EpicOnlineServices Extension

Free, cross-platform services that make it easier and faster for devs to successfully launch, operate, and scale high-quality games.

See the official page for more documentation

Setup

Follow these guides to get yourself going on everything you need for your new game.

- Setup

Modules

This extension API presents a variety of modules that can be used to push your game to the next level. These are the included modules:

- Achievements
- Auth

- Connect
- Friends
- Leaderboards
- Metrics
- Platform
- PlayerDataStorage
- ProgressionSnapshot
- Sanctions
- Stats
- TitleStorage
- User Interface
- UserInfo
- Other MISCELLANEOUS

Result Constants

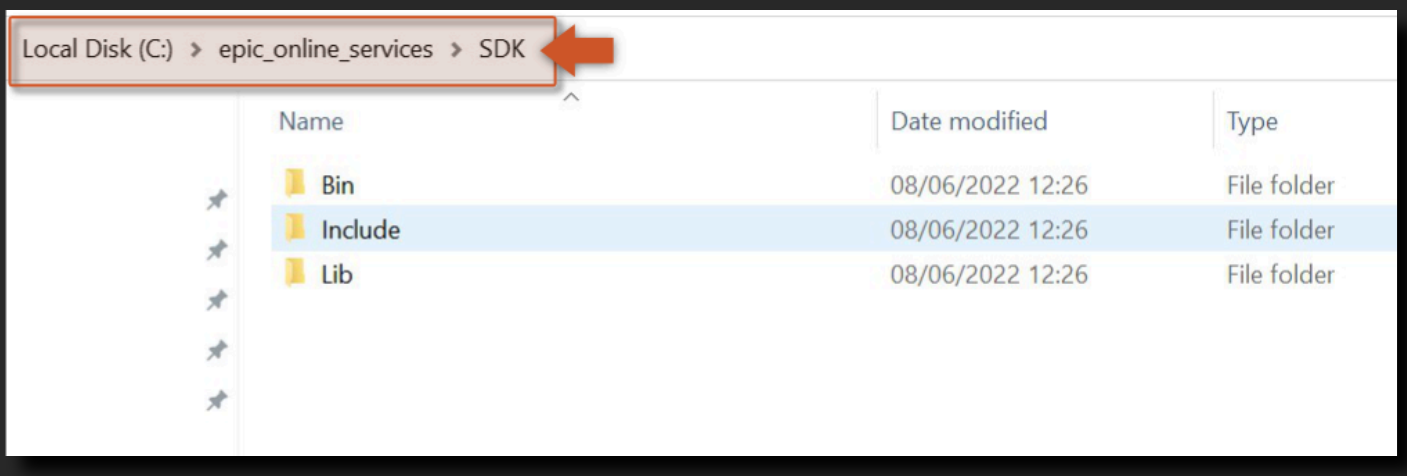
This extension API can return a large set of result constants depending on the function being called. These are the available result constants:

- EpicGames Result

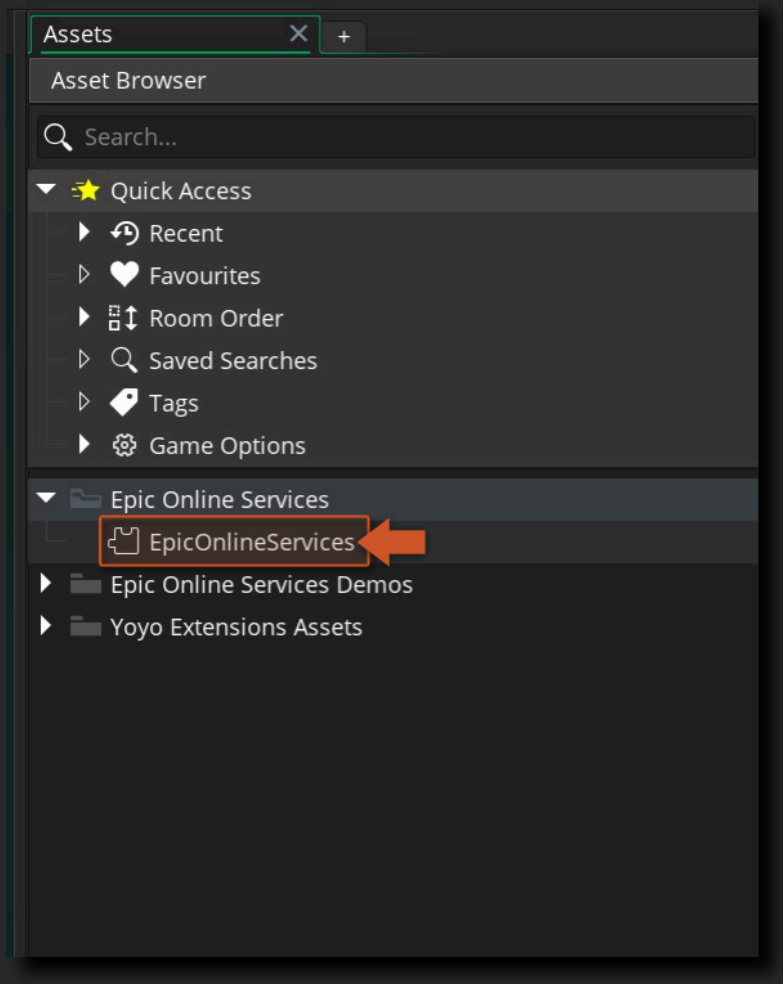
Setup Guide

To use the Epic Online Services API extension you should follow these steps:

1. Import this Epic Online Services extension into your project, if you haven't done that already.
2. The Epic Launcher App needs to be **installed**, **running** and with an account **logged in** (**official site**).
3. Download Epic Online Services SDK (C version, 1.14.2) from Epics's **developer portal** and extract the contents of the zip into a directory of your choice (e.g.: `C:\epic_online_services\sdk`).



4. To set up your game properties, double click on the EpicOnlineServices extension in your Asset Browser in the IDE.



5. At the bottom of the extension widow you will find all the configurable options of the Epic Online Services.

6. The options are split in two sections **Building Properties** and **Game Properties**. The first one is a folder path that should point to the extracted folder of step 3, the second section will allow you to configure all the settings that are required for running and publishing a game to Epic Store.

NOTE If you set **Debug Mode** to `false` this will force your app to be launched by the EOS launcher. This should only be used when you are ready to send your app to production (DO NOT try to run the game from the IDE while debug mode is disabled)

Heading

This function is used to do something.

Note that it may also do something.

EpicGames Result Constant	Descr
`EOS_Success`	Successful result. no further e
`EOS_NoConnection`	Failed due to no connection
`EOS_InvalidCredentials`	Failed login due to invalid cred
`EOS_InvalidUser`	Failed due to invalid or missin
`EOS_InvalidAuth`	Failed due to invalid or missin user (e.g. not logged in)
`EOS_AccessDenied`	Failed due to invalid access
`EOS_MissingPermissions`	If the client does not possess t
`EOS_Token_Not_Account`	If the token provided does not
`EOS_TooManyRequests`	Throttled due to too many req
`EOS_AlreadyPending`	Async request was already per
`EOS_InvalidParameters`	Invalid parameters specified fo
`EOS_InvalidRequest`	Invalid request
`EOS_UnrecognizedResponse`	Failed due to unable to parse response
`EOS_IncompatibleVersion`	Incompatible client for backen
`EOS_NotConfigured`	Not configured correctly for us
`EOS_AlreadyConfigured`	Already configured for use.
`EOS_NotImplemented`	Feature not available on this in
`EOS_Canceled`	Operation was canceled (likely
`EOS_NotFound`	The requested information wa
`EOS_OperationWillRetry`	An error occurred during an as will be retried. Callbacks receiv again in the future.
`EOS_NoChange`	The request had no effect
`EOS_VersionMismatch`	The request attempted to use versions

`EOS_LimitExceeded`	A maximum limit was exceede `EOS_TooManyRequests`
`EOS_Disabled`	Feature or client ID performin disabled.
`EOS_DuplicateNotAllowed`	Duplicate entry not allowed
`EOS_MissingParameters_DEPRECATED`	Required parameters are miss is no longer used.
`EOS_InvalidSandboxId`	Sandbox ID is invalid
`EOS_TimedOut`	Request timed out
`EOS_PartialResult`	A query returned some but no
`EOS_Missing_Role`	Client is missing the white-liste
`EOS_Missing_Feature`	Client is missing the white-liste
`EOS_Invalid_Sandbox`	The sandbox given to the back
`EOS_Invalid_Deployment`	The deployment given to the b
`EOS_Invalid_Product`	The product ID specified to the
`EOS_Invalid_ProductUserID`	The product user ID specified
`EOS_ServiceFailure`	There was a failure with the ba
`EOS_CacheDirectoryMissing`	Cache directory is not set in pl
`EOS_CacheDirectoryInvalid`	Cache directory is not accessib
`EOS_InvalidState`	The request failed because res
`EOS_RequestInProgress`	Request is in progress
`EOS_Auth_AccountLocked`	Account locked due to login fa
`EOS_Auth_AccountLockedForUpdate`	Account locked by update ope
`EOS_Auth_InvalidRefreshToken`	Refresh token used was invali
`EOS_Auth_InvalidToken`	Invalid access token, typically backend environments
`EOS_Auth_AuthenticationFailure`	Invalid bearer token
`EOS_Auth_InvalidPlatformToken`	Invalid platform token
`EOS_Auth_WrongAccount`	Auth parameters are not asso
`EOS_Auth_WrongClient`	Auth parameters are not asso
`EOS_Auth_FullAccountRequired`	Full account is required
`EOS_Auth_HeadlessAccountRequired`	Headless account is required
`EOS_Auth_PasswordResetRequired`	Password reset is required
`EOS_Auth_PasswordCannotBeReused`	Password was previously used

`EOS_Auth_Expired`	Authorization code/exchange
`EOS_Auth_ScopeConsentRequired`	Consent has not been given by
`EOS_Auth_ApplicationNotFound`	The application has no profile
`EOS_Auth_ScopeNotFound`	The requested consent wasn't
`EOS_Auth_AccountFeatureRestricted`	This account has been denied
`EOS_Auth_PinGrantCode`	Pin grant code initiated
`EOS_Auth_PinGrantExpired`	Pin grant code attempt expire
`EOS_Auth_PinGrantPending`	Pin grant code attempt pendir
`EOS_Auth_ExternalAuthNotLinked`	External auth source did not y
`EOS_Auth_ExternalAuthRevoked`	External auth access revoked
`EOS_Auth_ExternalAuthInvalid`	External auth token cannot be
`EOS_Auth_ExternalAuthRestricted`	External auth cannot be linked restrictions
`EOS_Auth_ExternalAuthCannotLogin`	External auth cannot be used
`EOS_Auth_ExternalAuthExpired`	External auth is expired
`EOS_Auth_ExternalAuthIsLastLoginType`	External auth cannot be remov way to login
`EOS_Auth_ExchangeCodeNotFound`	Exchange code not found
`EOS_Auth_OriginatingExchangeCodeSessionExpired`	Originating exchange code ses
`EOS_Auth_PersistentAuth_AccountNotActive`	The account has been disabled authentication
`EOS_Auth_MFARequired`	MFA challenge required
`EOS_Auth_ParentalControls`	Parental locks are in place
`EOS_Auth_NoRealId`	Korea real ID association requ
`EOS_Friends_InviteAwaitingAcceptance`	An outgoing friend invitation is another invite to the same use
`EOS_Friends_NoInvitation`	There is no friend invitation to
`EOS_Friends_AlreadyFriends`	Users are already friends, so s erroneous
`EOS_Friends_NotFriends`	Users are not friends, so delet
`EOS_Friends_TargetUserTooManyInvites`	Remote user has too many inv
`EOS_Friends_LocalUserTooManyInvites`	Local user has too many invite
`EOS_Friends_TargetUserFriendLimitExceeded`	Remote user has too many fri friendship

`EOS_Friends_LocalUserFriendLimitExceeded`	Local user has too many friend
`EOS_Presence_DataInvalid`	Request data was null or inval
`EOS_Presence_DataLengthInvalid`	Request contained too many d or the request would overflow data allowed
`EOS_Presence_DataKeyInvalid`	Request contained data with a
`EOS_Presence_DataKeyLengthInvalid`	Request contained data with a
`EOS_Presence_DataValueInvalid`	Request contained data with a
`EOS_Presence_DataValueLengthInvalid`	Request contained data with a
`EOS_Presence_RichTextInvalid`	Request contained an invalid r
`EOS_Presence_RichTextLengthInvalid`	Request contained a rich text s
`EOS_Presence_StatusInvalid`	Request contained an invalid s
`EOS_Ecom_EntitlementStale`	The entitlement retrieved is st information
`EOS_Ecom_CatalogOfferStale`	The offer retrieved is stale, re- information
`EOS_Ecom_CatalogItemStale`	The item or associated structu for updated information
`EOS_Ecom_CatalogOfferPriceInvalid`	The one or more offers has an caused by the price setup.
`EOS_Ecom_CheckoutLoadError`	The checkout page failed to lo
`EOS_Sessions_SessionInProgress`	Session is already in progress
`EOS_Sessions_TooManyPlayers`	Too many players to register w
`EOS_Sessions_NoPermission`	Client has no permissions to a
`EOS_Sessions_SessionAlreadyExists`	Session already exists in the sy
`EOS_Sessions_InvalidLock`	Session lock required for oper
`EOS_Sessions_InvalidSession`	Invalid session reference
`EOS_Sessions_SandboxNotAllowed`	Sandbox ID associated with au
`EOS_Sessions_InviteFailed`	Invite failed to send
`EOS_Sessions_InviteNotFound`	Invite was not found with the s
`EOS_Sessions_UpsertNotAllowed`	This client may not modify the
`EOS_Sessions_AggregationFailed`	Backend nodes unavailable to
`EOS_Sessions_HostAtCapacity`	Individual backend node is as
`EOS_Sessions_SandboxAtCapacity`	Sandbox on node is at capacit

`EOS_Sessions_SessionNotAnonymous`	An anonymous operation was anonymous session
`EOS_Sessions_OutOfSync`	Session is currently out of sync saved locally but needs to sync
`EOS_Sessions_TooManyInvites`	User has received too many in
`EOS_Sessions_PresenceSessionExists`	Presence session already exist
`EOS_Sessions_DeploymentAtCapacity`	Deployment on node is at cap
`EOS_Sessions_NotAllowed`	Session operation not allowed
`EOS_Sessions_PlayerSanctioned`	Session operation not allowed
`EOS_PlayerDataStorage_FilenameInvalid`	Request filename was invalid
`EOS_PlayerDataStorage_FilenameLengthInvalid`	Request filename was too long
`EOS_PlayerDataStorage_FilenameInvalidChars`	Request filename contained in
`EOS_PlayerDataStorage_FileSizeTooLarge`	Request operation would grow
`EOS_PlayerDataStorage_FileSizeInvalid`	Request file length is not valid
`EOS_PlayerDataStorage_FileHandleInvalid`	Request file handle is not valid
`EOS_PlayerDataStorage_DataInvalid`	Request data is invalid
`EOS_PlayerDataStorage_DataLengthInvalid`	Request data length was invali
`EOS_PlayerDataStorage_StartIndexInvalid`	Request start index was invalid
`EOS_PlayerDataStorage_RequestInProgress`	Request is in progress
`EOS_PlayerDataStorage_UserThrottled`	User is marked as throttled wh some operations because limi
`EOS_PlayerDataStorage_EncryptionKeyNotSet`	Encryption key is not set durin
`EOS_PlayerDataStorage_UserErrorFromDataCallback`	User data callback returned er (EOS_PlayerDataStorage_EWri or EOS_PlayerDataStorage_ERead
`EOS_PlayerDataStorage_FileHeaderHasNewerVersion`	User is trying to read file that version of SDK. Game/SDK nee
`EOS_PlayerDataStorage_FileCorrupted`	The file is corrupted. In some c
`EOS_Connect_ExternalTokenValidationFailed`	EOS Auth service deemed the
`EOS_Connect_UserAlreadyExists`	EOS Auth user already exists
`EOS_Connect_AuthExpired`	EOS Auth expired
`EOS_Connect_InvalidToken`	EOS Auth invalid token
`EOS_Connect_UnsupportedTokenType`	EOS Auth doesn't support this
`EOS_Connect_LinkAccountFailed`	EOS Auth Account link failure

`EOS_Connect_ExternalServiceUnavailable`	EOS Auth External service for
`EOS_Connect_ExternalServiceConfigurationFailure`	EOS Auth External Service con Portal
`EOS_Connect_LinkAccountFailedMissingNintendoIdAccount_DEPRECATED`	EOS Auth Account link failure. Network Service Account with Account. DEPRECATED: The re and this error is no longer use
`EOS_UI_SocialOverlayLoadError`	The social overlay page failed
`EOS_Lobby_NotOwner`	Client has no permissions to m
`EOS_Lobby_InvalidLock`	Lobby lock required for opera
`EOS_Lobby_LobbyAlreadyExists`	Lobby already exists in the sys
`EOS_Lobby_SessionInProgress`	Lobby is already in progress
`EOS_Lobby_TooManyPlayers`	Too many players to register v
`EOS_Lobby_NoPermission`	Client has no permissions to a
`EOS_Lobby_InvalidSession`	Invalid lobby session referenc
`EOS_Lobby_SandboxNotAllowed`	Sandbox ID associated with au
`EOS_Lobby_InviteFailed`	Invite failed to send
`EOS_Lobby_InviteNotFound`	Invite was not found with the s
`EOS_Lobby_UpsertNotAllowed`	This client may not modify the
`EOS_Lobby_AggregationFailed`	Backend nodes unavailable to
`EOS_Lobby_HostAtCapacity`	Individual backend node is as
`EOS_Lobby_SandboxAtCapacity`	Sandbox on node is at capacity
`EOS_Lobby_TooManyInvites`	User has received too many in
`EOS_Lobby_DeploymentAtCapacity`	Deployment on node is at cap
`EOS_Lobby_NotAllowed`	Lobby operation not allowed
`EOS_Lobby_MemberUpdateOnly`	While restoring a lost connecti and only local member data w
`EOS_Lobby_PresenceLobbyExists`	Presence lobby already exists
`EOS_TitleStorage_UserErrorFromDataCallback`	User callback that receives dat error.
`EOS_TitleStorage_EncryptionKeyNotSet`	User forgot to set Encryption K Storage can't work without it.
`EOS_TitleStorage_FileCorrupted`	Downloaded file is corrupted.
`EOS_TitleStorage_FileHeaderHasNewerVersion`	Downloaded file's format is ne

`EOS_Mods_ModSdkProcessIsAlreadyRunning`	ModSdk process is already run the EOSSDK.
`EOS_Mods_ModSdkCommandIsEmpty`	ModSdk command is empty. E configuration file is missing or empty.
`EOS_Mods_ModSdkProcessCreationFailed`	Creation of the ModSdk proce from the SDK.
`EOS_Mods_CriticalError`	A critical error occurred in the that we were unable to resolve
`EOS_Mods_ToolInternalError`	A internal error occurred in th that we were unable to resolve
`EOS_Mods_IPCFailure`	A IPC failure occurred in the ex
`EOS_Mods_InvalidIPCResponse`	A invalid IPC response receive process.
`EOS_Mods_URILaunchFailure`	A URI Launch failure occurred process.
`EOS_Mods_ModIsNotInstalled`	Attempting to perform an acti installed. This error comes fro process.
`EOS_Mods_UserDoesNotOwnTheGame`	Attempting to perform an acti doesn't own. This error comes process.
`EOS_Mods_OfferRequestByIdInvalidResult`	Invalid result of the request to This error comes from the ext
`EOS_Mods_CouldNotFindOffer`	Could not find the offer for the the external ModSdk process.
`EOS_Mods_OfferRequestByIdFailure`	Request to get the offer for th comes from the external Mod
`EOS_Mods_PurchaseFailure`	Request to purchase the mod the external ModSdk process.
`EOS_Mods_InvalidGameInstallInfo`	Attempting to perform an acti installed or is partially installe external ModSdk process.
`EOS_Mods_CannotGetManifestLocation`	Failed to get manifest location configuration file is missing or empty

