentialsSession(Account, String, Bundle, Activity, AccountManagerCallback, Handler)` should be called
`statusToken`	`String` : a String of token to check account staus
`callback`	`AccountManagerCallback` : Callback to invoke when the request completes, null for no callback
`handler`	`Handler` : `Handler` identifying the callback thread, null for the main thread

newChooseAccountIntent

```
public static Intent newChooseAccountIntent (Account selectedAccount,
                List<Account> allowableAccounts,
                String[] allowableAccountTypes,
                String descriptionOverrideText,
                String addAccountAuthTokenType,
                String[] addAccountRequiredFeatures,
                Bundle addAccountOptions)
```

Returns an intent to an `Activity` that prompts the user to choose from a list of accounts. The caller will then typically start the activity by calling `startActivityForResult(intent, ...);` .

On success the activity returns a Bundle with the account name and type specified using keys `KEY_ACCOUNT_NAME` and `KEY_ACCOUNT_TYPE` . Chosen account is marked as `VISIBILITY_USER_MANAGED_VISIBLE` to the caller (see `setAccountVisibility(Account, String, int)`) and will be returned to it in consequent `getAccountsByType(String)`) calls.

The most common case is to call this with one account type, e.g.:

```
 newChooseAccountIntent(null, null, new String[]{"com.google"}, null, null, null,
null);
```

Parameters	
`selectedAccount`	`Account` : if specified, indicates that the `Account` is the currently selected one, according to the caller's definition of selected.
`allowableAccounts`	`List` : an optional `List` of accounts that are allowed to be shown. If not specified then this field will not limit the displayed accounts.
`allowableAccountTypes`	`String` : an optional string array of account types. These are used both to filter the shown accounts and to filter the list of account types that are shown when adding an account. If not specified then this field will not limit the displayed account types when adding an account.
`descriptionOverrideText`	`String` : if non-null this string is used as the description in the accounts chooser screen rather than the default
`addAccountAuthTokenType`	`String` : this string is passed as the `addAccount(String, String, String, Bundle, Activity, AccountManagerCallback, Handler)` authTokenType parameter
`addAccountRequiredFeatures`	`String` : this string array is passed as the `addAccount(String, String, String, Bundle, Activity, AccountManagerCallback, Handler)` requiredFeatures parameter
`addAccountOptions`	`Bundle` : This `Bundle` is passed as the `addAccount(String, String, String, Bundle, Activity, AccountManagerCallback, Handler)` options parameter
Returns	
`Intent`	an `Intent` that can be used to launch the ChooseAccount activity flow.

newChooseAccountIntent

```
public static Intent newChooseAccountIntent (Account selectedAccount,
                ArrayList<Account> allowableAccounts,
                String[] allowableAccountTypes,
                boolean alwaysPromptForAccount,
                String descriptionOverrideText,
                String addAccountAuthTokenType,
                String[] addAccountRequiredFeatures,
                Bundle addAccountOptions)
```

Deprecated in favor of `newChooseAccountIntent(Account,List,String[],String,String,String[],Bundle)` . Returns an intent to an `Activity` that prompts the user to choose from a list of accounts. The caller will then typically start the activity by calling `startActivityForResult(intent, ...);` .

On success the activity returns a Bundle with the account name and type specified using keys `KEY_ACCOUNT_NAME` and `KEY_ACCOUNT_TYPE` . Chosen account is marked as `VISIBILITY_USER_MANAGED_VISIBLE` to the caller (see

`setAccountVisibility(Account, String, int)`) and will be returned to it in consequent `getAccountsByType(String)`) calls.

The most common case is to call this with one account type, e.g.:

```
  newChooseAccountIntent(null, null, new String[]{"com.google"}, false, null,
null, null, null);
```

Parameters	
`selectedAccount`	`Account` : if specified, indicates that the `Account` is the currently selected one, according to the caller's definition of selected.
`allowableAccounts`	`ArrayList` : an optional `List` of accounts that are allowed to be shown. If not specified then this field will not limit the displayed accounts.
`allowableAccountTypes`	`String` : an optional string array of account types. These are used both to filter the shown accounts and to filter the list of account types that are shown when adding an account. If not specified then this field will not limit the displayed account types when adding an account.
`alwaysPromptForAccount`	`boolean` : boolean that is ignored.
`descriptionOverrideText`	`String` : if non-null this string is used as the description in the accounts chooser screen rather than the default
`addAccountAuthTokenType`	`String` : this string is passed as the `addAccount(String, String, String, Bundle, Activity, AccountManagerCallback, Handler)` authTokenType parameter
`addAccountRequiredFeatures`	`String` : this string array is passed as the `addAccount(String, String, String, Bundle, Activity, AccountManagerCallback, Handler)` requiredFeatures parameter
`addAccountOptions`	`Bundle` : This `Bundle` is passed as the `addAccount(String, String, String, Bundle, Activity, AccountManagerCallback, Handler)` options parameter

