Peter Gustafson writing sample
Epic Games
March 19, 2021

Auth Interface

[Auth Interface](https://dev.epicgames.com/docs/services/en-US/Interfaces/Auth/index.html)

The Auth Interface handles account-related interactions with EOS to authenticate you and obtain access tokens.

 Activate EAS in your [Developer Portal](https://dev.epicgames.com/docs/services/en-US/DevPortal/index.html). You can initialize the EOS SDK and the Auth Interface without EAS enabled, but all Auth Interface calls will fail.

Authentication Functions

To access the authentication functions, create an `EOS_HAuth` handle by calling the [Platform Interface](https://dev.epicgames.com/docs/services/en-US/Interfaces/Platform/index.html) function `EOS_Platform_GetAuthInterface`.

[ARE THERE ANY MORE STEPS NEEDED HERE?]

Login

 1. To login, make a call to the `EOS_Auth_Login` function with an `EOS_Auth_LoginOptions` structure containing your account credentials.

 2. Whether the login attempt succeeds or fails, your callback of `EOS_Auth_OnLoginCallback`starts running upon completion.

 3. The `EOS_Auth_LoginOptions` must be initialized with its `ApiVersion` variable set to `EOS_AUTH_LOGIN_API_LATEST`, and its `Credentials` variable (of type `EOS_Auth_Credentials`) containing the following information:

Property	Value
`ApiVersion`	`EOS_AUTH_CREDENTIALS_API_LATEST`.
``Id``	The identity of the user logging in. Must be a user-readable identity, like an email address or display name.
`Token`	The user's login credentials or authentication token.
`Type`	The type of credential that this login attempt is using. See [EOS_ExternalCredentialType](https://dev.epicgames.com/docs/services/INT/API/Members/Enums/Connect/EOS_EExternalCredentialType/index.html "EOS372029427EExternalCredentialType") for methods.

4. Pass your `EOS_HAuth`, your `EOS_Auth_LoginOptions`, and your callback information to the function.
5
6. Provided the `EOS_HPlatform` handle is ticking, the callback will run when the operation finishes.

Problems setting up your account? Visit the [**developer help forum**](https://www.epicgames.com/help/en-US/).

Preferred Login Types

As of EOS SDK version 1.5, the preferred login types by platform are below:

Platform Type	Login Type	Notes
Console	`EOS_LCT_DeviceCode` with `EOS_LCT_PersistentAuth`	PIN code authorizes your local device to receive and store access tokens.
PC or Mobile Device	`EOS_LCT_AccountPortal` with `EOS_LCT_PersistentAuth`	Locally stored tokens.
Epic Games Launcher	`EOS_LCT_ExchangeCode`	Launcher exchange code for your login.

Login to Epic Account Outside the Epic Games Launcher

To support multiple logins outside of the Epic Games Launcher, use the `EOS_LCT_AccountPortal` login type to create long-lived refresh tokens.

 - The SDK automatically receives a refresh token from the authentication backend after a successful login to the user's Epic Account.

 - On PC and mobile platforms, the SDK automatically stores the token in the local keychain of the locally logged-in user on the device.

 - For the local keychain, the SDK uses the secure credentials store provided by the device's operating system.

 - On consoles, call the `EOS_Auth_CopyUserAuthToken` to retrieve the received `refresh_token` value.

> It's important to store the `refresh_token` on a per-user basis and for logged-in users of the device.

Automatic Login
When automatically logging in the locally, the game will first call `EOS_Auth_Login` with the `EOS_LCT_PersistentAuth` login type.

 - On PCs and mobile devices, the `Id` and `Token` input fields should be set to `NULL` since the SDK manages the long-lived access credentials.

 - The SDK will check for a refresh token in the keychain of the local user, and will automatically use a token, if it finds one, to log the user into their Epic Account.

 - Following a successful login on those platforms, the SDK will automatically update the refresh token in the local keychain.

 - On consoles, the `Id` field should be `NULL`, but the `Token` field should point to the refresh token string from a previous login.

 - Following a successful console login, the application calls `EOS_Auth_CopyUserAuthToken` to retrieve an updated refresh token and overwrite the previous one in the console's storage device. Doing so safeguards other local users from logging in with another wrong Epic Account.

Login Failure

If `EOS_Auth_Login` fails for any reason, proceed with the default login method for the platform.

 - If `EOS_Auth_Login` found a refresh token but failed to log in because the server rejected the token (call failed) the application should delete the token, since it is obsolete and will continue to cause failures in all future sessions.