`EOS_Mods_UnsupportedOS`	Attempting to perform an acti support the current operating
`EOS_AntiCheat_ClientProtectionNotAvailable`	The anti-cheat client protectio the game was started using th
`EOS_AntiCheat_InvalidMode`	The current anti-cheat mode i
`EOS_AntiCheat_ClientProductIdMismatch`	The ProductId provided to the executable does not match wh EOS SDK
`EOS_AntiCheat_ClientSandboxIdMismatch`	The SandboxId provided to th executable does not match wh EOS SDK
`EOS_AntiCheat_ProtectMessageSessionKeyRequired`	(ProtectMessage/UnprotectMe available, but it is required to
`EOS_AntiCheat_ProtectMessageValidationFailed`	(ProtectMessage/UnprotectMe invalid
`EOS_AntiCheat_ProtectMessageInitializationFailed`	(ProtectMessage/UnprotectMe
`EOS_AntiCheat_PeerAlreadyRegistered`	(RegisterPeer) Peer is already
`EOS_AntiCheat_PeerNotFound`	(UnregisterPeer) Peer does no
`EOS_AntiCheat_PeerNotProtected`	(ReceiveMessageFromPeer) In protected
`EOS_AntiCheat_ClientDeploymentIdMismatch`	The DeploymentId provided to executable does not match wh EOS SDK
`EOS_AntiCheat_DeviceIdAuthIsNotSupported`	EOS Connect DeviceID auth m anti-cheat
`EOS_RTC_TooManyParticipants`	EOS RTC room cannot accept
`EOS_RTC_RoomAlreadyExists`	EOS RTC room already exists
`EOS_RTC_UserKicked`	The user kicked out from the
`EOS_RTC_UserBanned`	The user is banned
`EOS_RTC_RoomWasLeft`	EOS RTC room was left succes
`EOS_RTC_ReconnectionTimegateExpired`	Connection dropped due to lo
`EOS_ProgressionSnapshot_SnapshotIdUnavailable`	The number of available Snap exhausted.
`EOS_KWS_ParentEmailMissing`	The KWS user does not have a with the account. The parent a deleted

`EOS_KWS_UserGraduated`	The KWS user is no longer a m parent email
`EOS_Android_JavaVMNotStored`	EOS Android VM not stored

Achievements

The Achievements Interface provides a way for developers to retrieve data about a player's Epic Online Services achievements, unlock achievements for that player, and retrieve data about all of the Epic Online Services achievements belonging to an application.

Functions

These functions are provided for handling achievements:

- EpicGames_Achievements_AddNotifyAchievementsUnlockedV2
- EpicGames_Achievements_CopyAchievementDefinitionV2ByAchievementId
- EpicGames_Achievements_CopyAchievementDefinitionV2ByIndex
- EpicGames_Achievements_CopyPlayerAchievementByAchievementId
- EpicGames_Achievements_CopyPlayerAchievementByIndex
- EpicGames_Achievements_GetAchievementDefinitionCount
- EpicGames_Achievements_GetPlayerAchievementCount
- EpicGames_Achievements_QueryDefinitions
- EpicGames_Achievements_QueryPlayerAchievements
- EpicGames_Achievements_RemoveNotifyAchievementsUnlocked
- EpicGames_Achievements_UnlockAchievement

Structures

These are the structures used by this API:

- AchievementDefinition
- PlayerAchievement

EpicGames_Achievements_AddNotifyAchievementsUnlockedV2

Register to receive achievement unlocked notifications.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Achievements_AddNotifyAchievementsUnlockedV2

Syntax:

```
EpicGames_Achievements_AddNotifyAchievementsUnlockedV2()
```

Returns:

```
int64
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"EpicGames_Achievements_AddNotifyAchievementsUnlockedV2"`
UnlockTime	int64	POSIX timestamp when the achievement was unlocked
AchievementId	string	The Achievement ID for the achievement that was unlocked. Pass this to EpicGames_Achievements_CopyPlayerAchievementByAchieveme to get the full achievement information
UserId	string	The Product User ID for the user who received the unlocked achievements notification

Example:

```
identifier = EpicGames_Achievements_AddNotifyAchievementsUnlockedV2()
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_Achievements_AddNotifyAchievementsUnlockedV2")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Achievements_CopyAchievementDefinitionV2ByAchievementId

Fetches an AchievementDefinition from a given achievement ID.

NOTE Requires a previous call to EpicGames_Achievements_QueryDefinitions to store values in cache.

EXTERNAL A wrapper around EOS_Achievements_CopyAchievementDefinitionV2ByAchievementId

Syntax:

```
EpicGames_Achievements_CopyAchievementDefinitionV2ByAchievementId(AchievementId)
```

Argument	Type	Description
AchievementId	string	Achievement ID to look for when copying the definition from the cache

Returns:

```
struct (AchievementDefinition)
```

Example:

```
var struct =
EpicGames_Achievements_CopyAchievementDefinitionV2ByAchievementId("MyAchievement1")
if(struct.status == EpicGames_Success)
{
    var AchievementId = struct.AchievementId
}
```

The above code will show an example of how the function should be used. The achievement definition data is returned providing an achievement id.

EpicGames_Achievements_CopyAchievementDefinitionV2ByIndex

Fetches an AchievementDefinition from a given index.

NOTE Requires a previous call to EpicGames_Achievements_QueryDefinitions to store values in cache.

EXTERNAL A wrapper around EOS_Achievements_CopyAchievementDefinitionV2ByIndex

Syntax:

```
EpicGames_Achievements_CopyAchievementDefinitionV2ByIndex(index)
```

Argument	Type	Description
index	real	Index of the achievement definition to retrieve from the cache

Returns:

```
struct (AchievementDefinition)
```

Example:

```
for(var i = 0 ; i < EpicGames_Achievements_GetAchievementDefinitionCount() ; i ++)
{
    var struct = EpicGames_Achievements_CopyAchievementDefinitionV2ByIndex(i)
    if(struct.status == EpicGames_Success)
    {
         var AchievementId = struct.AchievementId
    }
}
```

The above code will show an example of how the function should be used. The achievement definition data is returned providing an achievement index.

EpicGames_Achievements_CopyPlayerAchievementByAchievementId

Fetches a player achievement from a given achievement ID.

NOTE Requires a previous call to EpicGames_Achievements_QueryPlayerAchievements to store values in cache.

EXTERNAL A wrapper around EOS_Achievements_CopyPlayerAchievementByAchievementId

Syntax:

```
EpicGames_Achievements_CopyPlayerAchievementByAchievementId(userID, userID_target, achievementID)
```

Argument	Type	Description
userID	string	The Product User ID for the user who is querying for a player achievement. For a Dedicated Server this should be null.
userID_target	string	The Product User ID for the user whose achievement is to be retrieved.
achievementID	string	Achievement ID to search for when retrieving player achievement data from the cache.

Returns:

```
struct (PlayerAchievement)
```

Example:

```
var struct = EpicGames_Achievements_CopyPlayerAchievementByAchievementId(userID,userID_target,achievementID
if(struct.status == EpicGames_Success)
{
```

```
    var AchievementId = struct.AchievementId
}
```

The above code will show an example of how the function should be used. The player achievement data is returned providing an achievement id.

EpicGames_Achievements_CopyPlayerAchievementByIndex

Fetches a player achievement from a given index.

NOTE Requires a previous call to EpicGames_Achievements_QueryPlayerAchievements to store values in cache.

EXTERNAL A wrapper around EOS_Achievements_CopyPlayerAchievementByIndex

Syntax:

```
EpicGames_Achievements_CopyPlayerAchievementByIndex(userID, userID_target, index)
```

Argument	Type	Description
userID	string	The Product User ID for the user who is querying for a player achievement. For a Dedicated Server this should be null.
userID_target	string	The Product User ID for the user whose achievement is to be retrieved.
index	real	The index of the player achievement data to retrieve from the cache.

Returns:

```
struct (PlayerAchievement)
```

Example:

```
for(var i = 0 ; i < EpicGames_Achievements_GetPlayerAchievementCount(userID) ; i ++)
{
    var struct = EpicGames_Achievements_CopyPlayerAchievementByIndex(i)
    if(struct.status == EpicGames_Success)
    {
        var AchievementId = struct.AchievementId
```

```
    }
}
```

The above code will show an example of how the function should be used. The player achievement data is returned providing an achievement index.

EpicGames_Achievements_GetAchievementDefinitionCount

Fetch the number of achievement definitions that are cached locally.

NOTE Requires a previous call to EpicGames_Achievements_QueryDefinitions to store values in cache.

EXTERNAL A wrapper around EOS_Achievements_GetAchievementDefinitionCount

Syntax:

```
EpicGames_Achievements_GetAchievementDefinitionCount()
```

Returns:

```
real
```

Example:

```
for(var i = 0 ; i < EpicGames_Achievements_GetAchievementDefinitionCount() ; i ++)
{
    var struct = EpicGames_Achievements_CopyAchievementDefinitionV2ByIndex(i)
    if(struct.status == EpicGames_Success)
    {
         var AchievementId = struct.AchievementId
    }
}
```

The above code will show an example of how the function should be used. After a successcull call to EpicGames_Achievements_QueryDefinitions, the function EpicGames_Achievements_GetAchievementDefinitionCount will return the number of entries in the query array which can then be accessed using the EpicGames_Achievements_CopyAchievementDefinitionV2ByIndex function.

EpicGames_Achievements_GetPlayerAchievementCount

Fetch the number of player achievements that are cached locally.

NOTE Requires a previous call to EpicGames_Achievements_QueryPlayerAchievements to store values in cache.

EXTERNAL A wrapper around EOS_Achievements_GetPlayerAchievementCount

Syntax:

```
EpicGames_Achievements_GetPlayerAchievementCount(userID)
```

Argument	Type	Description
userID	string	The Product User ID for the user whose achievement count is being retrieved.

Returns:

```
real
```

Example:

```
for(var i = 0 ; i < EpicGames_Achievements_GetPlayerAchievementCount(userID) ; i ++)
{
    var struct = EpicGames_Achievements_CopyPlayerAchievementByIndex(i)
    if(struct.status == EpicGames_Success)
    {
         var AchievementId = struct.AchievementId
    }
}
```

The above code will show an example of how the function should be used. After a successcull call to EpicGames_Achievements_QueryPlayerAchievements, the function EpicGames_Achievements_GetPlayerAchievementCount will return the number of entries

in the query array which can then be accessed using the EpicGames_Achievements_CopyPlayerAchievementByIndex function.

EpicGames_Achievements_QueryDefinitions

Query for a list of definitions for all existing achievements, including localized text, icon IDs and whether an achievement is hidden.

Once the callback has been fired with a successful EpicGames Result, it is possible to call one of the following functions:

- EpicGames_Achievements_CopyAchievementDefinitionV2ByAchievementId
- EpicGames_Achievements_CopyAchievementDefinitionV2ByIndex
- EpicGames_Achievements_GetAchievementDefinitionCount

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Achievements_QueryDefinitions

Syntax:

```
EpicGames_Achievements_QueryDefinitions(userId)
```

Argument	Type	Description
userId	string	Product User ID for user who is querying definitions.

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	Type	Description

type	string	`"EpicGames_Achievements_QueryDefinitions"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID

Example:

```
identifier = EpicGames_Achievements_QueryDefinitions(userId)
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_Achievements_QueryDefinitions")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Achievements_QueryPlayerAchievements

Query for a list of achievements for a specific player, including progress towards completion for each achievement.

Once the callback has been fired with a successful EpicGames Result, it is possible to call one of the following functions:

- EpicGames_Achievements_CopyPlayerAchievementByAchievementId
- EpicGames_Achievements_CopyPlayerAchievementByIndex
- EpicGames_Achievements_GetPlayerAchievementCount

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Achievements_QueryPlayerAchievements

Syntax:

```
EpicGames_Achievements_QueryPlayerAchievements(userID, userID_target)
```

Argument	Type	Description
userID	string	The Product User ID for the user who is querying for player achievements. For a Dedicated Server this should be null.
userID_target	string	The Product User ID for the user whose achievements are to be retrieved.

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"EpicGames_Achievements_QueryPlayerAchievements"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID

Example:

```
identifier = EpicGames_Achievements_QueryPlayerAchievements(userID,userID)
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_Achievements_QueryPlayerAchievements")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Achievements_RemoveNotifyAchievementsUnlocked

Unregister from receiving achievement unlocked notifications, should be passed the identifier returned from the function:

- EpicGames_Achievements_AddNotifyAchievementsUnlockedV2

EXTERNAL A wrapper around EOS_Achievements_RemoveNotifyAchievementsUnlocked

Syntax:

```
EpicGames_Achievements_RemoveNotifyAchievementsUnlocked(id)
```

Argument	Type	Description
id	real	The notification registration handle (return by EpicGames_Achievements_AddNotifyAchievementsUnlockedV2)

Returns:

```
N/A
```

Example:

```
handle = EpicGames_Achievements_AddNotifyAchievementsUnlockedV2()
//...
//later...
//...
EpicGames_Achievements_RemoveNotifyAchievementsUnlocked(handle)
```

The code sample above enables the achievement unlock notifications (EpicGames_Achievements_AddNotifyAchievementsUnlockedV2) and later disables them by refering to the previous generated handle.

EpicGames_Achievements_UnlockAchievement

Unlock a achievement for a specific player.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Achievements_UnlockAchievements

Syntax:

```
EpicGames_Achievements_UnlockAchievement(userID, AchievementID)
```

Argument	Type	Description
userID	string	The Product User ID for the user whose achievements we want to unlock.
AchievementID	string	Achievement ID to unlock.

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

Key	Type	Description
type	string	The string `"EpicGames_Achievements_UnlockAchievement"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors

status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_Achievements_UnlockAchievement()
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_Achievements_UnlockAchievement")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

PlayerAchievement

A player achievement is represented by a struct and contains information about a single player achievement. This struct is returned by the following functions:

- EpicGames_Achievements_CopyPlayerAchievementByAchievementId
- EpicGames_Achievements_CopyPlayerAchievementByIndex

The ***status*** member present in the struct can be represented by one of the following values:

- `EpicGames_Success` if the information is available and was correctly returned;
- `EpicGames_InvalidParameters` (extension internal error, should never be returned);
- `EpicGames_NotFound` if the achievement definition is not found;
- `EpicGames_Invalid_ProductUserID` if you pass an invalid user ID;

Member	Type	Description
status	EResult	The result value of the task
status_message	string	Text representation of the ***status*** code
AchievementId	string	This achievement's unique identifier
Progress	real	Progress towards completing this achievement (as a percentage)
UnlockTime	string	The POSIX timestamp when the achievement was unlocked. If th achievement has not been unlocked, this value will be EpicGames_ACHIEVEMENTS_ACHIEVEMENT_UNLOCKTIME_UNDE
StatInfoCount	string	The number of player stat info entries associated with this achievement
StatInfo TODO	array	Array of EpicGames_Achievements_PlayerStatInfo structures containing information about stat thresholds used to unlock the achievement and the player's current values for those stats
DisplayName	string	Localized display name for the achievement based on this specif player's current progress on the achievement
Description	string	Localized description for the achievement based on this specific player's current progress on the achievement

IconURL	string	URL of an icon to display for the achievement based on this spec player's current progress on the achievement. This may be null i is no data configured in the developer portal
FlavorText	string	Localized flavor text that can be used by the game in an arbitrary manner. This may be null if there is no data configured in the developer portal

AchievementDefinition

An achievement definition is represented by a struct and contains information about a single achievement definition with localized text. This struct is returned by the following functions:

- EpicGames_Achievements_CopyAchievementDefinitionV2ByAchievementId
- EpicGames_Achievements_CopyAchievementDefinitionV2ByIndex

The ***status*** member present in the struct can be represented by one of the following values:

- `EpicGames_Success` if the information is available and was correctly returned;
- `EpicGames_InvalidParameters` (extension internal error, should never be returned);
- `EpicGames_NotFound` if the achievement definition is not found;
- `EpicGames_Invalid_ProductUserID` if any of the userid options are incorrect;

Member	Type	Description
status	EResult	The result value of the task
status_message	string	Text representation of the ***status*** code
AchievementId	string	Achievement ID that can be used to uniquely identify the achievement
UnlockedDisplayName	string	Localized display name for the achievement when it has been unlocked
UnlockedDescription	string	Localized description for the achievement when it has been unlocked
LockedDisplayName	string	Localized display name for the achievement when it is locked or hidden
LockedDescription	string	Localized description for the achievement when it is locked or hidden

FlavorText	string	Localized flavor text that can be used by the game in an arbitrary manner. This may be null if there is no data configured in the development portal
UnlockedIconURL	string	URL of an icon to display for the achievement when it is unlocked. This may be null if there is no data configured in the development portal
LockedIconURL	string	URL of an icon to display for the achievement when it is locked or hidden. This may be null if there is no data configured in the development portal
bIsHidden	bool	true if the achievement is hidden; false otherwise

Auth

The Auth Interface lets players (users) log into their Epic Account from your game (product) so they can access the features provided by **Epic Account Services** (EAS), such as Friends, Presence, UserInfo and Ecom interfaces. The **Auth Interface** handles Epic account-related interactions with EOS, providing the ability to authenticate users and obtain access tokens.

Guides

This guide provides details on the **Authentication Flow**.

Functions

These functions are provided for handling authentication:

- **EpicGames_Auth_AddNotifyLoginStatusChanged**
- **EpicGames_Auth_CopyIdToken**
- **EpicGames_Auth_CopyUserAuthToken**
- **EpicGames_Auth_DeletePersistentAuth**
- **EpicGames_Auth_GetLoginStatus**
- **EpicGames_Auth_GetSelectedAccountId**
- **EpicGames_Auth_LinkAccount**
- **EpicGames_Auth_Login**
- **EpicGames_Auth_Logout**
- **EpicGames_Auth_QueryIdToken**
- **EpicGames_Auth_RemoveNotifyLoginStatusChanged**
- **EpicGames_Auth_VerifyIdToken**

Constants

These are the constants used by this API:

- EAuth Scope Flags
- ELogin Credential Type
- External Credential Type
- Login Status

External Login Flow Guide

This is a detailed login flow for external accounts (the required credentials depend on the **External Credential Type** used with the **EpicGames_Auth_Login** API)

1. Game **EpicGames_Auth_Login** calls with the `EOS_LCT_ExternalAuth` credential type.
2. **EpicGames_Auth_Login** callback returns a status `EpicGames_InvalidUser` with a non-undefined EOS_ContinuanceToken in the EOS_Auth_LoginCallbackInfo data.
3. Game calls **EpicGames_Auth_LinkAccount** with the EOS_ContinuanceToken to initiate the login flow for linking the platform account with the user's Epic account.
4. EOS SDK automatically opens the local default web browser and takes the user to the Epic account portal web page.
    5. The user is able to login to their existing Epic account or create a new account if needed.
    6. In the meantime, EOS SDK will internally keep polling the backend for a completion status of the login flow.
7. Once user completes the login, cancels it or if the login flow times out, **EpicGames_Auth_LinkAccount** invokes the completion callback to the caller.
8. If the user was logged in successfully, `EpicGames_Success` is returned in the callback event. Otherwise, an error result code is returned accordingly.