Returns	
`Intent`	an `Intent` that can be used to launch the ChooseAccount activity flow.

notifyAccountAuthenticated

```
public boolean notifyAccountAuthenticated (Account account)
```

Notifies the system that the account has just been authenticated. This information may be used by other applications to verify the account. This should be called only when the user has entered correct credentials for the account.

It is not safe to call this method from the main thread. As such, call it from another thread.

This method requires the caller to have a signature match with the authenticator that owns the specified account. Requires android.Manifest.permission.AUTHENTICATE_ACCOUNTS

Parameters	
`account`	`Account` : The `Account` to be updated.

Returns	
`boolean`	boolean `true` if the authentication of the account has been successfully acknowledged. Otherwise `false` .

peekAuthToken

```
public String peekAuthToken (Account account,
                String authTokenType)
```

Gets an auth token from the AccountManager's cache. If no auth token is cached for this account, null will be returned -- a new auth token will not be generated, and the server will not be contacted. Intended for use by the authenticator, not directly by applications.

It is safe to call this method from the main thread.

This method requires the caller to have a signature match with the authenticator that manages the specified account.

NOTE: If targeting your app to work on API level 22 and before, AUTHENTICATE_ACCOUNTS permission and same UID as account's authenticator is needed for those platforms. See docs for this function in API level 22.
Requires android.Manifest.permission.AUTHENTICATE_ACCOUNTS

Parameters	
`account`	`Account` : The account for which an auth token is to be fetched. Cannot be `null` .
`authTokenType`	`String` : The type of auth token to fetch. Cannot be `null` .
Returns	
`String`	The cached auth token for this account and type, or null if no auth token is cached, the account does not exist, or the user is locked

removeAccount

```
public AccountManagerFuture<Bundle> removeAccount (Account account,
                Activity activity,
                AccountManagerCallback<Bundle> callback,
                Handler handler)
```

Removes an account from the AccountManager. Does nothing if the account does not exist. Does not delete the account from the server. The authenticator may have its own policies preventing account deletion, in which case the account will not be deleted.

This method may be called from any thread, but the returned `AccountManagerFuture` must not be used on the main thread.

This method requires the caller to have a signature match with the authenticator that manages the specified account, be a profile owner or have the `ERROR(/android.Manifest.permission#REMOVE_ACCOUNTS)` permission.

NOTE: If targeting your app to work on API level 22 and before, MANAGE_ACCOUNTS permission is needed for those platforms. See docs for this function in API level 22.

Parameters	
`account`	`Account` : The `Account` to remove
`activity`	`Activity` : The `Activity` context to use for launching a new authenticator-defined sub-Activity to prompt the user to delete an account; used only to call startActivity(); if null, the prompt will not be launched directly, but the `Intent` may be returned to the caller instead
`callback`	`AccountManagerCallback` : Callback to invoke when the request completes, null for no callback
`handler`	`Handler` : `Handler` identifying the callback thread, null for the main thread

removeAccount

```
public AccountManagerFuture<Boolean> removeAccount (Account account,
                AccountManagerCallback<Boolean> callback,
                Handler handler)
```

This method was deprecated in API level 22.
use `removeAccount(Account,Activity,AccountManagerCallback,Handler)` instead

Removes an account from the AccountManager. Does nothing if the account does not exist. Does not delete the account from the server. The authenticator may have its own policies preventing account deletion, in which case the account will not be deleted.

This method requires the caller to have a signature match with the authenticator that manages the specified account.