PCs and mobile devices
 - Call `EOS_Auth_DeletePersistentAuth` to explicitly remove any stored credentials in the local keychain for the user. This will delete the long-lived refresh token from the keychain of the local user.

Consoles
 - Delete the stored token from local storage. The application should then proceed to the platform's default login flow. On consoles, the application must remove the stored token from local storage.

 - In the case that a logged-in user wants to disable automatic login, call `EOS_Auth_Logout` to log out, and `EOS_Auth_DeletePersistentAuth` to revoke the user's long-lived logon session on the authentication backend.

Log Out

 1. To log out, call the `EOS_Auth_Logout` function with an

`EOS_Auth_LogoutOptions` data structure.

 2. When the operation completes, your callback function of `EOS_Auth_OnLogoutCallback`, will run.

 3. Initialize your `EOS_Auth_LogoutOptions` structure as follows:

Property	Value
`ApiVersion`	`EOS_AUTH_LOGOUT_API_LATEST`
``LocalUserId``	The `EOS_EpicAccountId`

 4. Pass your `EOS_HAuth`, your `EOS_Auth_LogoutOptions` structure, and your callback function to `EOS_Auth_Logout`. Provided the `EOS_HPlatform` handle is ticking, the callback you provide will run when the operation finishes.

 5. If the `EOS_LCT_PersistentAuth` login type has been used, call the function `EOS_Auth_DeletePersistentAuth` to revoke the long-lived login session on the authentication backend. On PC Desktop and Mobile, this permanently erases the local user login.

Console

When using the `EOS_LCT_DeviceCode` Console login type, an additional callback is made by the `EOS_Auth_Login` API at the beginning of the login flow.

 - In the first callback, an `EOS_Auth_PinGrantInfo` data structure is returned to the caller.
 - It contains the PIN code `UserCode` and the website URI the application needs to use a separate device, such as a mobile phone or PC.

 - In versions 1.5 and higher, a new field named `VerificationURIComplete` allows login directly. For example, it can be used in a button or hyperlink, or it can be rendered as a QR code to be scanned by a mobile phone.

PC & Mobile Device

PCs and mobile devices use long-lived refresh tokens granted by the authentication backend, and specific to the device and user account.

 - On these platforms, the SDK automatically stores and retrieves these tokens as needed, and updates them following each login.

 - This method is also available on consoles but the application must manage refresh token storage on a per-user basis to prevent local users from logging into each other's accounts.

Epic Games Launcher

When an application associated with the Epic Games Launcher starts, the it provides a command line with several parameters like below:

```
-AUTH_LOGIN=unused -AUTH_PASSWORD=<password> -AUTH_TYPE=exchangecode -epicapp=<appid> -epicenv=Prod -EpicPortal -epicusername=<username> -epicuserid=<userid> -epiclocale=en-US
```

Field Name Definitions

Property	Value
`AUTH_LOGIN`	This field may be the user ID, but it is presently unused.
`AUTH_PASSWORD`	Your Exchange Code which should be provided as the token during login.
``AUTH_TYPE``	The type will read `exchangecode` indicating the `EOS_Auth_LoginCredentials` should use the type `EOS_LCT_ExchangeCode`.

 - The application must parse this information and pass it into `EOS_Auth_Login` through the `EOS_Auth_Credentials` structure.

The `EOS_Auth_Credentials` have three variables:

Variable	Value
`Id`	Leave `Id` blank since this method doesn't require an `Id`.
`Token`	Provide the Exchange Code from the `AUTH_PASSWORD` parameter.
`Type`	Set to `EOS_LCT_ExchangeCode`.

Auth Scopes

As of EOS SDK version 1.5, `EOS_Auth_LoginOptions` contains a new field named `ScopeFlags`, of type [`EOS_EAuthScopeFlags`](https://dev.epicgames.com/docs/services/INT/API/Members/Enums/Auth/EOS_EAuthScopeFlags/index.html "EOS372029427EAuthScopeFlags").

 - Scopes are sets of permissions that are required for your application to function properly.

 - For example, if your application needs to see the user's friends list, you must request the `EOS_AS_FriendsList` scope consent. When requesting consent, your request must match the scopes configured for

the product in the Developer Portal.

 - Multiple users can be logged in simultaneously on a single local device using the same shared `EOS_HPlatform` instance.