EpicGames_Auth_AddNotifyLoginStatusChanged

Register to receive login status updates.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Auth_AddNotifyLoginStatusChanged

Syntax:

```
EpicGames_Auth_AddNotifyLoginStatusChanged()
```

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_Auth_AddNotifyLoginStatusChanged"`
CurrentStatus	EpicGames Login Status	The status at the time of the notification
PrevStatus	EpicGames Login Status	The status prior to the change

Example:

```
identifier = EpicGames_Auth_AddNotifyLoginStatusChanged()
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_Auth_AddNotifyLoginStatusChanged")
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Auth_CopyIdToken

Fetch an ID token for an Epic Account ID. ID tokens are used to securely verify user identities with online services. The most common use case is using an ID token to authenticate the local user by their selected account ID, which is the account ID that should be used to access any game-scoped data for the current application. An ID token for the selected account ID of a locally authenticated user will always be readily available. To retrieve it for the selected account ID, you can use EpicGames_Auth_CopyIdToken directly after a successful user login.

EXTERNAL A wrapper around EOS_Auth_CopyIdToken

Syntax:

```
EpicGames_Auth_CopyIdToken(accountID)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the user being queried.

Returns:

```
Struct
```

key	type	Description
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
JsonWebToken	string	The ID token as a Json Web Token (JWT) string.
AccountId	string	The Epic Account ID described by the ID token. Use EpicGames_EpicAccountId_FromString to

		populate this field when validating a received ID token.

Example:

```
var struct = EpicGames_Auth_CopyIdToken(accountID)
if(struct.status == EpicGames_Success)
{
    JsonWebToken = struct.JsonWebToken
}
```

The above code will show an example of how the function should be used. The token associated with the provided account id is returned inside the struct, alongside other useful information.

EpicGames_Auth_CopyUserAuthToken

Fetch a user auth token for an Epic Account ID. A user authentication token allows any code with possession (backend/client) to perform certain actions on behalf of the user. Because of this, for the purposes of user identity verification, the EpicGames_Auth_CopyIdToken should be used instead.

EXTERNAL A wrapper around EOS_Auth_CopyUserAuthToken

Syntax:

```
EpicGames_Auth_CopyUserAuthToken(accountID)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the user being queried

Returns:

```
struct
```

key	type	Description
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
JsonWebToken	string	The ID token as a Json Web Token (JWT) string
AccountId	string	The Epic Account ID associated with this auth token
AccessToken	string	Access token for the current user login session
App	string	Name of the app related to the client ID involved with this token

AuthType	real	Type of auth token (EpicGames_ATT_Client or EpicGames_ATT_User)
ClientId	string	Client ID that requested this token
ExpiresAt	string	Absolute time in UTC before the access token expires, in ISO 8601 format
ExpiresIn	real	Time before the access token expires, in seconds, relative to the call to EpicGames_Auth_CopyUserAuthToken
RefreshToken	string	Refresh token.
RefreshExpiresAt	string	Absolute time in UTC before the refresh token expires, in ISO 8601 format
RefreshExpiresIn	real	Time before the access token expires, in seconds, relative to the call to EpicGames_Auth_CopyUserAuthToken

Example:

```
var struct = EpicGames_Auth_CopyUserAuthToken(accountID)
if(struct.status == EpicGames_Success)
{
     var AccessToken = struct.AccessToken
}
```

The above code will show an example of how the function should be used. The access token associated with the provided account id is returned inside the struct along side other useful information.

EpicGames_Auth_DeletePersistentAuth

Deletes a previously received and locally stored persistent auth access token for the currently logged in user of the local device. On Desktop and Mobile platforms, the access token is deleted from the keychain of the local user and a backend request is made to revoke the token on the authentication server. On Console platforms, even though the caller is responsible for storing and deleting the access token on the local device, this function should still be called with the access token before its deletion to make the best effort in attempting to also revoke it on the authentication server. If the function would fail on Console, the caller should still proceed as normal to delete the access token locally as intended.

EXTERNAL A wrapper around EOS_Auth_DeletePersistentAuth

Syntax:

```
EpicGames_Auth_DeletePersistentAuth(refreshToken)
```

Argument	Type	Description
refreshToken	string	A long-lived refresh token that is used with the `EpicGames_LCT_PersistentAuth` login type and is to be revoked from the authentication server. Only used on Console platforms. On Desktop and Mobile platforms, set this parameter to `undefined`.

Returns:

```
N/A
```

Example:

```
EpicGames_Auth_DeletePersistentAuth(refreshtoken)
```

The above code will show an example of how the function should be used. The refresh token provided will be revoked and invalidated.

EpicGames_Auth_GetLoginStatus

Fetches the login status for an Epic Account ID.

EXTERNAL A wrapper around EOS_Auth_GetLoginStatus

Syntax:

```
EpicGames_Auth_GetLoginStatus(accountId)
```

Argument	Type	Description
accountId	string	The Epic Account ID of the user being queried

Returns:

```
real (EpicGames Login Status)
```

Example:

```
switch(EpicGames_Auth_GetLoginStatus(AccountID))
{
    case EpicGames_LS_NotLoggedIn:
        draw_text(100,190,"LoginStatus: NotLoggedIn");
    break;

    case EpicGames_LS_UsingLocalProfile:
        draw_text(100,190,"LoginStatus: UsingLocalProfile");
    break;

    case EpicGames_LS_LoggedIn:
        draw_text(100,190,"LoginStatus: LoggedIn");
    break;
}
```

The above code will show an example of how the function should be used. A login status constant is returned and checked against the provided builtin constants.

EpicGames_Auth_GetSelectedAccountId

Fetch the selected account ID to the current application for a local authenticated user.

EXTERNAL A wrapper around EOS_Auth_GetSelectedAccountId

Syntax:

```
EpicGames_Auth_GetSelectedAccountId(account)
```

Argument	Type	Description
account	string	The selected account ID corresponding to the given account ID.

Returns:

```
string
```

Example:

```
var _accountID = EpicGames_Auth_GetSelectedAccountId(accountID)
```

The above code will show an example of how the function should be used.

EpicGames_Auth_LinkAccount

Link external account by continuing previous login attempt with a continuance token. On Desktop and Mobile platforms, the user will be presented the Epic Account Portal to resolve their identity. On Console, the user will login to their Epic Account using an external device, e.g. a mobile device or a desktop PC, by browsing to the presented authentication URL and entering the device code presented by the game on the console. On success, the user will be logged in at the completion of this action. This will commit this external account to the Epic Account and cannot be undone in the SDK.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Auth_LinkAccount

Syntax:

```
EpicGames_Auth_LinkAccount(accountID, scope_flags)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the logged in local user whose Epic Account will be linked with the local Nintendo NSA ID Account. By default set to `undefined`.
scope_flags	EAuth Scope Flags	Combination of the enumeration flags to specify how the account linking operation will be performed.

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_Auth_LinkAccount"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
AccountID	string	The epic account ID used upon calling the function that generated this callback.
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_Auth_Login(
                accountID,
                EpicGames_AS_BasicProfile | EpicGames_AS_FriendsList |
EpicGames_AS_Presence)
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_Auth_LinkAccount")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Auth_Login

Login/Authenticate with user credentials.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Auth_Login

Syntax:

```
EpicGames_Auth_Login(type, scope_flags, id, token, external_type)
```

Argument	Type	Description
type	ELogin Credential Type	Type of login. Needed to identify the auth method to use
scope_flags	EAuth Scope Flags	Auth scope flags are permissions to request from the user while they are logging in. This is a bitwise-or union (|, pipe symbol) of EAuth Scope Flags.
id	string	ID of the user logging in, based on ELogin Credential Type
token	string	Credentials or token related to the user logging in
external_type	External Credential Type	Type of external login. Needed to identify the external auth method to use. Used when login type is set to `EpicGames_LCT_ExternalAuth`, ignored otherwise (see External Login Flow Guide for more details)

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_Auth_Login"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID
AccountID	string	The Epic Account ID of the local user who has logged in
SelectedAccountId	string	The Epic Account ID that has been previously selected to be used for the current application. Applications should use this ID to authenticate with online backend services that store game-scoped data for users. Only when status is success OPTIONAL

Example:

```
identifier = EpicGames_Auth_Login(
                EpicGames_LCT_ExchangeCode,
                EpicGames_AS_BasicProfile | EpicGames_AS_FriendsList |
EpicGames_AS_Presence,
                "",
                code,
                noone)
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_Auth_Login")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
```

```
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Auth_Logout

Signs the player out of the online service.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Auth_Logout

Syntax:

```
EpicGames_Auth_Logout(accountID)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the local user who is being logged out

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_Auth_Logout"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the *status* code

identifier	real	The asynchronous listener ID

Example:

```
identifier = EpicGames_Auth_Logout(accountID)
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_Auth_Logout")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Auth_QueryIdToken

Query the backend for an ID token that describes one of the merged account IDs of a local authenticated user. The ID token can be used to impersonate a merged account ID when communicating with online services. An ID token for the selected account ID of a locally authenticated user will always be readily available and does not need to be queried explicitly.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Auth_QueryIdToken

Syntax:

```
EpicGames_Auth_QueryIdToken(accountID, accountID_target)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the local authenticated user.
accountID_target	string	The target Epic Account ID for which to query an ID token. This account id may be the same as the input LocalUserId or another merged account id associated with the local user's Epic account. An ID token for the selected account id of a locally authenticated user will always be readily available. To retrieve it for the selected account ID, you can use EpicGames_Auth_CopyIdToken directly after a successful user login.

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_Auth_QueryIdToken"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_Auth_QueryIdToken(accountID,accountID)
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_Auth_QueryIdToken")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Auth_RemoveNotifyLoginStatusChanged

Unregister from receiving login status updates.

EXTERNAL A wrapper around EOS_Auth_RemoveNotifyLoginStatusChanged

Syntax:

```
EpicGames_Auth_RemoveNotifyLoginStatusChanged(id)
```

Argument	Type	Description
id	real	handle ID representing the registered callback (from EpicGames_Auth_AddNotifyLoginStatusChanged)

Returns:

```
N/A
```

Example:

```
var handle = EpicGames_Auth_AddNotifyLoginStatusChanged()
//...
//...later
//...
EpicGames_Auth_RemoveNotifyLoginStatusChanged(handle)
```

The code sample above enables the login status notifications (EpicGames_Auth_AddNotifyLoginStatusChanged) and later disables them by refering to the previous generated handle.

Login Status

These constants are used to describe the Login status of a given account or connection and are returned by the following functions:

- EpicGames_Auth_AddNotifyLoginStatusChanged CALLBACK
- EpicGames_Auth_GetLoginStatus
- EpicGames_Connect_AddNotifyLoginStatusChanged CALLBACK
- EpicGames_Connect_GetLoginStatus

Login Status Constant	Description
`EpicGames_LS_NotLoggedIn`	Player has not logged in or chosen a local profile
`EpicGames_LS_UsingLocal`	Player is using a local profile but is not logged in
`EpicGames_LS_LoggedIn`	Player has been validated by the platform specific authentication service

External Credential Type

List of the supported identity providers to authenticate a user. The type of authentication token is specific to each provider. Tokens in string format should be passed as-is to the function.

Argument	Description
`EpicGames_ECT_EPIC`	Epic Account Services Token Using ID Token is preferred, retrieved with EpicGames_Auth_CopyIdToken that returns **JsonWebToken**. Using Auth Token is supported for backwards compatibility, retrieved with EpicGames_Auth_CopyUserAuthToken that returns **AccessToken**. Supported with EpicGames_Connect_Login.
`EpicGames_ECT_STEAM_APP_TICKET`	Steam Encrypted App Ticket Generated using the **RequestEncryptedAppTicket** API of Steamworks SDK. The retrieved App is then passed into the EpicGames_Auth_Login or EpicGames_Connect_Login APIs.
`EpicGames_ECT_PSN_ID_TOKEN`	PlayStation(TM)Network ID Token Retrieved from the PlayStation(R) SDK. Please see first-party documentation for additional information. Supported with EpicGames_Auth_Login, EpicGames_Connect_Login.
`EpicGames_ECT_XBL_XSTS_TOKEN`	Xbox Live XSTS Token Retrieved from the GDK and XDK. Please see first-party documentation for additional information. Supported with EpicGames_Auth_Login, EpicGames_Connect_Login.
`EpicGames_ECT_DISCORD_ACCESS_TOKEN`	Discord Access Token Retrieved using the **GetOAuth2Token** API of Discord SDK. Supported with EpicGames_Auth_Login.
`EpicGames_ECT_GOG_SESSION_TICKET`	GOG Galaxy Encrypted App Ticket Generated using the **RequestEncryptedAppTicket** API of GOG Galaxy SDK. The retrieved App is then passed into the EpicGames_Auth_Login API.

`EpicGames_ECT_NINTENDO_ID_TOKEN`	Nintendo Account ID Token Identifies a Nintendo user account and is acquired through web flow authentication where the local user logs in using their email address/sign-in ID and password. This is the common Nintendo account that users login with outside the Nintendo Switch device. Supported with EpicGames_Auth_Login, EpicGames_Connect_Login.
`EpicGames_ECT_NINTENDO_NSA_ID_TOKEN`	Nintendo Service Account ID Token (NSA ID) The NSA ID identifies uniquely the local Nintendo Switch device. The authentication token is acquired locally without explicit user credentials. As such, it is the primary authentication method for seamless login on Nintendo Switch. The NSA ID is not exposed directly to the user and does not provide any means for login outside the local device. Because of this, Nintendo Switch users will need to link their Nintendo Account or another external user account to their Product User ID in order to share their game progression across other platforms. Otherwise, the user will not be able to login to their existing Product User ID on another platform due to missing login credentials to use. It is recommended that the game explicitly communicates this restriction to the user so that they will know to add the first linked external account on the Nintendo Switch device and then proceed with login on another platform. In addition to sharing cross-platform game progression, linking the Nintendo Account or another external account will allow preserving the game progression permanently. Otherwise, the game progression will be tied only to the local device. In case the user loses access to their local device, they will not be able to recover the game progression if it is only associated with this account type. Supported with EpicGames_Auth_Login, EpicGames_Connect_Login.
`EpicGames_ECT_UPLAY_ACCESS_TOKEN`	Uplay Access Token
`EpicGames_ECT_OPENID_ACCESS_TOKEN`	OpenID Provider Access Token Supported with EpicGames_Connect_Login.

`EpicGames_ECT_DEVICEID_ACCESS_TOKEN`	Device ID access token that identifies the current locally logged in user profile on the local device. The local user profile here refers to the operating system user login, for example the user's Windows Account or on a mobile device the default active user profile. This credential type is used to automatically login the local user using the EOS Connect Device ID feature. The intended use of the Device ID feature is to allow automatically logging in the user on a mobile device and to allow playing the game without requiring the user to necessarily login using a real user account at all. This makes a seamless first-time experience possible and allows linking the local device with a real external user account at a later time, sharing the same EpicGames_ProductUserId that is being used with the Device ID feature. Supported with EpicGames_Connect_Login.
`EpicGames_ECT_APPLE_ID_TOKEN`	Apple ID Token; Supported with EpicGames_Connect_Login.
`EpicGames_ECT_GOOGLE_ID_TOKEN`	Google ID Token; Supported with EpicGames_Connect_Login.
`EpicGames_ECT_OCULUS_USERID_NONCE`	Oculus User ID and Nonce (call **ovr_User_GetUserProof**, to retrieve the nonce). Then pass the local User ID and the Nonce as a "{UserID}|{Nonce}" formatted string for the EpicGames_Connect_Login Token parameter. Note that in order to successfully retrieve a valid non-zero id for the local user using **ovr_User_GetUser**, your Oculus App needs to be configured in the Oculus Developer Dashboard to have the User ID feature enabled. Supported with EpicGames_Connect_Login.
`EpicGames_ECT_ITCHIO_JWT`	itch.io JWT Access Token. Use the itch.io app manifest to receive a JWT access token for the local user via the ITCHIO_API_KEY process environment variable. The itch.io access token is valid for 7 days after which the game needs to be restarted by the user as otherwise EOS Connect authentication session can no longer be refreshed. Supported with EpicGames_Connect_Login.

`EpicGames_ECT_ITCHIO_KEY`	itch.io Key Access Token. This access token type is retrieved through the OAuth 2.0 authentication flow for the itch.io application. Supported with **EpicGames_Connect_Login**.
`EpicGames_ECT_EPIC_ID_TOKEN`	Epic Games ID Token. Acquired using **EpicGames_Auth_CopyIdToken** that returns **JsonWebToken**. Supported with **EpicGames_Connect_Login**.
`EpicGames_ECT_AMAZON_ACCESS_TOKEN`	Amazon Access Token. Supported with **EpicGames_Connect_Login**.

ELogin Credential Type

These constants represent all possible types of login methods, availability depends on permissions granted to the client.

Argument	Description
`EpicGames_LCT_Password`	The argument to be passed in
`EpicGames_LCT_ExchangeCode`	A short-lived one-time use exchange code to login the local user. When started, the application is expected to consume the exchange code by using the EpicGames_Auth_Login API as soon as possible. This is needed in order to authenticate the local user before the exchange code would expire. Attempting to consume an already expired exchange code will return `AuthExchangeCodeNotFound` error by the EpicGames_Auth_Login API.
`EpicGames_LCT_PersistentAuth`	Desktop and Mobile only; deprecated on Console platforms in favor of EOS_LCT_ExternalAuth login method. Long-lived access token that is stored on the local device to allow persisting a user login session over multiple runs of the application. When using this login type, if an existing access token is not found or it is invalid or otherwise expired, the error result `EpicGames_InvalidAuth` is returned. On Console platforms, after a successful login using the `EOS_LCT_DeviceCode` login type, the persistent access token is retrieved using the EpicGames_Auth_CopyUserAuthToken API and stored by the application for the currently logged in user of the local device.

`EpicGames_LCT_DeviceCode`	Deprecated and no longer used.
`EpicGames_LCT_Developer`	Login with named credentials hosted by the EOS SDK Developer Authentication Tool.
`EpicGames_LCT_RefreshToken`	Refresh token that was retrieved from a previous call to **EpicGames_Auth_Login** API in another local process context. Mainly used in conjunction with custom launcher applications. in-between that requires authenticating the user before eventually starting the actual game client application. In such scenario, an intermediate launcher will log in the user by consuming the exchange code it received from the Epic Games Launcher. To allow the game client to also authenticate the user, it can copy the refresh token using the **EpicGames_Auth_CopyUserAuthToken** API and pass it via launch parameters to the started game client. The game client can then use the refresh token to log in the user.
`EpicGames_LCT_AccountPortal`	Desktop and Mobile only. Initiate a login through the Epic account portal. for example when starting the application through a proprietary ecosystem launcher or otherwise.
`EpicGames_LCT_ExternalAuth`	Login using external account provider credentials, such as Steam, PlayStation(TM)Network, Xbox Live, or Nintendo. This is the intended login method on Console. On Desktop and Mobile, used when launched through any of the commonly supported platform clients (see the **External Login Flow Guide** for more details)

EAuth Scope Flags

List of the supported scope flags associated with the login API calls:

- EpicGames_Auth_Login
- EpicGames_Auth_LinkAccount

EAuth Scope Flags	Description
`EpicGames_AS_NoFlags`	
`EpicGames_AS_BasicProfile`	Permissions to see your account ID, display name, language and country
`EpicGames_AS_FriendsList`	Permissions to see a list of your friends who use this application
`EpicGames_AS_Presence`	Permissions to set your online presence and see presence of your friends
`EpicGames_AS_FriendsManagement`	Permissions to manage the Epic friends list. This scope is restricted to Epic first party products, and attempting to use it will result in authentication failures.
`EpicGames_AS_Email`	Permissions to see email in the response when fetching information for a user. This scope is restricted to Epic first party products, and attempting to use it will result in authentication failures.