NOTE: If targeting your app to work on API level 22 and before, MANAGE_ACCOUNTS permission is needed for those platforms. See docs for this function in API level 22.
Requires android.Manifest.permission.MANAGE_ACCOUNTS

Parameters	
`account`	`Account` : The `Account` to remove
`callback`	`AccountManagerCallback` : Callback to invoke when the request completes, null for no callback
`handler`	`Handler` : `Handler` identifying the callback thread, null for the main thread

removeAccountExplicitly

```
public boolean removeAccountExplicitly (Account account)
```

Removes an account directly. Normally used by authenticators, not directly by applications. Does not delete the account from the server. The authenticator may have its own policies preventing account deletion, in which case the account will not be deleted.

It is safe to call this method from the main thread.

This method requires the caller to have a signature match with the authenticator that manages the specified account.

NOTE: If targeting your app to work on API level 22 and before, AUTHENTICATE_ACCOUNTS permission and same UID as account's authenticator is needed for those platforms. See docs for this function in API level 22.
Requires android.Manifest.permission.AUTHENTICATE_ACCOUNTS

Parameters	
`account`	`Account` : The `Account` to delete.
Returns	
`boolean`	True if the account was successfully deleted, false if the account did not exist, the account is null, or another error occurs.

renameAccount

```
public AccountManagerFuture<Account> renameAccount (Account account,
                String newName,
                AccountManagerCallback<Account> callback,
                Handler handler)
```

Rename the specified `Account` . This is equivalent to removing the existing account and adding a new renamed account with the old account's user data.

It is safe to call this method from the main thread.

This method requires the caller to have a signature match with the authenticator that manages the specified account.

NOTE: If targeting your app to work on API level 22 and before, AUTHENTICATE_ACCOUNTS permission and same UID as account's authenticator is needed for those platforms. See docs for this function in API level 22.
Requires android.Manifest.permission.AUTHENTICATE_ACCOUNTS

Parameters	
`account`	`Account` : The `Account` to rename
`newName`	`String` : String name to be associated with the account.
`callback`	`AccountManagerCallback` : Callback to invoke when the request completes, null for no callback
`handler`	`Handler` : `Handler` identifying the callback thread, null for the main thread

setAuthToken

```
public void setAuthToken (Account account,
                String authTokenType,
                String authToken)
```

Adds an auth token to the AccountManager cache for an account. If the account does not exist then this call has no effect. Replaces any previous auth token for this account and auth token type. Intended for use by the authenticator, not directly by applications.

It is safe to call this method from the main thread.

This method requires the caller to have a signature match with the authenticator that manages the specified account.

NOTE: If targeting your app to work on API level 22 and before, AUTHENTICATE_ACCOUNTS permission and same UID as account's authenticator is needed for those platforms. See docs for this function in API level 22.
Requires android.Manifest.permission.AUTHENTICATE_ACCOUNTS

Parameters	
`account`	`Account` : The account to set an auth token for
`authTokenType`	`String` : The type of the auth token, see {#getAuthToken}
`authToken`	`String` : The auth token to add to the cache

setPassword

```
public void setPassword (Account account,
                String password)
```

Sets or forgets a saved password. This modifies the local copy of the password used to automatically authenticate the user; it does not change the user's account password on the server. Intended for use by the authenticator, not directly by applications.

Calling this method does not update the last authenticated timestamp, referred by `KEY_LAST_AUTHENTICATED_TIME`. To update it, call `notifyAccountAuthenticated(Account)` after getting success.

It is safe to call this method from the main thread.

This method requires the caller to have a signature match with the authenticator that manages the specified account.

NOTE: If targeting your app to work on API level 22 and before, AUTHENTICATE_ACCOUNTS permission and same UID as account's authenticator is needed for those platforms. See docs for this function in API level 22.

Requires android.Manifest.permission.AUTHENTICATE_ACCOUNTS

Parameters	
`account`	`Account` : The account whose password is to be set. Cannot be `null` .
`password`	`String` : The password to set, null to clear the password

setUserData

```
public void setUserData (Account account,
                String key,
                String value)
```

Sets one userdata key for an account. Intended by use for the authenticator to stash state for itself, not directly by applications. The meaning of the keys and values is up to the authenticator.

It is safe to call this method from the main thread.

This method requires the caller to have a signature match with the authenticator that manages the specified account.