Status Change Notifications

The EOS SDK periodically verifies your local user authentication status during the application's lifetime. Below are the functions and descriptions:

Function	Description
`EOS_Auth_OnLoginStatusChangedCallback`	The Auth Interface invokes the callback type `EOS_Auth_OnLoginStatusChangedCallback` upon any such change for any local player.
`EOS_Auth_AddNotifyLoginStatusChanged`	You can attach your own callback function to this process with the `EOS_Auth_AddNotifyLoginStatusChanged` function.
`EOS_Auth_OnLoginStatusChangedCallback`	The `EOS_Auth_OnLoginStatusChangedCallback` runs when a local user authentication status changes.

 You receive both the callback for the log in or log out events.

Connectivity
Connectivity loss during an application's lifetime doesn't mean a user is logged out.

 - The EOS backend notifies the Auth Interface when a logout event takes place. This is the only case in which it is safe to assume that the user is officially considered offline.

 - Connectivity problems like service outages or local hardware failure can cause API features to fail.

Checking Current Authentication Status

To check the player's current status on demand, use the `EOS_Auth_GetLoginStatus` function.

Verifying Authentication

Some Epic users want to verify a remote user's identity, such as when a user attempts to join a dedicated server.

 1. Call `EOS_Auth_VerifyUserAuth` with an `EOS_Auth_VerifyUserAuthOptions` asks EOS to verify the user's authentication status.

 2. Run your callback `EOS_Auth_OnVerifyUserAuthCallback` with `EOS_Auth_VerifyUserAuthCallbackInfo`.

 3. To initialize your `EOS_Auth_VerifyUserAuthOptions` structure, set the following values:

Property	Value
`ApiVersion`	`EOS_AUTH_VERIFYUSERAUTH_API_LATEST`.
``AuthToken``	`EOS_Auth_Token` type used by the remote user.

4. Pass your `EOS_HAuth`, your `EOS_Auth_LogoutOptions` structure, and your callback function to `EOS_Auth_Logout`. Provided the `EOS_HPlatform` handle is ticking, the callback you provide will run when the operation finishes.

5. Pass your `EOS_HAuth`, your `EOS_Auth_VerifyUserAuthOptions` data structure, and your callback information to the function.

6. When the operation finishes, your callback runs. If a failure occurs, you should remove the user from the server as it's likely the user is attempting to spoof another user.

External Account Authentication

 1. To login with `EOS_Auth_Login` using an external account, set the `Type` in `EOS_Auth_Credentials` to `EOS_LCT_ExternalAuth`.

 2. Set the `ExternalType` to an external credential type.

 3. Set the `Token` to the external authentication token.

For example, to login with Steam, use `EOS_ECT_STEAM_APP_TICKET` as the `ExternalType`, and the `Token` set to Encrypted Steam App Ticket.

If an external account is not linked when the user begins the authentication flow, the Account Portal will enable authentication to continue once a user has logged in. Afterwards, the external account will be linked to the user's Epic Account.

The Identity Providers settings in the Developer Portal will need to be updated to enable the linking of the providers using external account authentication. See [Configuring Identity Providers](https://dev.epicgames.com/docs/services/en-US/DevPortal/IdentityProviderManagement/index.html#configuringidentityproviders) for more information.

Integrate Game Launcher With Epic Games Store

If your game provides a launcher to include additional launch options,

promotions or other news, your launcher must manage the login flow.

Exchange codes generated by the Epic Games Launcher expire after a short period of time.

- The Epic Games Launcher starts the third-party launcher by passing the exchange code on the command line as described above in the section `Associating With the Epic Games Launcher`.

- The third-party launcher uses the Exchange Code to login the player by using the `EOS_Auth_Login` API.

Steps
1. Initialize `EOS_Auth_LoginOptions` by setting the `Type` and `Token` fields of the `EOS_Auth_ClientCredentials` to `EOS_LCT_ExchangeCode` and the exchange code from the command line respectively.

2. To launch a game, use the `EOS_Auth_CopyUserAuthToken` API to get a copy of the token details.

3. Copy the `RefreshToken` from the `EOS_Auth_Token` and call the `EOS_Auth_Token_Release` API to free the memory allocated by the SDK.

4. Pass the refresh token to the game application by setting an environment variable that the game can read on startup.

5. When the game starts, log another player in by using the `EOS_Auth_Login` API.

6. Initialize `EOS_Auth_LoginOptions` by setting the `Type` and `Token` fields of the `EOS_Auth_ClientCredentials` to `EOS_LCT_RefreshToken` and the refresh token from the environment variable respectively.

If you get stuck, visit the [**developer help forum**](https://www.epicgames.com/help/en-US/).