Connect

The **Connect Interface** enables an external identity provider to integrate with and use the **Epic Online Services** (EOS) ecosystem.

Functions

These functions are provided for handling connectivity:

- **EpicGames_Connect_AddNotifyAuthExpiration**
- **EpicGames_Connect_AddNotifyLoginStatusChanged**
- **EpicGames_Connect_CopyIdToken**
- **EpicGames_Connect_CopyProductUserInfo**
- **EpicGames_Connect_CreateUser**
- **EpicGames_Connect_GetLoginStatus**
- **EpicGames_Connect_Login**
- **EpicGames_Connect_RemoveNotifyAuthExpiration**
- **EpicGames_Connect_RemoveNotifyLoginStatusChanged**

Structures

These are the structures used by this API:

- **ExternalAccountInfo**

EpicGames_Connect_AddNotifyAuthExpiration

Register to receive upcoming authentication expiration notifications. Notification is approximately 10 minutes prior to expiration. Call EpicGames_Connect_Login again with valid third party credentials to refresh access.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Connect_AddNotifyAuthExpiration

Syntax:

```
EpicGames_Connect_AddNotifyAuthExpiration()
```

Returns:

```
N/A
```

Triggers:

```
Asynchronous Social Event
```

Key	Type	Description
type	string	The string `"EpicGames_Connect_AddNotifyAuthExpiration"`
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_Connect_AddNotifyAuthExpiration()
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_Connect_AddNotifyAuthExpiration")
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Connect_AddNotifyLoginStatusChanged

Register to receive user login status updates.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Connect_AddNotifyLoginStatusChanged

Syntax:

```
EpicGames_Connect_AddNotifyLoginStatusChanged()
```

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_Connect_AddNotifyLoginStatusChanged"`
CurrentStatus	EpicGames Login Status	The status at the time of the notification
PrevStatus	EpicGames Login Status	The status prior to the change

Example:

```
identifier = EpicGames_Connect_AddNotifyLoginStatusChanged()
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_Connect_AddNotifyLoginStatusChanged")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Connect_CopyIdToken

Fetches an ID token for a Product User ID.

EXTERNAL A wrapper around EOS_Connect_CopyIdToken

Syntax:

```
EpicGames_Connect_CopyIdToken(user)
```

Argument	Description
user	The local Product User ID whose ID token should be copied.

Returns:

```
struct
```

keys	type	Description
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
JsonWebToken	string	The ID token as a Json Web Token (JWT) string.
ProductUserId	string	The Product User ID described by the ID token.

Example:

```
var struct = EpicGames_Connect_CopyIdToken(user)
if(struct.status = EpicGames_Success)
{
    JsonWebToken = struct.JsonWebToken
}
```

The above code will show an example of how the function should be used. The JWT associated with the provided product user id is returned inside the struct, alongside other useful information.

EpicGames_Connect_CopyProductUserInfo

Fetch information about a Product User, using the external account that they most recently logged in with as the reference.

EXTERNAL A wrapper around EOS_Connect_CopyProductUserInfo

Syntax:

```
EpicGames_Connect_CopyProductUserInfo(userID_target)
```

Argument	Description
userID_target	Product user ID to look for when copying external account info from the cache.

Returns:

```
struct (ExternalAccountInfo)
```

Example:

```
var struct = EpicGames_Connect_CopyProductUserInfo(userID_target)
if(struct.status = EpicGames_Success)
{
    // access the data here
}
```

The above code will show an example of how the function should be used.

EpicGames_Connect_CreateUser

Create an account association with the Epic Online Service as a product user given their external auth credentials.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Connect_CreateUser

Syntax:

```
EpicGames_Connect_CreateUser()
```

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

Key	Type	Description
type	string	The string `"EpicGames_Connect_CreateUser"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.
LocalUserId	string	If the operation succeeded, this is the Product User ID of the local user who was created. OPTIONAL

Example:

```
identifier = EpicGames_Connect_CreateUser()
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_Connect_CreateUser")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Connect_GetLoginStatus

Fetches the login status for an Product User ID. This Product User ID is considered logged in as long as the underlying access token has not expired.

EXTERNAL A wrapper around EOS_Connect_GetLoginStatus

Syntax:

```
EpicGames_Connect_GetLoginStatus(user)
```

Argument	Type	Description
user	string	The Product User ID of the user being queried.

Returns:

```
(real) EpicGames Login Status
```

Example:

```
if(EpicGames_Connect_GetLoginStatus(user) == EpicGames_LS_LoggedIn)
    show_debug_message(user + ": is logged")
else
    show_debug_message(user + ": not logged")
```

The above code will show an example of how the function should be used. A login status constant is returned and checked against the provided builtin constants.

EpicGames_Connect_Login

Login/Authenticate given a valid set of external auth credentials.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Connect_Login

Syntax:

```
EpicGames_Connect_Login(type, access_token, display_name)
```

Argument	Type	Description
type	External Credential Type	Type of external login; identifies the auth method to use.
access_token	string	External token associated with the user logging in.
display_name	string	The user's display name on the identity provider systems.

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

Key	Type	Description
type	string	The string `"EpicGames_Connect_Login"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the

		operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.
LocalUserId	string	If the operation succeeded, this is the Product User ID of the local user who logged in. OPTIONAL

Example:

```
identifier = EpicGames_Connect_Login()
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_Connect_Login")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Connect_RemoveNotifyAuthExpiration

Unregister from receiving expiration notifications.

EXTERNAL A wrapper around EOS_Connect_RemoveNotifyAuthExpiration

Syntax:

```
EpicGames_Connect_RemoveNotifyAuthExpiration(id)
```

Argument	Type	Description
id	real	The handle representing the registered callback (return by EpicGames_Connect_AddNotifyAuthExpiration)

Returns:

```
N/A
```

Example:

```
AddNotifyAuthExpiration_id = EpicGames_Connect_AddNotifyAuthExpiration()
//...
//...Later
//...
EpicGames_Connect_RemoveNotifyAuthExpiration(AddNotifyAuthExpiration_id)
```

The code sample above enables the auth expiration notifications (EpicGames_Connect_AddNotifyAuthExpiration) and later disables them by refering to the previous generated handle.

EpicGames_Connect_RemoveNotifyLoginStatusChanged

Unregister from receiving user login status updates.

EXTERNAL A wrapper around EOS_Connect_RemoveNotifyLoginStatusChanged

Syntax:

```
EpicGames_Connect_RemoveNotifyLoginStatusChanged(Id)
```

Argument	Type	Description
Id	real	The handle representing the registered callback (return by EpicGames_Connect_AddNotifyLoginStatusChanged)

Returns:

```
N/A
```

Example:

```
NotifyLoginStatusChanged_id = EpicGames_Connect_AddNotifyLoginStatusChanged()
//...
//...Later
//...
EpicGames_Connect_RemoveNotifyLoginStatusChanged(NotifyLoginStatusChanged_id)
```

The code sample above enables the login status changed notifications (EpicGames_Connect_AddNotifyLoginStatusChanged) and later disables them by refering to the previous generated handle.

Friends

Playing games with your friends and meeting new players online are important parts of many online services. The **Epic Online Services** (EOS) SDK uses the **Friends Interface** to retrieve the friends lists for a logged-in user.

Friends lists are stored by the online service's servers, and can change during a session as friends are added or removed or if friends grant or revoke consent for the game to use their information.

Functions

These functions are provided for handling friend lists:

- **EpicGames_Friends_AcceptInvite**
- **EpicGames_Friends_AddNotifyFriendsUpdate**
- **EpicGames_Friends_GetFriendAtIndex**
- **EpicGames_Friends_GetFriendsCount**
- **EpicGames_Friends_GetStatus**
- **EpicGames_Friends_QueryFriends**
- **EpicGames_Friends_RejectInvite**
- **EpicGames_Friends_RemoveNotifyFriendsUpdate**
- **EpicGames_Friends_SendInvite**

Constants

These are the constants used by this API:

- **EpicGames Friendship Status**

EpicGames_Friends_AcceptInvite

Starts an asynchronous task that accepts a friend invitation from another user. The completion delegate is executed after the backend response has been received.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Friends_AcceptInvite

Syntax:

```
EpicGames_Friends_AcceptInvite(accountID, accountID_target)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the local, logged-in user who is accepting the friends list invitation
accountID_target	string	The Epic Account ID of the user who sent the friends list invitation

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_Friends_AcceptInvite"`

status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_Friends_AcceptInvite(accountID, accountID_target)
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_Friends_AcceptInvite")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Friends_AddNotifyFriendsUpdate

Listen for changes to friends for a particular account.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Friends_AddNotifyFriendsUpdate

Syntax:

```
EpicGames_Friends_AddNotifyFriendsUpdate()
```

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_Friends_AddNotifyFriendsUpdate"`
CurrentStatus	EpicGames Friendship Status	The current status of the user.
PreviousStatus	EpicGames Friendship Status	The previous status of the user.
TargetUserId	string	The Epic Account ID of the user whose status is being updated.

LocalUserId	string	The Epic Account ID of the local user who is receiving the update

Example:

```
identifier = EpicGames_Friends_AddNotifyFriendsUpdate()
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_Friends_AddNotifyFriendsUpdate")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Friends_GetFriendAtIndex

Retrieves the Epic Account ID of an entry from the friends list that has already been cached. The Epic Account ID returned by this function may belong to an account that has been invited to be a friend or that has invited the local user to be a friend. To determine if the Epic Account ID returned by this function is a friend or a pending friend invitation, use the EpicGames_Friends_GetStatus function.

NOTE Requires a previous call to EpicGames_Friends_QueryFriends to store values in cache.

EXTERNAL A wrapper around EOS_Friends_GetFriendAtIndex

Syntax:

```
EpicGames_Friends_GetFriendAtIndex(accountID, index)
```

Argument	Type	Description
accountID	string	The user account identifier to get the friend data from.
index	real	Index into the friend list. This value must be between 0 and EpicGames_Friends_GetFriendsCount() - 1 inclusively.

Returns:

```
string
```

Example:

```
var count = EpicGames_Friends_GetFriendsCount(accountID)
for(var i = 0 ; i < count; i++)
{
```

```
    var friend_account = EpicGames_Friends_GetFriendAtIndex(accountID,i)
}
```

The above code will show an example of how the function should be used. The friends data is returned providing an index.

EpicGames_Friends_GetFriendsCount

Retrieves the number of friends on the friends list.

NOTE Requires a previous call to EpicGames_Friends_QueryFriends to store values in cache.

EXTERNAL A wrapper around EOS_Friends_GetFriendsCount

Syntax:

```
EpicGames_Friends_GetFriendsCount(accountID)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the user whose friends should be counted

Returns:

```
real
```

Example:

```
var account = EpicGames_Friends_GetFriendsCount(accountID)
for(var i = 0 ; i < account ; i++)
{
    var friend_account = EpicGames_Friends_GetFriendAtIndex(accountID,i)
}
```

The above code will show an example of how the function should be used. After a successcull call to EpicGames_Friends_QueryFriends, the function EpicGames_Friends_GetFriendsCount will return the number of entries in the query array which can then be accessed using the EpicGames_Friends_GetFriendAtIndex function.

EpicGames_Friends_GetStatus

Retrieve the friendship status between the local user and another user.

EXTERNAL A wrapper around EOS_Friends_GetStatus

Syntax:

```
EpicGames_Friends_GetStatus(accountID, accountID_target)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the local, logged in user
accountID_target	string	The Epic Account ID of the user whose friendship status with the local user is being queried

Returns:

```
real (EpicGames Friendship Status)
```

Example:

```
if(EpicGames_Friends_GetStatus(accountID,accountID_target) == EpicGames_FS_Friends)
{
    show_debug_message("It's my friend!!!")
}
else
{
    show_debug_message("Not my friend :(")
}
```

The above code will show an example of how the function should be used. The friendship status is returned from the function call.

EpicGames_Friends_QueryFriends

Starts an asynchronous task that reads the user's friends list from the backend service, caching it for future use.

Once the callback has been fired with a successful EpicGames Result, it is possible to call one of the following functions:

- EpicGames_Friends_GetFriendAtIndex
- EpicGames_Friends_GetFriendsCount

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Friends_QueryFriends

Syntax:

```
EpicGames_Friends_QueryFriends(accountID)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the local, logged-in user whose friends list you want to retrieve

Returns:

```
N/A
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents

Key	Type	Description
type	string	The string `"EpicGames_Friends_QueryFriends"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_Friends_QueryFriends(accountID)
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_Friends_QueryFriends")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Friends_RejectInvite

Starts an asynchronous task that rejects a friend invitation from another user. The completion delegate is executed after the backend response has been received.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Friends_RejectInvite

Syntax:

```
EpicGames_Friends_RejectInvite(accountID, accountID_target)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the local, logged-in user who is rejecting a friends list invitation
accountID_target	string	The Epic Account ID of the user who sent the friends list invitation

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_Friends_RejectInvite"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the

		operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_Friends_RejectInvite(accountID, accountID_target)
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_Friends_RejectInvite")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Friends_RemoveNotifyFriendsUpdate

Stop listening for friends changes on a previously bound handler.

EXTERNAL A wrapper around EOS_Friends_RemoveNotifyFriendsUpdate

Syntax:

```
EpicGames_Friends_RemoveNotifyFriendsUpdate(id)
```

Argument	Type	Description
id	real	The handle representing the registered callback (return by EpicGames_Friends_AddNotifyFriendsUpdate)

Returns:

```
N/A
```

Example:

```
handle = EpicGames_Friends_AddNotifyFriendsUpdate()
//...
//...later
//...
EpicGames_Friends_RemoveNotifyFriendsUpdate(handle)
```

The code sample above enables the friend update notifications (EpicGames_Friends_AddNotifyFriendsUpdate) and later disables them by refering to the previous generated handle.

EpicGames_Friends_SendInvite

Starts an asynchronous task that sends a friend invitation to another user. The completion delegate is executed after the backend response has been received. It does not indicate that the target user has responded to the friend invitation.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Friends_SendInvite

Syntax:

```
EpicGames_Friends_SendInvite(accountID,accountID_target)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the local, logged-in user who is sending the friends list invitation
accountID_target	string	The Epic Account ID of the user who is receiving the friends list invitation

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_Friends_SendInvite"`

status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_Friends_SendInvite(accountID, accountID_target)
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_Friends_SendInvite")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames Friendship Status

These contants are used to describe the friendship status with a given account and are returned by the following functions:

- EpicGames_Friends_AddNotifyFriendsUpdate CALLBACK
- EpicGames_Friends_GetStatus

EpicGames Friendship Status Constant	Description
`EpicGames_FS_NotFriends`	The two accounts have no friendship status
`EpicGames_FS_InviteSent`	The local account has sent a friend invite to the other account
`EpicGames_FS_InviteRecieved`	The other account has sent a friend invite to the local account
`EpicGames_FS_Friends`	The accounts have accepted friendship

Leaderboards

The **Leaderboards Interface** gives developers using **Epic Online Services** (EOS) the ability to rank scores from their entire player base, so that players can compete with their friends or other players worldwide for the top score. Each game can support multiple leaderboards, collecting scores from different sources, and ranking them with different scoring modes.

Functions

These functions are provided for handling leaderboards:

- **EpicGames_Leaderboards_CopyLeaderboardDefinitionByIndex**
- **EpicGames_Leaderboards_CopyLeaderboardDefinitionByLeaderboardId**
- **EpicGames_Leaderboards_CopyLeaderboardRecordByIndex**
- **EpicGames_Leaderboards_CopyLeaderboardRecordByUserId**
- **EpicGames_Leaderboards_CopyLeaderboardUserScoreByIndex**
- **EpicGames_Leaderboards_CopyLeaderboardUserScoreByUserId**
- **EpicGames_Leaderboards_GetLeaderboardDefinitionCount**
- **EpicGames_Leaderboards_GetLeaderboardRecordCount**
- **EpicGames_Leaderboards_GetLeaderboardUserScoreCount**
- **EpicGames_Leaderboards_QueryLeaderboardDefinitions**
- **EpicGames_Leaderboards_QueryLeaderboardRanks**
- **EpicGames_Leaderboards_QueryLeaderboardUserScores**

Structures

These are the structures used by this API:

- **LeaderboardDefinition**
- **LeaderboardRecord**

- LeaderboardUserScore

EpicGames_Leaderboards_CopyLeaderboardDefinitionByIndex

Fetches a leaderboard definition from the cache using an index.

NOTE Requires a previous call to EpicGames_Leaderboards_QueryLeaderboardDefinitions to store values in cache.

EXTERNAL A wrapper around EOS_Leaderboards_CopyLeaderboardDefinitionByIndex

Syntax:

```
EpicGames_Leaderboards_CopyLeaderboardDefinitionByIndex(index)
```

Argument	Type	Description
index	real	Index of the leaderboard definition to retrieve from the cache

Returns:

```
struct (LeaderboardDefinition)
```

Example:

```
var count = EpicGames_Leaderboards_GetLeaderboardDefinitionCount()
for(var i = 0 ; i < count ; i++)
{
    var struct = EpicGames_Leaderboards_CopyLeaderboardDefinitionByIndex(i)
    var LeaderboardId = struct.LeaderboardId
}
```

The above code will show an example of how the function should be used. The leaderboard definition data is returned providing an leaderboard index.

EpicGames_Leaderboards_CopyLeaderboardDefinitionByLeaderboardId

Fetches a leaderboard definition from the cache using a leaderboard ID.

NOTE Requires a previous call to EpicGames_Leaderboards_QueryLeaderboardDefinitions to store values in cache.

EXTERNAL A wrapper around EOS_Leaderboards_CopyLeaderboardDefinitionByLeaderboardId

Syntax:

```
EpicGames_Leaderboards_CopyLeaderboardDefinitionByLeaderboardId(leaderboardID)
```

Argument	Type	Description
leaderboardID	string	The ID of the leaderboard whose definition you want to copy from the cache

Returns:

```
struct (LeaderboardDefinition)
```

Example:

```
var struct =
EpicGames_Leaderboards_CopyLeaderboardDefinitionByLeaderboardId("MyLeaderboard")
if(struct.status == EpicGames_Success)
{
    var LeaderboardId = struct.LeaderboardId
}
```

The above code will show an example of how the function should be used. The leaderboard definition data is returned providing an leaderboard id.

EpicGames_Leaderboards_CopyLeaderboardRecordByIndex

Fetches a leaderboard record from a given index.

NOTE Requires a previous call to EpicGames_Leaderboards_QueryLeaderboardRanks to store values in cache.

EXTERNAL A wrapper around EOS_Leaderboards_CopyLeaderboardRecordByIndex

Syntax:

```
EpicGames_Leaderboards_CopyLeaderboardRecordByIndex(index)
```

Argument	Type	Description
index	real	Index of the leaderboard record to retrieve from the cache

Returns:

```
struct (LeaderboardRecord)
```

Example:

```
var count = EpicGames_Leaderboards_GetLeaderboardRecordCount()
for(var i = 0 ; i < count ; i++)
{
    var struct = EpicGames_Leaderboards_CopyLeaderboardRecordByIndex(i)
    var Rank = struct.Rank
}
```

The above code will show an example of how the function should be used. The leaderboard record data is returned providing an leaderboard index.

EpicGames_Leaderboards_CopyLeaderboardRecordByUserId

Fetches a leaderboard record from a given user ID.

NOTE Requires a previous call to EpicGames_Leaderboards_QueryLeaderboardRanks to store values in cache.

EXTERNAL A wrapper around EOS_Leaderboards_CopyLeaderboardRecordByUserId

Syntax:

```
EpicGames_Leaderboards_CopyLeaderboardRecordByUserId(userId)
```

Argument	Type	Description
userId	string	Leaderboard data will be copied from the cache if it relates to the user matching this Product User ID

Returns:

```
struct (LeaderboardRecord)
```

Example:

```
var struct = EpicGames_Leaderboards_CopyLeaderboardRecordByUserId("MyLeaderboard")
if(struct.status == EpicGames_Success)
{
    var Rank = struct.Rank
}
```

The above code will show an example of how the function should be used. The leaderboard record data is returned providing an user id.

EpicGames_Leaderboards_CopyLeaderboardUserScoreByIndex

Fetches leaderboard user score from a given index.

NOTE Requires a previous call to EpicGames_Leaderboards_QueryLeaderboardUserScores to store values in cache.

EXTERNAL A wrapper around EOS_Leaderboards_CopyLeaderboardUserScoreByIndex

Syntax:

```
EpicGames_Leaderboards_CopyLeaderboardUserScoreByIndex(index, statName)
```

Argument	Type	Description
index	real	Index of the sorted leaderboard user score to retrieve from the cache.
statName	string	Name of the stat used to rank the leaderboard.

Returns:

```
struct (LeaderboardUserScore)
```

Example:

```
var count = EpicGames_Leaderboards_GetLeaderboardUserScoreCount()
for(var i = 0 ; i < count ; i++)
{
    var struct = EpicGames_Leaderboards_CopyLeaderboardUserScoreByIndex(i)
    var Score = struct.Score
}
```

The above code will show an example of how the function should be used. The leaderboard user score is returned providing an index.

EpicGames_Leaderboards_CopyLeaderboardUserScoreByUserId

Fetches leaderboard user score from a given user ID.

NOTE Requires a previous call to EpicGames_Leaderboards_QueryLeaderboardUserScores to store values in cache.

EXTERNAL A wrapper around EOS_Leaderboards_CopyLeaderboardUserScoreByUserId

Syntax:

```
EpicGames_Leaderboards_CopyLeaderboardUserScoreByUserId(userId, statName)
```

Argument	Type	Description
userId	string	The Product User ID to look for when copying leaderboard score data from the cache
statName	string	The name of the stat that is used to rank this leaderboard

Returns:

```
struct (LeaderboardUserScore)
```

Example:

```
var struct = EpicGames_Leaderboards_CopyLeaderboardUserScoreByUserId("MyLeaderboard")
if(struct.status == EpicGames_Success)
{
    var Score = struct.Score
}
```

The above code will show an example of how the function should be used. The leaderboard user score is returned providing an user id.

EpicGames_Leaderboards_GetLeaderboardDefinitionCount

Fetch the number of leaderboards definitions that are cached locally.

NOTE Requires a previous call to EpicGames_Leaderboards_QueryLeaderboardDefinitions to store values in cache.

EXTERNAL A wrapper around EOS_Leaderboards_GetLeaderboardDefinitionCount

Syntax:

```
EpicGames_Leaderboards_GetLeaderboardDefinitionCount()
```

Returns:

```
real
```

Example:

```
var count = EpicGames_Leaderboards_GetLeaderboardDefinitionCount()
for(var i = 0 ; i < count ; i++)
{
    var struct = EpicGames_Leaderboards_CopyLeaderboardDefinitionByIndex(i)
    var LeaderboardId = struct.LeaderboardId
}
```

The above code will show an example of how the function should be used. After a successcull call to EpicGames_Leaderboards_QueryLeaderboardDefinitions, the function EpicGames_Leaderboards_GetLeaderboardDefinitionCount will return the number of entries in the query array which can then be accessed using theEpicGames_Leaderboards_CopyLeaderboardDefinitionByIndex function.

EpicGames_Leaderboards_GetLeaderboardRecordCount

Fetch the number of leaderboard records that are cached locally.

NOTE Requires a previous call to EpicGames_Leaderboards_QueryLeaderboardRanks to store values in cache.

EXTERNAL A wrapper around EOS_Leaderboards_GetLeaderboardRecordCount

Syntax:

```
EpicGames_Leaderboards_GetLeaderboardRecordCount()
```

Returns:

```
real
```

Example:

```
var count = EpicGames_Leaderboards_GetLeaderboardRecordCount()
for(var i = 0 ; i < count ; i++)
{
    var struct = EpicGames_Leaderboards_CopyLeaderboardRecordByIndex(i)
    var Rank = struct.Rank
}
```

The above code will show an example of how the function should be used. After a successcull call to EpicGames_Leaderboards_QueryLeaderboardRanks, the function EpicGames_Leaderboards_GetLeaderboardRecordCount will return the number of entries in the query array which can then be accessed using the EpicGames_Leaderboards_CopyLeaderboardRecordByIndex function.

EpicGames_Leaderboards_GetLeaderboardUserScoreCount

Fetch the number of leaderboard user scores that are cached locally.

NOTE Requires a previous call to EpicGames_Leaderboards_QueryLeaderboardUserScores to store values in cache.

EXTERNAL A wrapper around EOS_Leaderboards_GetLeaderboardUserScoreCount

Syntax:

```
EpicGames_Leaderboards_GetLeaderboardUserScoreCount()
```

Returns:

```
real
```

Example:

```
var count = EpicGames_Leaderboards_GetLeaderboardUserScoreCount()
for(var i = 0 ; i < count ; i++)
{
    var struct = EpicGames_Leaderboards_CopyLeaderboardUserScoreByIndex(i)
    var Score = struct.Score
}
```

The above code will show an example of how the function should be used. After a successcull call to EpicGames_Leaderboards_QueryLeaderboardUserScores, the function EpicGames_Leaderboards_GetLeaderboardUserScoreCount will return the number of entries in the query array which can then be accessed using the EpicGames_Leaderboards_CopyLeaderboardUserScoreByIndex function.

EpicGames_Leaderboards_QueryLeaderboardDefinitions

Query for a list of existing leaderboards definitions including their attributes.

Once the callback has been fired with a successful EpicGames Result, it is possible to call one of the following functions:

- EpicGames_Leaderboards_CopyLeaderboardDefinitionByIndex
- EpicGames_Leaderboards_CopyLeaderboardDefinitionByLeaderboardId
- EpicGames_Leaderboards_GetLeaderboardDefinitionCount

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Leaderboards_QueryLeaderboardDefinitions

Syntax:

```
EpicGames_Leaderboards_QueryLeaderboardDefinitions(userID, startTime, endTime)
```

Argument	Type	Description
userID	string	Product User ID for user who is querying definitions. Must be set when using a client policy that requires a valid logged in user. Not used for Dedicated Server where no user is available.
startTime	int64	An optional POSIX timestamp for the leaderboard's start time, or EpicGames_LEADERBOARDS_TIME_UNDEFINED
endTime	int64	An optional POSIX timestamp for the leaderboard's end time, or EpicGames_LEADERBOARDS_TIME_UNDEFINED

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"EpicGames_Leaderboards_QueryLeaderboardDefinitions"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID

Example:

```
identifier = EpicGames_Leaderboards_QueryLeaderboardDefinitions
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_Leaderboards_QueryLeaderboardDefinitions")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Leaderboards_QueryLeaderboardRanks

Retrieves top leaderboard records by rank in the leaderboard matching the given leaderboard ID.

Once the callback has been fired with a successful EpicGames Result, it is possible to call one of the following functions:

- EpicGames_Leaderboards_CopyLeaderboardRecordByIndex
- EpicGames_Leaderboards_CopyLeaderboardRecordByUserId
- EpicGames_Leaderboards_GetLeaderboardRecordCount

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Leaderboards_QueryLeaderboardRanks

Syntax:

```
EpicGames_Leaderboards_QueryLeaderboardRanks(userID, LeaderboardId)
```

Argument	Type	Description
userID	string	The ID of the leaderboard whose information you want to retrieve
LeaderboardId	string	Product User ID for user who is querying ranks. Must be set when using a client policy that requires a valid logged in user. Not used for Dedicated Server where no user is available

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"EpicGames_Leaderboards_QueryLeaderboardRanks"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID

Example:

```
identifier = EpicGames_Leaderboards_QueryLeaderboardRanks(userID, LeaderboardId);
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_Leaderboards_QueryLeaderboardRanks")
if (async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Leaderboards_QueryLeaderboardUserScores

Query for a list of scores for a given list of users.

Once the callback has been fired with a successful EpicGames Result, it is possible to call one of the following functions:

- EpicGames_Leaderboards_CopyLeaderboardUserScoreByIndex
- EpicGames_Leaderboards_CopyLeaderboardUserScoreByUserId
- EpicGames_Leaderboards_GetLeaderboardUserScoreCount

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Leaderboards_QueryLeaderboardUserScores

Syntax:

```
EpicGames_Leaderboards_QueryLeaderboardUserScores(userID, LeaderboardId, name, aggregation, startTime, endTime)
```

Argument	Type	Description
userID	string	The argument to be passed in
LeaderboardId	string	Product User ID indicating the users whose scores you want to retrieve
name	string	The name of the stat to query.
aggregation	real	Aggregation used to sort the cached user scores.
startTime	int64	An optional POSIX timestamp, or EpicGames_LEADERBOARDS_TIME_UNDEFINED; results will only include scores made after this time
endTime	int64	An optional POSIX timestamp, or EpicGames_LEADERBOARDS_TIME_UNDEFINED; results will only include scores made before this time

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"EpicGames_Leaderboards_QueryLeaderboardUserScores"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID

Example:

```
identifier = EpicGames_Leaderboards_QueryLeaderboardUserScores(userID, LeaderboardId,
name, aggregation, startTime, endTime)
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_Leaderboards_QueryLeaderboardUserScores")
if (async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

LeaderboardUserScore

An leaderboard definition is represented by a struct and contains information about a single leaderboard user score. This struct is returned by the following functions:

- EpicGames_Leaderboards_CopyLeaderboardUserScoreByIndex
- EpicGames_Leaderboards_CopyLeaderboardUserScoreByUserId

Key	Type	Description
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
UserId	string	The Product User ID of the user who got this score
Score	real	Leaderboard score

LeaderboardRecord

An leaderboard definition is represented by a struct and contains information about a single leaderboard record. This struct is returned by the following functions:

- EpicGames_Leaderboards_CopyLeaderboardRecordByIndex
- EpicGames_Leaderboards_CopyLeaderboardRecordByUserId

Key	Type	Description
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
UserId	string	The Product User ID associated with this record
Rank	real	Sorted position on leaderboard
Score	real	Leaderboard score
UserDisplayName	string	The latest display name seen for the user since they last time logged in. This is empty if the user does not have a display name set

LeaderboardDefinition

An leaderboard definition is represented by a struct and contains information about a single leaderboard definition. This struct is returned by the following functions:

- EpicGames_Leaderboards_CopyLeaderboardDefinitionByIndex
- EpicGames_Leaderboards_CopyLeaderboardDefinitionByLeaderboardId

Key	Type	Description
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
LeaderboardId	string	Unique ID to identify leaderboard.
StatName	string	Name of stat used to rank leaderboard.
StartTime	real	The POSIX timestamp for the start time, or EpicGames_LEADERBOARDS_TIME_UNDEFINED.
EndTime	real	The POSIX timestamp for the end time, or EpicGames_LEADERBOARDS_TIME_UNDEFINED.
Aggregation	real	Used to sort leaderboard. EpicGames_LA_Min, EpicGames_LA_Max, EpicGam or EpicGames_LA_Latest

Metrics

The **Metrics Interface** tracks your game's usage and populates the **Game Analytics dashboard** in the **Developer Portal**. This data includes active, online instances of the game's client and server, and past sessions played by local players.

Functions

These functions are provided for handling metrics:

- **EpicGames_Metrics_BeginPlayerSession**
- **EpicGames_Metrics_EndPlayerSession**

Constants

These are the constants used by this API:

- **EpicGames AccountId Type**
- **EpicGames Controller Type**

EpicGames_Metrics_BeginPlayerSession

Logs the start of a new game session for a local player. The game client should call this function whenever it joins into a new multiplayer, peer-to-peer or single player game session. Each call to EpicGames_Metrics_BeginPlayerSession must be matched with a corresponding call to EpicGames_Metrics_EndPlayerSession.

EXTERNAL A wrapper around EOS_Metrics_BeginPlayerSession

Syntax:

```
EpicGames_Metrics_BeginPlayerSession(accountID, DisplayName, AccountIdType, ControllerType, Se
```

Argument	Type	Description
accountID	string	An Epic Account ID. Set this field when AccountIdType is set to `EpicGames_MAIT_Epic`.
DisplayName	string	The in-game display name for the user.
AccountIdType	EpicGames AccountId Type	Account ID type that is set in the union.
ControllerType	EpicGames Controller Type	The user's game controller type.
ServerIp	string	IP address of the game server hosting the game session. For a localhost session, set to `undefined`. If both IPv4 and IPv6 addresses are available, use the **IPv6** address.
GameSessionId	string	Optional, application-defined custom match session identifier. If the identifier is not used, set to `undefined` which will be shown in the Played Sessions listing at the user profile dashboard.

Returns:

```
struct
```

key	type	Description
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code

Example:

```
var struct = EpicGames_Metrics_BeginPlayerSession(
        AccountID,
        "YYEpicGames",
        EpicGames_MAIT_Epic,
        EpicGames_UCT_MouseKeyboard,
        "No Server",
        "AnyRandomIDString")

if(struct.status != EpicGames_Success)
{
    show_debug_message("EpicGames_Metrics_BeginPlayerSession Failed")
}
```

The above code will show an example of how the function should be used.

EpicGames_Metrics_EndPlayerSession

Logs the end of a game session for a local player. Call once when the game client leaves the active game session. Each call to EpicGames_Metrics_BeginPlayerSession must be matched with a corresponding call to EpicGames_Metrics_EndPlayerSession.

EXTERNAL A wrapper around EOS_Metrics_EndPlayerSession

Syntax:

```
EpicGames_Metrics_EndPlayerSession(accountID,AccountIdType)
```

Argument	Type	Description
accountID	string	Set this field when AccountIdType is set to `EpicGames_MAIT_Epic`
AccountIdType	real	The Account ID type that is set in the union

Returns:

```
struct
```

key	type	Description
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code

Example:

```
var struct = EpicGames_Metrics_EndPlayerSession(AccountID,EpicGames_MAIT_Epic)
show_debug_message("EpicGames_Metrics_EndPlayerSession: " +
string(struct.status_message))
```

The above code will show an example of how the function should be used.

EpicGames AccountId Type

These constants are used to describe the AccountId type are used by the following functions:

- EpicGames_Metrics_BeginPlayerSession
- EpicGames_Metrics_EndPlayerSession

EpicGames AccountId Type Constant	Description
`EpicGames_MAIT_Epic`	An Epic Account ID
`EpicGames_MAIT_External`	An external service Account ID

EpicGames Controller Type

These constants are used to describe the type of controller being used during the current player session, and are used by:

- EpicGames_Metrics_BeginPlayerSession

EpicGames AccountId Type Constant	Description
`EpicGames_UCT_Unknown`	The game controller type is unknown
`EpicGames_UCT_MouseKeyboard`	Mouse and keyboard controller
`EpicGames_UCT_GamepadControl`	Gamepad controller
`EpicGames_UCT_UCT_TouchControl`	Touch controller

Platform

The Platform Interface sits at the heart of the **Epic Online Services** (EOS) SDK and holds the handles you need to access every other interface and keep them all running. When your application starts up, you can initialize the SDK and get a handle to the Platform Interface. This handle is usable for the lifetime of the SDK.

Functions

These functions are provided for handling platform functionality:

- EpicGames_Platform_CheckForLauncherAndRestart
- EpicGames_Platform_GetActiveCountryCode
- EpicGames_Platform_GetActiveLocaleCode
- EpicGames_Platform_GetOverrideCountryCode
- EpicGames_Platform_GetOverrideLocaleCode
- EpicGames_Platform_Release
- EpicGames_Platform_SetOverrideCountryCode
- EpicGames_Platform_SetOverrideLocaleCode
- EpicGames_Platform_Tick

EpicGames_Platform_CheckForLauncherAndRestart

Checks if the app was launched through the Epic Launcher, and relaunches it through the Epic Launcher if it wasn't.

Returns one of 3 possible results:

1. `EpicGames_Success` is returned if the app is being restarted. You should quit your process as soon as possible.
2. `EpicGames_NoChange` is returned if the app was already launched through the Epic Launcher, and no action needs to be taken.
3. `EpicGames_UnexpectedError` is returned if the **LauncherCheck** module failed to initialize, or the module tried and failed to restart the app.

EXTERNAL A wrapper around EOS_Platform_CheckForLauncherAndRestart

Syntax:

```
EpicGames_Platform_CheckForLauncherAndRestart()
```

Returns:

```
real
```

Example:

```
if (EpicGames_Platform_CheckForLauncherAndRestart() != EpicGames_NoChange)
game_end();
```

The above code will show an example of how the function should be used. If the output of the function is other than `EpicGames_NoChange` it will force close the project.

EpicGames_Platform_GetActiveCountryCode

Returns the active country code. This only will return the value set as the override otherwise empty string is returned.

NOTE This is NOT currently used for anything internally.

EXTERNAL A wrapper around EOS_Platform_GetActiveCountryCode

Syntax:

```
EpicGames_Platform_GetActiveCountryCode(accountID)
```

Argument	Type	Description
accountID	string	The account to get the active country code of

Returns:

```
string
```

Example:

```
show_debug_message("CountryCode: "
+ EpicGames_Platform_GetActiveCountryCode(accountID))
```

The above code will show an example of how the function should be used.

EpicGames_Platform_GetActiveLocaleCode

Get the active locale code that the SDK will send to services which require it. This returns the override value otherwise it will use the locale code of the given user. This is used for localization. This follows ISO 639.

EXTERNAL A wrapper around EOS_Platform_GetActiveLocaleCode

Syntax:

```
EpicGames_Platform_GetActiveLocaleCode(accountID)
```

Argument	Type	Description
accountID	string	The account to get the local code of

Returns:

```
string
```

Example:

```
show_debug_message("LocaleCode: "
+ EpicGames_Platform_GetActiveLocaleCode(accountID))
```

The above code will show an example of how the function should be used.

EpicGames_Platform_GetOverrideCountryCode

Get the override country code that the SDK will send to services which require it. This is not currently used for anything internally.

NOTE This is NOT currently used for anything internally.

EXTERNAL A wrapper around EOS_Platform_GetOverrideCountryCode

Syntax:

```
EpicGames_Platform_GetOverrideCountryCode()
```

Returns:

```
string
```

Example:

```
show_debug_message("CountryCode: " + EpicGames_Platform_GetOverrideCountryCode())
```

The above code will show an example of how the function should be used.

EpicGames_Platform_GetOverrideLocaleCode

Get the override locale code that the SDK will send to services which require it. This is used for localization. This follows ISO 639.