NOTE: If targeting your app to work on API level 22 and before, AUTHENTICATE_ACCOUNTS permission and same UID as account's authenticator is needed for those platforms. See docs for this function in API level 22.
Requires android.Manifest.permission.AUTHENTICATE_ACCOUNTS

Parameters	
`account`	`Account` : Account whose user data is to be set. Must not be `null` .
`key`	`String` : String user data key to set. Must not be null
`value`	`String` : String value to set, `null` to clear this user data key

startAddAccountSession

```
public AccountManagerFuture<Bundle> startAddAccountSession (String accountType,
                String authTokenType,
                String[] requiredFeatures,
                Bundle options,
                Activity activity,
                AccountManagerCallback<Bundle> callback,
                Handler handler)
```

Asks the user to authenticate with an account of a specified type. The authenticator for this account type processes this request with the appropriate user interface. If the user does elect to authenticate with a new account, a bundle of session data for installing the account later is returned with optional account password and account status token.

This method may be called from any thread, but the returned `AccountManagerFuture` must not be used on the main thread.

NOTE: The account will not be installed to the device by calling this api alone. #finishSession should be called after this to install the account on device.

Parameters	
`accountType`	`String` : The type of account to add; must not be null
`authTokenType`	`String` : The type of auth token (see `getAuthToken(Account, String, Bundle, Activity, AccountManagerCallback, Handler)`) this account will need to be able to generate, null for none

`requiredFeatures`	`String` : The features (see `hasFeatures(Account, String, AccountManagerCallback, Handler)`) this account must have, null for none
`options`	`Bundle` : Authenticator-specific options for the request, may be null or empty
`activity`	`Activity` : The `Activity` context to use for launching a new authenticator-defined sub-Activity to prompt the user to create an account; used only to call startActivity(); if null, the prompt will not be launched directly, but the necessary `Intent` will be returned to the caller instead
`callback`	`AccountManagerCallback` : Callback to invoke when the request completes, null for no callback
`handler`	`Handler` : `Handler` identifying the callback thread, null for the main thread

startUpdateCredentialsSession

```
public AccountManagerFuture<Bundle> startUpdateCredentialsSession (Account account,
                String authTokenType,
                Bundle options,
                Activity activity,
                AccountManagerCallback<Bundle> callback,
                Handler handler)
```

Asks the user to enter a new password for the account but not updating the saved credentials for the account until `finishSession(Bundle, Activity, AccountManagerCallback, Handler)` is called.

This method may be called from any thread, but the returned `AccountManagerFuture` must not be used on the main thread.

NOTE: The saved credentials for the account alone will not be updated by calling this API alone. #finishSession should be called after this to update local credentials

Parameters	
`account`	`Account` : The account to update credentials for
`authTokenType`	`String` : The credentials entered must allow an auth token of this type to be created (but no actual auth token is returned); may be null
`options`	`Bundle` : Authenticator-specific options for the request; may be null or empty
`activity`	`Activity` : The `Activity` context to use for launching a new authenticator-defined sub-Activity to prompt the user to enter a password; used only to call startActivity(); if null, the prompt will not be launched directly, but the necessary `Intent` will be returned to the caller instead
`callback`	`AccountManagerCallback` : Callback to invoke when the request completes, null for no callback
`handler`	`Handler` : `Handler` identifying the callback thread, null for the main thread

updateCredentials

```
public AccountManagerFuture<Bundle> updateCredentials (Account account,
                String authTokenType,
                Bundle options,
                Activity activity,
                AccountManagerCallback<Bundle> callback,
                Handler handler)
```

Asks the user to enter a new password for an account, updating the saved credentials for the account. Normally this happens automatically when the server rejects credentials during an auth token fetch, but this can be invoked directly to ensure we have the correct credentials stored.

This method may be called from any thread, but the returned `AccountManagerFuture` must not be used on the main thread.

NOTE: If targeting your app to work on API level 22 and before, MANAGE_ACCOUNTS permission is needed for those platforms. See docs for this function in API level 22.
Requires android.Manifest.permission.MANAGE_ACCOUNTS

Parameters	
`account`	`Account` : The account to update credentials for
`authTokenType`	`String` : The credentials entered must allow an auth token of this type to be created (but no actual auth token is returned); may be null
`options`	`Bundle` : Authenticator-specific options for the request; may be null or empty
`activity`	`Activity` : The `Activity` context to use for launching a new authenticator-defined sub-Activity to prompt the user to enter a password; used only to call startActivity(); if null, the prompt will not be launched directly, but the necessary `Intent` will be returned to the caller instead
`callback`	`AccountManagerCallback` : Callback to invoke when the request completes, null for no callback
`handler`	`Handler` : `Handler` identifying the callback thread, null for the main thread

---