EXTERNAL A wrapper around EOS_Platform_GetOverrideLocaleCode

Syntax:

```
EpicGames_Platform_GetOverrideLocaleCode()
```

Returns:

```
string
```

Example:

```
show_debug_message("LocaleCode: " + EpicGames_Platform_GetOverrideCountryCode())
```

The above code will show an example of how the function should be used.

EpicGames_Platform_Release

Release an Epic Online Services platform. This function should only be called when terminating your application right before calling EpicGames_Shutdown.

Undefined behavior will result in calling it more than once.

EXTERNAL A wrapper around EOS_Platform_Release

Syntax:

```
EpicGames_Platform_Release()
```

Returns:

```
N/A
```

Example:

```
EpicGames_Platform_Release();
EpicGames_Shutdown();

game_end();
```

The above code will show an example of how the function should be used.

EpicGames_Platform_SetOverrideCountryCode

Set the override country code that the SDK will send to services which require it. This is not currently used for anything internally.

EXTERNAL A wrapper around EOS_Platform_SetOverrideCountryCode

Syntax:

```
EpicGames_Platform_SetOverrideCountryCode(countryCode)
```

Argument	Type	Description
countryCode	string	New country code ISO 639

Returns:

```
N/A
```

Example:

```
EpicGames_Platform_SetOverrideCountryCode("UK")
```

The above code will show an example of how the function should be used.

EpicGames_Platform_SetOverrideLocaleCode

Set the override locale code that the SDK will send to services which require it. This is used for localization. This follows ISO 639.

EXTERNAL A wrapper around EOS_Platform_SetOverrideLocaleCode

Syntax:

```
EpicGames_Platform_SetOverrideLocaleCode(localCode)
```

Argument	Type	Description
localCode	string	New local code

Returns:

```
N/A
```

Example:

```
EpicGames_Platform_SetOverrideLocaleCode("en")
```

The above code will show an example of how the function should be used.

EpicGames_Platform_Tick

Notify the platform instance to do work. This function must be called frequently in order for the services provided by the SDK to properly.

1. For tick-based applications, it is usually desirable to call this once per-tick.

EXTERNAL A wrapper around EOS_Platform_Tick

Syntax:

```
EpicGames_Platform_Tick()
```

Returns:

```
N/A
```

Example:

```
EpicGames_Platform_Tick()
```

The above code will show a code example.

Player Data Storage

The **Player Data Storage Interface** enables developers using **Epic Online Services** (EOS) to save encrypted, user-specific, game-specific data to cloud servers. Data that you store through this interface is accessible to the user on any device where they can log in. The Player Data Storage Interface supports any file format; typical use cases would include saved games and replay data.

Functions

These functions are provided for handling player data storage:

- **EpicGames_PlayerDataStorage_CopyFileMetadataAtIndex**
- **EpicGames_PlayerDataStorage_CopyFileMetadataByFilename**
- **EpicGames_PlayerDataStorage_DeleteCache**
- **EpicGames_PlayerDataStorage_DeleteFile**
- **EpicGames_PlayerDataStorage_DuplicateFile**
- **EpicGames_PlayerDataStorage_GetFileMetadataCount**
- **EpicGames_PlayerDataStorage_QueryFile**
- **EpicGames_PlayerDataStorage_QueryFileList**
- **EpicGames_PlayerDataStorage_ReadFile**
- **EpicGames_PlayerDataStorage_WriteFile**
- **EpicGames_PlayerDataStorageFileTransferRequest_CancelRequest**

Structures

These are the structures used by this API:

- **PlayerFileMetadata**

EpicGames_PlayerDataStorage_CopyFileMetadataAtIndex

Get the cached copy of a file's metadata by index. The metadata will be for the last retrieved or successfully saved version, and will not include any local changes that have not been committed by calling EpicGames_PlayerDataStorage_WriteFile.

NOTE Requires a previous call to EpicGames_PlayerDataStorage_QueryFileList to store values in cache.

EXTERNAL A wrapper around EOS_PlayerDataStorage_CopyFileMetadataAtIndex

Syntax:

```
EpicGames_PlayerDataStorage_CopyFileMetadataAtIndex(userID,index)
```

Argument	Type	Description
userID	string	The Product User ID of the local user who is requesting file metadata
index	real	The index to get metadata for

Returns:

```
struct (PlayerFileMetadata)
```

Example:

```
var count = EpicGames_PlayerDataStorage_GetFileMetadataCount(userID)
for(var i = 0 ; i < count ; i ++)
{
    var struct = EpicGames_PlayerDataStorage_CopyFileMetadataAtIndex(userID,i);
    Filename = struct.Filename;
}
```

The above code will show an example of how the function should be used. The player file metadata is returned providing an index.

EpicGames_PlayerDataStorage_CopyFileMetadataByFilename

Create the cached copy of a file's metadata by filename. The metadata will be for the last retrieved or successfully saved version, and will not include any changes that have not completed writing.

NOTE Requires a previous call to EpicGames_PlayerDataStorage_QueryFileList to store values in cache.

EXTERNAL A wrapper around EOS_PlayerDataStorage_CopyFileMetadataByFilename

Syntax:

```
EpicGames_PlayerDataStorage_CopyFileMetadataByFilename(userID, filename)
```

Argument	Type	Description
userID	string	The Product User ID of the local user who is requesting file metadata
filename	string	The file's name to get metadata for

Returns:

```
struct (PlayerFileMetadata)
```

Example:

```
var struct = EpicGames_PlayerDataStorage_CopyFileMetadataByFilename(userID, filename)
if(struct.status == EpicGames_Success)
{
    var Filename = struct.Filename
}
```

The above code will show an example of how the function should be used. The player file metadata is returned for a provided filename.

EpicGames_PlayerDataStorage_DeleteCache

Clear previously cached file data. This operation will be done asynchronously. All cached files except those corresponding to the transfers in progress will be removed. Warning: Use this with care. Cache system generally tries to clear old and unused cached files from time to time. Unnecessarily clearing cache can degrade performance as SDK will have to re-download data.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_PlayerDataStorage_DeleteCache

Syntax:

```
EpicGames_PlayerDataStorage_DeleteCache(userID)
```

Argument	Type	Description
userID	string	Product User ID of the local user who is deleting his cache

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"EpicGames_PlayerDataStorage_DeleteCache"`

status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_PlayerDataStorage_DeleteCache(userID)
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_PlayerDataStorage_DeleteCache")
if (async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_PlayerDataStorage_DeleteFile

Deletes an existing file in the cloud. If successful, the file's data will be removed from our local cache.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_PlayerDataStorage_DeleteFile

Syntax:

```
EpicGames_PlayerDataStorage_DeleteFile(userID,filename)
```

Argument	Type	Description
userID	string	The Product User ID of the local user who authorizes deletion of the file; must be the file's owner
filename	string	The name of the file to delete

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"EpicGames_PlayerDataStorage_DeleteFile"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the

		operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_PlayerDataStorage_DeleteFile(userID,"MyFilename.txt")
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_PlayerDataStorage_DeleteFile")
if (async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_PlayerDataStorage_DuplicateFile

Copies the data of an existing file to a new filename. This action happens entirely on the server and will not upload the contents of the source destination file from the host. This function paired with a subsequent EpicGames_PlayerDataStorage_DeleteFile can be used to rename a file. If successful, the destination file's metadata will be updated in our local cache.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_PlayerDataStorage_DuplicateFile

Syntax:

```
EpicGames_PlayerDataStorage_DuplicateFile(userID,source,destination)
```

Argument	Type	Description
userID	string	The Product User ID of the local user who authorized the duplication of the requested file; must be the original file's owner
source	string	The name of the existing file to duplicate
destination	string	The name of the new file

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents

Key	Type	Description
type	string	`"EpicGames_PlayerDataStorage_DuplicateFile"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier =
EpicGames_PlayerDataStorage_DuplicateFile(userID,"myNiceFile.dat","myNiceFile2.dat")
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_PlayerDataStorage_DuplicateFile")
if (async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_PlayerDataStorage_GetFileMetadataCount

Get the count of files we have previously queried information for and files we have previously read from / written to.

NOTE Requires a previous call to EpicGames_PlayerDataStorage_QueryFileList to store values in cache.

EXTERNAL A wrapper around EOS_PlayerDataStorage_GetFileMetadataCount

Syntax:

```
EpicGames_PlayerDataStorage_GetFileMetadataCount(userID)
```

Argument	Type	Description
userID	string	The Product User ID of the local user who is requesting file metadata

Returns:

```
real
```

Example:

```
var count = EpicGames_PlayerDataStorage_GetFileMetadataCount(userID)
for (var i = 0 ; i < count ; i ++)
{
    var struct = EpicGames_PlayerDataStorage_CopyFileMetadataAtIndex(userID,i);
    Filename = struct.Filename;
}
```

The above code will show an example of how the function should be used. After a successcull call to EpicGames_PlayerDataStorage_QueryFileList, the function EpicGames_PlayerDataStorage_GetFileMetadataCount will return the number of

entries in the query array which can then be accessed using the EpicGames_PlayerDataStorage_CopyFileMetadataAtIndex function.

EpicGames_PlayerDataStorage_QueryFile

Query a specific file's metadata, such as file names, size, and a MD5 hash of the data. This is not required before a file may be opened, saved, copied, or deleted. Once a file has been queried, its metadata will be available by the one of the following functions:

- EpicGames_PlayerDataStorage_CopyFileMetadataAtIndex
- EpicGames_PlayerDataStorage_CopyFileMetadataByFilename

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_PlayerDataStorage_QueryFile

Syntax:

```
EpicGames_PlayerDataStorage_QueryFile(userID,filename)
```

Argument	Type	Description
userID	string	The Product User ID of the local user requesting file metadata
filename	string	The name of the file being queried

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	Type	Description

type	string	`"EpicGames_PlayerDataStorage_QueryFile"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_PlayerDataStorage_QueryFile(userID,"myFile.txt")
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_PlayerDataStorage_QueryFile")
if (async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_PlayerDataStorage_QueryFileList

Query the file metadata, such as file names, size, and a MD5 hash of the data, for all files owned by this user for this application. This is not required before a file may be opened, saved, copied, or deleted.

Once the callback has been fired with a successful EpicGames Result, it is possible to call one of the following functions:

- EpicGames_PlayerDataStorage_CopyFileMetadataAtIndex
- EpicGames_PlayerDataStorage_CopyFileMetadataByFilename
- EpicGames_PlayerDataStorage_GetFileMetadataCount

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_PlayerDataStorage_QueryFileList

Syntax:

```
EpicGames_PlayerDataStorage_QueryFileList(userID)
```

Argument	Description
userID	The Product User ID of the local user who requested file metadata

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	Type	Description
type	string	`"EpicGames_PlayerDataStorage_QueryFileList"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_PlayerDataStorage_QueryFileList(userID)
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_PlayerDataStorage_QueryFileList")
if (async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_PlayerDataStorage_ReadFile

Retrieve the contents of a specific file, potentially downloading the contents if we do not have a local copy, from the cloud. This request will occur asynchronously, potentially over multiple frames. All callbacks for this function will come from the same thread that the SDK is ticked from. If specified, the FileTransferProgressCallback will always be called at least once if the request is started successfully.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_PlayerDataStorage_ReadFile

Syntax:

```
EpicGames_PlayerDataStorage_ReadFile(userID, file, path)
```

Argument	Description
userID	The Product User ID of the local user who is reading the requested file
filen	The file name to read; this file must already exist
path	local path where save the file

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	Type	Description

type	string	`"EpicGames_PlayerDataStorage_ReadFile"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_PlayerDataStorage_ReadFile(userID,"MyFavPic.png","path/to/
save/MyFavPic.png")
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_PlayerDataStorage_ReadFile")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_PlayerDataStorage_WriteFile

Write new data to a specific file, potentially overwriting any existing file by the same name, to the cloud. This request will occur asynchronously, potentially over multiple frames. All callbacks for this function will come from the same thread that the SDK is ticked from. If specified, the FileTransferProgressCallback will always be called at least once if the request is started successfully.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_PlayerDataStorage_WriteFile

Syntax:

```
EpicGames_PlayerDataStorage_WriteFile(userID,file,path)
```

Argument	Description
userID	The Product User ID of the local user who is writing the requested file to the cloud
file	The name of the file to write; if this file already exists, the contents will be replaced if the write request completes successfully
path	Local path of the file to upload

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"EpicGames_PlayerDataStorage_WriteFile"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_PlayerDataStorage_WriteFile(userID,"myData.dat",/path/to/file/
myData.dat)
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_PlayerDataStorage_WriteFile")
if (async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_PlayerDataStorageFileTransferRequest_CancelRequest

Attempt to cancel this file request in progress. This is a best-effort command and is not guaranteed to be successful if the request has completed before this function is called.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_PlayerDataStorageFileTransferRequest_CancelRequest

Syntax:

```
EpicGames_PlayerDataStorageFileTransferRequest_CancelRequest(filename)
```

Argument	Type	Description
filename	string	Filename contained in the process to cancel

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"EpicGames_PlayerDataStorageFileTransferRequest_CancelRequest"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code

identifier	real	The asynchronous listener ID

Example:

```
identifier =
EpicGames_PlayerDataStorageFileTransferRequest_CancelRequest("myFile.txt")
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] ==
"EpicGames_PlayerDataStorageFileTransferRequest_CancelRequest")
if (async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

PlayerFileMetadata

The player file metadata is represented by a struct and contains metadata information for a specific player file. This struct is returned by the following functions:

- EpicGames_PlayerDataStorage_CopyFileMetadataAtIndex
- EpicGames_PlayerDataStorage_CopyFileMetadataByFilename

Key	Type	Description
FileSizeBytes	real	The total size of the file in bytes (Includes file header in addition to file contents)
MD5Hash	string	The MD5 Hash of the entire file (including additional file header), in hex digits
Filename	string	The file's name
LastModifiedTime	real	The POSIX timestamp when the file was saved last time.
UnencryptedDataSizeBytes	real	The size of data (payload) in file in unencrypted (original) form.

Progression Snapshot

The **Progression Snapshot Interface** allows storing player-specific game data for the purposes of supporting an end-user experience for Epic account merging. As of today, the concept of merging two separate Epic accounts, owned by the same user, into a single Epic account does not exist. However, this type of account merging is a feature that will be made available to users of Epic Accounts in the future.

The progression snapshot feature becomes relevant for users in cases where they have two separate Epic accounts, and have played the same game on both of the accounts. In such a case, if the user chooses to merge their Epic accounts into a single account, the Epic overlay will be able to present a snapshot view of their game progress for both accounts. This allows users to choose their preferred game progression to preserve as a result of the account merge operation.

Functions

These functions are provided for handling progression snapshot:

- **EpicGames_ProgressionSnapshot_AddProgression**
- **EpicGames_ProgressionSnapshot_BeginSnapshot**
- **EpicGames_ProgressionSnapshot_DeleteSnapshot**
- **EpicGames_ProgressionSnapshot_EndSnapshot**
- **EpicGames_ProgressionSnapshot_SubmitSnapshot**

EpicGames_ProgressionSnapshot_AddProgression

Stores a Key/Value pair in memory for a given snapshot. If multiple calls happen with the same key, the last invocation wins, overwriting the previous value for that given key. The order in which the Key/Value pairs are added is stored as is for later retrieval/display. Ideally, you would make multiple calls to EpicGames_ProgressionSnapshot_AddProgression followed by a single call to EpicGames_ProgressionSnapshot_SubmitSnapshot.

EXTERNAL A wrapper around EOS_ProgressionSnapshot_AddProgression

Syntax:

```
EpicGames_ProgressionSnapshot_AddProgression(snapshotId, key, value);
```

Argument	Type	Description
snapshotId	real	The Snapshot Id received via a EpicGames_ProgressionSnapshot_BeginSnapshot function.
key	string	The key in a key/value pair of progression entry
value	string	The value in a key/value pair of progression entry

Returns:

```
N/A
```

Example:

```
identifier = EpicGames_ProgressionSnapshot_BeginSnapshot(local_UserId)
EpicGames_ProgressionSnapshot_AddProgression(identifier, "PlayerName", "Hero");
```

The code sample above shows an example of how to create a snapshot (EpicGames_ProgressionSnapshot_BeginSnapshot) and add a progression value to it.

EpicGames_ProgressionSnapshot_BeginSnapshot

Creates a new progression-snapshot resource for a given user. This function will return a progression-snapshot identifier output parameter. Use that identifier to reference the snapshot in the other functions:

- EpicGames_ProgressionSnapshot_AddProgression
- EpicGames_ProgressionSnapshot_DeleteSnapshot
- EpicGames_ProgressionSnapshot_EndSnapshot
- EpicGames_ProgressionSnapshot_SubmitSnapshot

EXTERNAL A wrapper around EOS_ProgressionSnapshot_BeginSnapshot

Syntax:

```
EpicGames_ProgressionSnapshot_BeginSnapshot(local_UserId);
```

Argument	Type	Description
local_UserId	string	The Product User ID of the local user to whom the key/value pair belong

Returns:

```
real
```

Example:

```
identifier = EpicGames_ProgressionSnapshot_BeginSnapshot(local_UserId)
EpicGames_ProgressionSnapshot_AddProgression(identifier, "PlayerName", "Hero");
```

The code sample above shows how to create a snapshot and add a progression value to it (EpicGames_ProgressionSnapshot_AddProgression).

EpicGames_ProgressionSnapshot_DeleteSnapshot

Wipes out all progression data for the given user from the service. However, any previous progression data that haven't been submitted yet are retained.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_ProgressionSnapshot_DeleteSnapshot

Syntax:

```
EpicGames_ProgressionSnapshot_DeleteSnapshot(userId);
```

Argument	Type	Description
userId	string	The Product User ID of the local user to whom the key/value pair belong

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	real	`"EpicGames_ProgressionSnapshot_DeleteSnapshot"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors

status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID

Example:

```
identifier = EpicGames_ProgressionSnapshot_DeleteSnapshot(userId);
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_ProgressionSnapshot_DeleteSnapshot")
if (async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_ProgressionSnapshot_EndSnapshot

Cleans up and releases resources associated with the given progression snapshot identifier.

NOTE This function should be called after submission (EpicGames_ProgressionSnapshot_SubmitSnapshot).

EXTERNAL A wrapper around EOS_ProgressionSnapshot_EndSnapshot

Syntax:

```
EpicGames_ProgressionSnapshot_EndSnapshot(snapshotId);
```

Argument	Type	Description
snapshotId	string	The Snapshot Id received via a EpicGames_ProgressionSnapshot_BeginSnapshot function.

Returns:

```
struct
```

Key	Type	Description
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the *status* code

Example:

```
result = EpicGames_ProgressionSnapshot_EndSnapshot(snapshotId)
if (result.status == EpicGames_Success)
{
    show_debug_message("EpicGames_ProgressionSnapshot_EndSnapshot: success");
}
```

The code above matches the response status and logs the success of the task.

EpicGames_ProgressionSnapshot_SubmitSnapshot

Saves the previously added Key/Value pairs of a given Snapshot to the service.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

NOTE This will overwrite any prior progression data stored with the service that's associated with the user.

EXTERNAL A wrapper around EOS_ProgressionSnapshot_SubmitSnapshot

Syntax:

```
EpicGames_ProgressionSnapshot_SubmitSnapshot(snapshotId);
```

Argument	Type	Description
snapshotId	real	The Snapshot Id received via a EpicGames_ProgressionSnapshot_BeginSnapshot function.

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	`"EpicGames_ProgressionSnapshot_SubmitSnapshot"`

status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID

Example:

```
identifier = EpicGames_ProgressionSnapshot_SubmitSnapshot(snapshotId);
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_ProgressionSnapshot_SubmitSnapshot")
if (async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

Sanctions

The Sanctions Interface manages punitive actions taken against your users. Actions may include temporary or permanent bans from gameplay or communication bans that limit the social aspects of your product for a particular user. You define the disciplinary actions for your product to handle negative behavior based on your use cases.

Functions

These functions are provided for handling sanctions:

- EpicGames_Sanctions_CopyPlayerSanctionByIndex
- EpicGames_Sanctions_GetPlayerSanctionCount
- EpicGames_Sanctions_QueryActivePlayerSanctions

EpicGames_Sanctions_CopyPlayerSanctionByIndex

Copies an active player sanction.

NOTE Requires a previous call to EpicGames_Sanctions_QueryActivePlayerSanctions to store values in cache.

EXTERNAL A wrapper around EOS_Sanctions_CopyPlayerSanctionByIndex

Syntax:

```
EpicGames_Sanctions_CopyPlayerSanctionByIndex(UserID_target, index)
```

Argument	Type	Description
UserID_target	string	Product User ID of the user whose active sanctions are to be copied
index	real	Index of the sanction to retrieve from the cache

Returns:

```
struct
```

key	type	Description
Action	string	The action associated with this sanction
ReferenceId	string	A unique identifier for this specific sanction
TimeExpires	real	The POSIX timestamp when the sanction will expire. If the sanction is permanent, this will be 0
TimePlaced	real	The POSIX timestamp when the sanction was placed

Example:

```
var count = EpicGames_Sanctions_GetPlayerSanctionCount(UserID_target)
for(var i = 0 ; i < count ; i++)
{
    EpicGames_Sanctions_CopyPlayerSanctionByIndex(UserID_target,i)
    var Action = struct.Action
}
```

The above code will show an example of how the function should be used. The player sanction data is returned for the provided index.

EpicGames_Sanctions_GetPlayerSanctionCount

Fetch the number of player sanctions that have been retrieved for a given player.

NOTE Requires a previous call to EpicGames_Sanctions_QueryActivePlayerSanctions to store values in cache.

EXTERNAL A wrapper around EOS_Sanctions_GetPlayerSanctionCount

Syntax:

```
EpicGames_Sanctions_GetPlayerSanctionCount(UserID_target)
```

Argument	Description
UserID_target	Product User ID of the user whose sanction count should be returned

Returns:

```
real
```

Example:

```
var count = EpicGames_Sanctions_GetPlayerSanctionCount(UserID_target)
for(var i = 0 ; i < count ; i++)
{
    EpicGames_Sanctions_CopyPlayerSanctionByIndex(UserID_target,i)
    var Action = struct.Action
}
```

The above code will show an example of how the function should be used. After a successcull call to EpicGames_Sanctions_QueryActivePlayerSanctions, the function EpicGames_Sanctions_GetPlayerSanctionCount will return the number of entries in the query array which can then be accessed using the EpicGames_Sanctions_CopyPlayerSanctionByIndex function.

EpicGames_Sanctions_QueryActivePlayerSanctions

Start an asynchronous query to retrieve any active sanctions for a specified user.

Once the callback has been fired with a successful EpicGames Result, it is possible to call one of the following functions:

- EpicGames_Sanctions_CopyPlayerSanctionByIndex
- EpicGames_Sanctions_GetPlayerSanctionCount

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Sanctions_QueryActivePlayerSanctions

Syntax:

```
EpicGames_Sanctions_QueryActivePlayerSanctions(UserID,UserID_target)
```

Argument	Description
UserID	The Product User ID of the local user who initiated this request. Dedicated servers should set this to null.
UserID_target	Product User ID of the user whose active sanctions are to be retrieved.

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_Sanctions_QueryActivePlayerSanctions"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_Sanctions_QueryActivePlayerSanctions()
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_Sanctions_QueryActivePlayerSanctions")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

Stats

The Stats Interface provides the ability for developers to manage users' **stats** for an application, which can include any statistical data that a developer wishes to track, such as the number of items collected, the player's fastest completion time for a level, the total number of victories or losses, or the number of times that a user has performed a certain action. You can use stats to determine when to unlock Achievements and how to use rank users in Leaderboards.

Functions

These functions are provided for handling stats:

- EpicGames_Stats_CopyStatByIndex
- EpicGames_Stats_CopyStatByName
- EpicGames_Stats_GetStatsCount
- EpicGames_Stats_IngestStat
- EpicGames_Stats_QueryStats

Structures

These are the structures used by this API:

- StatData

EpicGames_Stats_CopyStatByIndex

Fetches a stat from a given index.

NOTE Requires a previous call to EpicGames_Stats_QueryStats to store values in cache.

EXTERNAL A wrapper around EOS_Stats_CopyStatByIndex

Syntax:

```
EpicGames_Stats_CopyStatByIndex(userID_target, index)
```

Argument	Type	Description
userID_target	string	The Product User ID of the user who owns the stat
index	real	Index of the stat to retrieve from the cache

Returns:

```
struct (StatData)
```

Example:

```
var count = EpicGames_Stats_GetStatsCount(userID_target)
for(var i = 0 ; i < count ; i ++)
{
    var struct = EpicGames_Stats_CopyStatByIndex(userID_target,i)
    var Name = struct.Name
}
```

The above code will show an example of how the function should be used. The stats data is returned for the provided stat index.

EpicGames_Stats_CopyStatByName

Fetches a stat from cached stats by name.

NOTE Requires a previous call to EpicGames_Stats_QueryStats to store values in cache.

EXTERNAL A wrapper around EOS_Stats_CopyStatByName

Syntax:

```
EpicGames_Stats_CopyStatByName(userID_target,name)
```

Argument	Type	Description
user_target	string	The Product User ID of the user who owns the stat
name	string	Name of the stat to retrieve from the cache

Returns:

```
struct (StatData)
```

Example:

```
var struct = EpicGames_Stats_CopyStatByName(userID_target,"MyStatName")
var Name = struct.Name
```

The above code will show an example of how the function should be used. The stats data is returned for the provided stat name.

EpicGames_Stats_GetStatsCount

Fetch the number of stats that are cached locally.

NOTE Requires a previous call to EpicGames_Stats_QueryStats to store values in cache.

EXTERNAL A wrapper around EOS_Stats_GetStatsCount

Syntax:

```
EpicGames_Stats_GetStatsCount(userID_target)
```

Argument	Type	Description
userID_target	string	The Product User ID for the user whose stats are being counted

Returns:

```
real
```

Example:

```
var count = EpicGames_Stats_GetStatsCount(userID_target)
for(var i = 0 ; i < count ; i ++)
{
    var struct = EpicGames_Stats_CopyStatByIndex(userID_target,i)
    var Name = struct.Name
}
```

The above code will show an example of how the function should be used. After a successcull call to EpicGames_Stats_QueryStats, the function EpicGames_Stats_GetStatsCount will return the number of entries in the query array which can then be accessed using the EpicGames_Stats_CopyStatByIndex function.

EpicGames_Stats_IngestStat

Ingest a stat by the amount specified in Options. When the operation is complete and the delegate is triggered the stat will be uploaded to the backend to be processed. The stat may not be updated immediately and an achievement using the stat may take a while to be unlocked once the stat has been uploaded.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Stats_IngestStat

Syntax:

```
EpicGames_Stats_IngestStat(userID, userID_target, statName, amount)
```

Argument	Type	Description
userID	string	The Product User ID of the local user requesting the ingest. Set to `undefiend` for dedicated server
userID_target	string	The Product User ID for the user whose stat is being ingested
statName	string	Name of the Stat to ingest
amount	real	Amount of the Stat to ingest

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_Stats_IngestStat"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_Stats_IngestStat(userID,userID,"Leaderboard_Stat",183)
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_Stats_IngestStat")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_Stats_QueryStats

Query for a list of stats for a specific player.

Once the callback has been fired with a successful EpicGames Result, it is possible to call one of the following functions:

- EpicGames_Stats_CopyStatByIndex
- EpicGames_Stats_CopyStatByName
- EpicGames_Stats_GetStatsCount

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Stats_QueryStats

Syntax:

```
EpicGames_Stats_QueryStats(userID, userID_target, startTime, endTime)
```

Argument	Description
userID	The Product User ID of the local user requesting the stats. Set to `undefined` for dedicated server
userID_target	The Product User ID for the user whose stats are being retrieved
startTime	The POSIX timestamp for start time OPTIONAL
endTime	The POSIX timestamp for end time OPTIONAL

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_Stats_QueryStats"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_Stats_QueryStats()
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_Stats_QueryStats")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

StatData

The stat data is represented by a struct and contains information for a specific stat. This struct is returned by the following functions:

- EpicGames_Stats_CopyStatByIndex
- EpicGames_Stats_CopyStatByName

Key	Type	Description
Name	string	Name of the stat.
StartTime	real	If not EpicGames_STATS_TIME_UNDEFINED then this is the POSIX timestamp for start time
EndTime	real	If not EpicGames_STATS_TIME_UNDEFINED then this is the POSIX timestamp for end time
Value	real	Current value for the stat

Title Storage

The **Title Storage Interface** enables developers using **Epic Online Services** (EOS) to retrieve encrypted data from cloud servers. Data that you store through this interface is accessible to any user on any device where they can log in. While similar to the **Player Data Storage**, this interface is specialized to handle game-specific data rather than user-specific data, and can provide different versions of files based on the user's platform, region, or other conditions.

Functions

These functions are provided for handling title storage:

- **EpicGames_TitleStorage_CopyFileMetadataAtIndex**
- **EpicGames_TitleStorage_CopyFileMetadataByFilename**
- **EpicGames_TitleStorage_DeleteCache**
- **EpicGames_TitleStorage_GetFileMetadataCount**
- **EpicGames_TitleStorage_QueryFile**
- **EpicGames_TitleStorage_QueryFileList**
- **EpicGames_TitleStorage_ReadFile**
- **EpicGames_TitleStorageFileTransferRequest_CancelRequest**

Structures

These are the structures used by this API:

- **TitleFileMetadata**

EpicGames_TitleStorage_CopyFileMetadataAtIndex

Get the cached copy of a file's metadata by index. The metadata will be for the last retrieved version.

NOTE Requires a previous call to EpicGames_TitleStorage_QueryFileList to store values in cache.

EXTERNAL A wrapper around EOS_TitleStorage_CopyFileMetadataAtIndex

Syntax:

```
EpicGames_TitleStorage_CopyFileMetadataAtIndex(userID,index)
```

Argument	Description
userID	Product User ID of the local user who is requesting file metadata (optional)
index	The index to get data for

Returns:

```
struct (TitleFileMetadata)
```

Example:

```
var count = EpicGames_TitleStorage_GetFileMetadataCount(userID)
for(var i = 0 ; i < count ; i ++)
{
    var struct = EpicGames_TitleStorage_CopyFileMetadataAtIndex(userID,i);
    Filename = struct.Filename;
}
```

The above code will show an example of how the function should be used. The title file metadata is returned for the provided file index.

EpicGames_TitleStorage_CopyFileMetadataByFilename

Create a cached copy of a file's metadata by filename. The metadata will be for the last retrieved or successfully saved version, and will not include any changes that have not completed writing.

NOTE Requires a previous call to EpicGames_TitleStorage_QueryFileList to store values in cache.

EXTERNAL A wrapper around EOS_TitleStorage_CopyFileMetadataByFilename

Syntax:

```
EpicGames_TitleStorage_CopyFileMetadataByFilename(userID,name)
```

Argument	Description
userID	Product User ID of the local user who is requesting file metadata (optional)
name	The file's name to get data for

Returns:

```
struct (TitleFileMetadata)
```

Example:

```
var struct = EpicGames_TitleStorage_CopyFileMetadataByFilename(userID,i);
if(struct.status == EpicGames_Success)
{
    Filename = struct.Filename;
}
```

The above code will show an example of how the function should be used. The title file metadata is returned for the provided file name.

EpicGames_TitleStorage_DeleteCache

Clear previously cached file data. This operation will be done asynchronously. All cached files except those corresponding to the transfers in progress will be removed. Warning: Use this with care. Cache system generally tries to clear old and unused cached files from time to time. Unnecessarily clearing cache can degrade performance as SDK will have to re-download data.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_TitleStorage_DeleteCache

Syntax:

```
EpicGames_TitleStorage_DeleteCache(userID)
```

Argument	Description
userID	Product User ID of the local user who is deleting his cache (optional)

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_TitleStorage_DeleteCache"`

status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_TitleStorage_DeleteCache(userID)
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_TitleStorage_DeleteCache")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_TitleStorage_GetFileMetadataCount

Get the count of files we have previously queried information for and files we have previously read from / written to.

NOTE Requires a previous call to EpicGames_TitleStorage_QueryFileList to store values in cache.

EXTERNAL A wrapper around EOS_TitleStorage_GetFileMetadataCount

Syntax:

```
EpicGames_TitleStorage_GetFileMetadataCount(userID)
```

Argument	Description
userID	Object containing properties related to which user is requesting the metadata count

Returns:

```
real
```

Example:

```
var count = EpicGames_TitleStorage_GetFileMetadataCount(userID)
for(var i = 0 ; i < count ; i ++)
{
    var struct = EpicGames_TitleStorage_CopyFileMetadataAtIndex(userID,i);
    Filename = struct.Filename;
}
```

The above code will show an example of how the function should be used. After a successcull call toEpicGames_TitleStorage_QueryFileList, the function EpicGames_TitleStorage_GetFileMetadataCount will return the number of entries

in the query array which can then be accessed using the EpicGames_TitleStorage_CopyFileMetadataAtIndex function.

EpicGames_TitleStorage_QueryFile

Query a specific file's metadata, such as file names, size, and a MD5 hash of the data. This is not required before a file may be opened. Once a file has been queried, its metadata will be available by the EpicGames_TitleStorage_CopyFileMetadataAtIndex and EpicGames_TitleStorage_CopyFileMetadataByFilename functions.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_TitleStorage_QueryFile

Syntax:

```
EpicGames_TitleStorage_QueryFile(userID,file)
```

Argument	Description
userID	Product User ID of the local user requesting file metadata (optional)
file	The requested file's name

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_TitleStorage_QueryFile"`

status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_TitleStorage_QueryFile(userID,"myFile.dat")
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_TitleStorage_QueryFile")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_TitleStorage_QueryFileList

Query the file metadata, such as file names, size, and a MD5 hash of the data, for all files available for current user based on their settings (such as game role) and tags provided. This is not required before a file can be downloaded by name.

Once the callback has been fired with a successful EpicGames Result, it is possible to call one of the following functions:

- EpicGames_TitleStorage_CopyFileMetadataAtIndex
- EpicGames_TitleStorage_CopyFileMetadataByFilename
- EpicGames_TitleStorage_GetFileMetadataCount

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_TitleStorage_QueryFileList

Syntax:

```
EpicGames_TitleStorage_QueryFileList(userID, tag)
```

Argument	Description
userID	Product User ID of the local user who requested file metadata
tag	List of tags to use for lookup.

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_TitleStorage_QueryFileList"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_TitleStorage_QueryFileList(userID,"Tag1")
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_TitleStorage_QueryFileList")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_TitleStorage_ReadFile

Retrieve the contents of a specific file, potentially downloading the contents if we do not have a local copy, from the cloud. This request will occur asynchronously, potentially over multiple frames.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_TitleStorage_ReadFile

Syntax:

```
EpicGames_TitleStorage_ReadFile(userID,file,path)
```

Argument	Description
userID	Product User ID of the local user who is reading the requested file
file	The file name to read; this file must already exist
path	Local path where save the file

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_TitleStorage_ReadFile"`

status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_TitleStorage_ReadFile(userID,"Preferences.json","/path/to/
save/Preferences.json")
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_TitleStorage_ReadFile")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_TitleStorageFileTransferRequest_CancelRequest

Attempt to cancel this file request in progress. This is a best-effort command and is not guaranteed to be successful if the request has completed before this function is called.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_TitleStorageFileTransferRequest_CancelRequest

Syntax:

```
EpicGames_TitleStorageFileTransferRequest_CancelRequest(filename)
```

Argument	Description
filename	Filename contained in the process to cancel

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_TitleStorageFileTransferRequest_CancelRequest"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the *status* code

identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_TitleStorageFileTransferRequest_CancelRequest("myFile.txt")
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] ==
"EpicGames_TitleStorageFileTransferRequest_CancelRequest")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

TitleFileMetadata

The player file metadata is represented by a struct and contains metadata information for a specific title file. This struct is returned by the following functions:

- EpicGames_TitleStorage_CopyFileMetadataAtIndex
- EpicGames_TitleStorage_CopyFileMetadataByFilename

Key	Type	Description
FileSizeBytes	double	The total size of the file in bytes (Includes file header in addition to file contents)
MD5Hash	string	The MD5 Hash of the entire file (including additional file header), in hex digits
Filename	string	The file's name
UnencryptedDataSizeBytes	double	The size of data (payload) in file in un-encrypted (original) form.

User Interface

Epic Online Services (EOS) supports product-independent overlay windows through its overlay system; these overlays give users product-independent access to various features. The **UI Interface** manages interactions with the overlays by providing status updates, showing or hiding the overlays, and adjusting display and hotkey preferences.

Functions

These functions are provided for handling title storage:

- EpicGames_UI_AddNotifyDisplaySettingsUpdated
- EpicGames_UI_GetFriendsVisible
- EpicGames_UI_GetNotificationLocationPreference
- EpicGames_UI_HideFriends
- EpicGames_UI_RemoveNotifyDisplaySettingsUpdated
- EpicGames_UI_SetDisplayPreference
- EpicGames_UI_ShowFriends

Constants

These are the constants used by this API:

- UINotificationLocation

EpicGames_UI_AddNotifyDisplaySettingsUpdated

Register to receive notifications when the overlay display settings are updated. Newly registered handlers will always be called the next tick with the current state.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_UI_AddNotifyDisplaySettingsUpdated

Syntax:

```
EpicGames_UI_AddNotifyDisplaySettingsUpdated()
```

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_UI_AddNotifyDisplaySettingsUpdated"`
bIsExclusiveInput	bool	True when the overlay has switched to exclusive input mode. While in exclusive input mode, no keyboard or mouse input will be sent to the game.
bIsVisible	bool	True when any portion of the overlay is visible.

Example:

```
identifier = EpicGames_UI_AddNotifyDisplaySettingsUpdated()
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_UI_AddNotifyDisplaySettingsUpdated")
{
    var bIsExclusiveInput = async_load[?"bIsExclusiveInput"]
    var bIsVisible = async_load[?"bIsVisible"]
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_UI_GetFriendsVisible

Gets the friends overlay visibility.

EXTERNAL A wrapper around EOS_UI_GetFriendsVisible

Syntax:

```
EpicGames_UI_GetFriendsVisible(accountID)
```

Argument	Description
accountID	The Epic Account ID of the user whose overlay is being updated.

Returns:

```
bool
```

Example:

```
if (EpicGames_UI_GetFriendsVisible(accountID))
{
    PauseGame(); // add logic to pause the game here
}
```

The above code will show an example of how the function should be used. If the Friends UI is being displayed the developer must make sure the game is set to pause.

EpicGames_UI_GetNotificationLocationPreference

Returns the current notification location display preference.

EXTERNAL A wrapper around EOS_UI_GetNotificationLocationPreference

Syntax:

```
EpicGames_UI_GetNotificationLocationPreference()
```

Returns:

```
string
```

Example:

```
show_debug_message("LocationPreference: "
+ EpicGames_UI_GetNotificationLocationPreference())
```

The above code will show an example of how the function should be used.

EpicGames_UI_HideFriends

Hides the active Social Overlay.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_UI_HideFriends

Syntax:

```
EpicGames_UI_HideFriends(accountID)
```

Argument	Description
accountID	The Epic Account ID of the user whose friend list is being shown.

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

Key	Type	Description
type	string	The string `"EpicGames_UI_HideFriends"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_UI_HideFriends()
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_UI_HideFriends")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_UI_RemoveNotifyDisplaySettingsUpdated

Unregister from receiving notifications when the overlay display settings are updated.

EXTERNAL A wrapper around EOS_UI_RemoveNotifyDisplaySettingsUpdated

Syntax:

```
EpicGames_UI_RemoveNotifyDisplaySettingsUpdated(id)
```

Argument	Type	Description
id	real	The handle representing the registered callback (return by EpicGames_UI_AddNotifyDisplaySettingsUpdated)

Returns:

```
N/A
```

Example:

```
handle = EpicGames_UI_AddNotifyDisplaySettingsUpdated()
//...
//...Later
//...
EpicGames_UI_RemoveNotifyDisplaySettingsUpdated(handle)
```

The code sample above enables the display settings update notifications (EpicGames_UI_AddNotifyDisplaySettingsUpdated) and later disables them by refering to the previous generated handle.

EpicGames_UI_SetDisplayPreference

Define any preferences for any display settings.

EXTERNAL A wrapper around EOS_UI_SetDisplayPreference

Syntax:

```
EpicGames_UI_SetDisplayPreference(location)
```

Argument	Type	Description
Location	UINotificationLocation	Preference for notification pop-up locations.

Returns:

```
struct
```

key	type	Description
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code

Example:

```
EpicGames_UI_SetDisplayPreference(EpicGames_UNL_TopLeft)
```

The above code will show an example of how the function should be used. The position of the notifications will now be the rop left corner of the game screen.

EpicGames_UI_ShowFriends

Opens the Social Overlay with a request to show the friends list.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_UI_ShowFriends

Syntax:

```
EpicGames_UI_ShowFriends(accountID)
```

Argument	Description
arg	The argument to be passed in

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

Key	Type	Description
type	string	The string `"EpicGames_UI_ShowFriends"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the *status* code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_UI_ShowFriends()
```

The code sample above save the identifier that can be used inside an **Async Social** event.

```
if (async_load[? "type"] == "EpicGames_UI_ShowFriends")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

UINotificationLocation

The UI Notification Location is used in the following function:

- EpicGames_UI_SetDisplayPreference

Allows to change the positioning of the EOS notifications.

EAuth Scope Flags	Description
`EpicGames_UNL_TopLeft`	Positions the overlay notification on the top left corner
`EpicGames_UNL_TopRight`	Positions the overlay notification on the top right corner
`EpicGames_UNL_BottomLeft`	Positions the overlay notification on the bottom left corner
`EpicGames_UNL_BottomRight`	Positions the overlay notification on the bottom right corner

User Info

Each **Epic Online Services** (EOS) user account has a unique identifier that the service uses internally to refer to the account. The **User Info Interface** bridges the gap between the user's account identifier and information about the user, such as display name, country and preferred language, and so on. You can retrieve this information for both remote users and logged-in, local users.

Functions

These functions are provided for handling user info:

- EpicGames_UserInfo_CopyExternalUserInfoByAccountId
- EpicGames_UserInfo_CopyExternalUserInfoByAccountType
- EpicGames_UserInfo_CopyExternalUserInfoByIndex
- EpicGames_UserInfo_CopyUserInfo
- EpicGames_UserInfo_GetExternalUserInfoCount
- EpicGames_UserInfo_QueryUserInfo
- EpicGames_UserInfo_QueryUserInfoByDisplayName
- EpicGames_UserInfo_QueryUserInfoByExternalAccount

Structures

These are the structures used by this API:

- ExternalUserInfo
- UserInfo

Constants

These are the constants used by this API:

- ExternalAccountType

EpicGames_UserInfo_CopyExternalUserInfoByAccountId

Fetches an external user info for a given external account ID.

NOTE Requires a previous call to EpicGames_UserInfo_QueryUserInfoByExternalAccount to store values in cache.

EXTERNAL A wrapper around EOS_UserInfo_CopyExternalUserInfoByAccountId

Syntax:

```
EpicGames_UserInfo_CopyExternalUserInfoByAccountId(accountID, accountID_target, accountID_exte
```

Argument	Type	Description
accountID	string	The Epic Account ID of the local player requesting the information
accountID_target	string	The Epic Account ID of the player whose information is being retrieved
accountID_external	string	The external account ID associated with the (external) user info to retrieve from the cache; cannot be null

Returns:

```
struct (ExternalUserInfo)
```

Example:

```
var struct =
EpicGames_UserInfo_CopyExternalUserInfoByAccountId(accountID,accountID_target,accountID_externa

if(struct.status == EOS_Success)
{
```

```
        DisplayName = struct.DisplayName
    }
```

The above code will show an example of how the function should be used. The external user info data is returned given the provided accountID.

EpicGames_UserInfo_CopyExternalUserInfoByAccountType

Fetches an external user info for a given external account type.

NOTE Requires a previous call to EpicGames_UserInfo_QueryUserInfoByExternalAccount to store values in cache.

EXTERNAL A wrapper around EOS_UserInfo_CopyExternalUserInfoByAccountType

Syntax:

```
EpicGames_UserInfo_CopyExternalUserInfoByAccountType(accountID,accountID_target,accountType)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the local player requesting the information
accountID_target	string	The Epic Account ID of the player whose information is being retrieved
accountType	ExternalAccountType	Account type of the external user info to retrieve from the cache

Returns:

```
struct (ExternalUserInfo)
```

Example:

```
var struct =
EpicGames_UserInfo_CopyExternalUserInfoByAccountType(accountID,accountID_target,accountType)

if(struct.status == EOS_Success)
{
```

```
        DisplayName = struct.DisplayName
    }
```

The above code will show an example of how the function should be used. The external user info data is returned given the provided account type.

EpicGames_UserInfo_CopyExternalUserInfoByIndex

Fetches an external user info from a given index.

NOTE Requires a previous call to EpicGames_UserInfo_QueryUserInfoByExternalAccount to store values in cache.

EXTERNAL A wrapper around EOS_UserInfo_CopyExternalUserInfoByIndex

Syntax:

```
EpicGames_UserInfo_CopyExternalUserInfoByIndex(accountID, accountID_target, index)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the local player requesting the information
accountID_target	string	The Epic Account ID of the player whose information is being retrieved
index	real	Index of the external user info to retrieve from the cache

Returns:

```
struct (ExternalUserInfo)
```

Example:

```
var count = EpicGames_UserInfo_GetExternalUserInfoCount(accountID,accountID_target)
for(var i = 0 ; i < count ; i ++)
{
    var struct = EpicGames_UserInfo_CopyExternalUserInfoByIndex(i)
    DisplayName = struct.DisplayName
}
```

The above code will show an example of how the function should be used. The external user info data is returned given the provided index (from a cached array).

EpicGames_UserInfo_CopyUserInfo

This function is used to immediately retrieve a copy of user information based on an Epic Account ID, cached by a previous call to EpicGames_UserInfo_QueryUserInfo.

EXTERNAL A wrapper around EOS_UserInfo_CopyUserInfo

Syntax:

```
EpicGames_UserInfo_CopyUserInfo(accountID, accountID_target)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the local player requesting the information
accountID_target	string	The Epic Account ID of the player whose information is being retrieved

Returns:

```
struct (UserInfo)
```

Example:

```
var struct = EpicGames_UserInfo_CopyUserInfo();
if(struct.status == EpicGames_Success)
{
     nickname = struct.Nickname
}
```

The above code will show an example of how the function should be used. The user info data is returned given the provided accountID.

EpicGames_UserInfo_GetExternalUserInfoCount

Fetch the number of external user information that are cached locally.

NOTE Requires a previous call to EpicGames_UserInfo_QueryUserInfoByExternalAccount to store values in cache.

EXTERNAL A wrapper around EOS_UserInfo_GetExternalUserInfoCount

Syntax:

```
EpicGames_UserInfo_GetExternalUserInfoCount(accountID, accountID_target)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the local player requesting the information
accountID_target	string	The Epic Account ID of the player whose information is being retrieved

Returns:

```
real
```

Example:

```
var count = EpicGames_UserInfo_GetExternalUserInfoCount(accountID,accountID_target)
for(var i = 0 ; i < count ; i ++)
{
    var struct = EpicGames_UserInfo_CopyExternalUserInfoByIndex(i)
    DisplayName = struct.DisplayName
}
```

The above code will show an example of how the function should be used. After a successcull call to EpicGames_UserInfo_QueryUserInfoByExternalAccount, the function EpicGames_UserInfo_GetExternalUserInfoCount will return the number of entries in the

query array which can then be accessed using the EpicGames_UserInfo_CopyExternalUserInfoByIndex function.

EpicGames_UserInfo_QueryUserInfo

This function is used to start an asynchronous query to retrieve information, such as display name, about another account.

Once the callback has been fired with a successful **EpicGames Result**, it is possible to call the function:

- EpicGames_UserInfo_CopyUserInfo

to receive an **UserInfo** containing the available information.

This is an asynchronous function that will trigger the **Social Async Event** when the task is finished.

EXTERNAL A wrapper around **EOS_UserInfo_QueryUserInfo**

Syntax:

```
EpicGames_UserInfo_QueryUserInfo(accountID,accountID_target)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the local player requesting the information
accountID_target	string	The Epic Account ID of the player whose information is being retrieved

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_UserInfo_QueryUserInfo"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_UserInfo_QueryUserInfo
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_UserInfo_QueryUserInfo")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_UserInfo_QueryUserInfoByDisplayName

This function is used to start an asynchronous query to retrieve user information by display name. This can be useful for getting the AccountId for a display name.

Once the callback has been fired with a successful **EpicGames Result**, it is possible to call the function:

- EpicGames_UserInfo_CopyUserInfo

to receive an **UserInfo** containing the available information.

This is an asynchronous function that will trigger the **Social Async Event** when the task is finished.

EXTERNAL A wrapper around **EOS_UserInfo_QueryUserInfoByDisplayName**

Syntax:

```
EpicGames_UserInfo_QueryUserInfoByDisplayName(accountID, DisplayName)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the local player requesting the information
DisplayName	string	Display name of the player being queried

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_UserInfo_QueryUserInfoByDisplayName"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_UserInfo_QueryUserInfoByDisplayName()
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_UserInfo_QueryUserInfoByDisplayName")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

EpicGames_UserInfo_QueryUserInfoByExternalAccount

This function is used to start an asynchronous query to retrieve user information by external accounts. This can be useful for getting the AccountId for external accounts.

Once the callback has been fired with a successful EpicGames Result, it is possible to call one of the following functions:

- EpicGames_UserInfo_CopyExternalUserInfoByAccountId
- EpicGames_UserInfo_CopyExternalUserInfoByAccountType
- EpicGames_UserInfo_CopyExternalUserInfoByIndex

to receive a ExternalUserInfo containing the available information.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_UserInfo_QueryUserInfoByExternalAccount

Syntax:

```
EpicGames_UserInfo_QueryUserInfoByExternalAccount(accountID, ExternalAccountId, accountType)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the local player requesting the information
ExternalAccountId	string	External account ID of the user whose information is being retrieved
accountType	real	Account type of the external user info to query

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	**Type**	**Description**
type	string	The string `"EpicGames_UserInfo_QueryUserInfoByExternalAccount"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_UserInfo_QueryUserInfoByExternalAccount()
```

The code sample above save the identifier that can be used inside an Async Social event.

```
if (async_load[? "type"] == "EpicGames_UserInfo_QueryUserInfoByExternalAccount")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.

ExternalUserInfo

The external user info is represented by a struct and contains information about a single external user info. This struct is returned by the following functions:

- EpicGames_UserInfo_CopyExternalUserInfoByAccountId
- EpicGames_UserInfo_CopyExternalUserInfoByAccountType
- EpicGames_UserInfo_CopyExternalUserInfoByIndex

Key	Type	Description
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
DisplayName	string	The display name of the external account. Can be null
AccountId	string	The ID of the external account. Can be null
AccountType	real	The type of the external account, check constants ***EpicGames_EAT_****

Heading

The user info is represented by a struct and contains information about a single external user info. This struct is returned by the function EpicGames_UserInfo_CopyUserInfo.

Key	Type	Description
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
DisplayName	string	The display name. This may be null
Country	string	The name of the owner's country. This may be null
Nickname	string	A nickname/alias for the target user assigned by the local user. This may be null
PreferredLanguage	string	The ISO 639 language code for the user's preferred language. This may be null
AccountID	string	The Epic Account ID of the user

ExternalAccountType

These constants are used to describe the type of an external account or connection and are used by the following functions:

- EpicGames_UserInfo_CopyExternalUserInfoByAccountType

This contants are also part of the ExternalAccountInfo struct.

Account Type Constant	Description
EpicGames_EAT_EPIC	External account is associated with Epic Games
EpicGames_EAT_STEAM	External account is associated with Steam
EpicGames_EAT_PSN	External account is associated with PlayStation(TM)Network
EpicGames_EAT_XBL	External account is associated with Xbox Live
EpicGames_EAT_DISCORD	External account is associated with Discord
EpicGames_EAT_GOG	External account is associated with GOG
EpicGames_EAT_NINTENDO	External account is associated with Nintendo With both EOS Connect and EOS UserInfo APIs, the associated account type is Nintendo Service Account ID. Local user authentication is possible using Nintendo Account ID, while the account type does not get exposed to the SDK in queries related to linked accounts information.
EpicGames_EAT_UPLAY	External account is associated with Uplay
EpicGames_EAT_OPENID	External account is associated with an OpenID Provider
EpicGames_EAT_APPLE	External account is associated with Apple
EpicGames_EAT_GOOGLE	External account is associated with Google
EpicGames_EAT_OCULUS	External account is associated with Oculus
EpicGames_EAT_ITCHIO	External account is associated with itch.io
EpicGames_EAT_AMAZON	External account is associated with Amazon

Other

This section provides miscellaneous/debug functions.

Functions

Provided functions:

- EpicGames_GetVersion
- EpicGames_Logging_SetLogLevel
- EpicGames_Shutdown

Constants

These are the constants used by this API:

- EpicGames Logging Category
- EpicGames Logging Level

EpicGames_GetVersion

Get the version of the EOSSDK binary

EXTERNAL A wrapper around EOS_Friends_AcceptInvite

Syntax:

```
EpicGames_GetVersion()
```

Returns:

```
string
```

Example:

```
show_debug_message("EpicOnlineServices Version: " + EpicGames_GetVersion())
```

The above code will show an example of how the function should be used. The function EpicGames_GetVersion will return the current version of the SDK.

EpicGames_Logging_SetLogLevel

Set the logging level for the specified logging category.

NOTE By default all log categories will callback for Warnings, Errors, and Fatals.

EXTERNAL A wrapper around EOS_Logging_SetLogLevel

Syntax:

```
EpicGames_Logging_SetLogLevel(category, logLevel)
```

Argument	Type	Description
category	EpicGames Logging Category	The specific log category to configure. Use `EpicGames_LC_ALL_CATEGORIES` to configure all categories simultaneously to the same log level. Check the constants EpicGames Logging Category
logLevel	EpicGames Logging Level	the log level to use for the log category. Check the constants EpicGames Logging Level

Returns:

```
N/A
```

Example:

```
EpicGames_Logging_SetLogLevel(EpicGames_LC_ALL_CATEGORIES, EpicGames_LOG_Off)
```

The above code will show an example of how the function should be used. This will turn off logging for all categories.

EpicGames_Shutdown

Tear down the Epic Online Services SDK. Once this function has been called, no more SDK calls are permitted.

Calling anything after EpicGames_Shutdown will result in undefined behavior.

1. `EpicGames_Success` is the returned status if the SDK is successfully torn down.
2. `EpicGames_NotConfigured` is returned status if the extension didn't initialize correctly.
3. `EpicGames_UnexpectedError` is returned status if EpicGames_Shutdown has already been called.

IMPORTANT This should be called at the end of your game.

EXTERNAL A wrapper around EOS_Shutdown

Syntax:

```
EpicGames_Shutdown()
```

Returns:

```
struct
```

key	type	Description
status	EpicGames Result	EpicGames_EResult
status_message	string	Text representation of the ***status*** code

Example:

```
var struct = EpicGames_Shutdown()
if(struct.status == EpicGames_Success)
{
     show_debug_message("Shutdown Success")
}
else
{
     show_debug_message("Shutdown Failed")
}
```

The above code will show an example of how the function should be used. This should be called at the end of your game.

EpicGames Logging Category

These constants represent the available logging categories, and are to be used with the function:

- EpicGames_Logging_SetLogLevel

EpicGames Logging Category Constant	Description
`EpicGames_LC_Core`	Low level logs unrelated to specific services
`EpicGames_LC_Auth`	Logs related to the Auth service
`EpicGames_LC_Friends`	Logs related to the Friends service
`EpicGames_LC_Presence`	Logs related to the Presence service
`EpicGames_LC_UserInfo`	Logs related to the UserInfo service
`EpicGames_LC_HttpSerialization`	Logs related to HTTP serialization
`EpicGames_LC_Ecom`	Logs related to the Ecommerce service
`EpicGames_LC_P2P`	Logs related to the P2P service
`EpicGames_LC_Sessions`	Logs related to the Sessions service
`EpicGames_LC_RateLimiter`	Logs related to rate limiting
`EpicGames_LC_PlayerDataStorage`	Logs related to the PlayerDataStorage service
`EpicGames_LC_Analytics`	Logs related to sdk analytics
`EpicGames_LC_Messaging`	Logs related to the messaging service
`EpicGames_LC_Connect`	Logs related to the Connect service
`EpicGames_LC_Overlay`	Logs related to the Overlay
`EpicGames_LC_Achievements`	Logs related to the Achievements service
`EpicGames_LC_Stats`	Logs related to the Stats service
`EpicGames_LC_UI`	Logs related to the UI service
`EpicGames_LC_Lobby`	Logs related to the lobby service

`EpicGames_LC_Leaderboards`	Logs related to the Leaderboards service
`EpicGames_LC_Keychain`	Logs related to an internal Keychain feature that the authentication interfaces use
`EpicGames_LC_IdentityProvider`	Logs related to external identity providers
`EpicGames_LC_TitleStorage`	Logs related to Title Storage
`EpicGames_LC_Mods`	Logs related to the Mods service
`EpicGames_LC_AntiCheat`	Logs related to the Anti-Cheat service
`EpicGames_LC_Reports`	Logs related to reports client
`EpicGames_LC_Sanctions`	Logs related to the Sanctions service
`EpicGames_LC_ProgressionSnapshots`	Logs related to the Progression Snapshot service
`EpicGames_LC_KWS`	Logs related to the Kids Web Services integration
`EpicGames_LC_RTC`	Logs related to the RTC API
`EpicGames_LC_RTCAdmin`	Logs related to the RTC Admin API
`EpicGames_LC_Inventory`	Logs related to the Inventory service
`EpicGames_LC_ReceiptValidator`	Logs related to the Receipt Validator API
`EpicGames_LC_CustomInvites`	Logs related to the Custom Invites API
`EpicGames_LC_ALL_CATEGORIES`	Not a real log category. Used by EOS_Logging_SetLogLevel to set the log level for all categories at the same time

EpicGames Logging Level

These constants represent the available logging levels. When a log message is output, it has an associated log level. Messages will only be sent to the callback function if the message's associated log level is less than or equal to the configured log level for that category. Use these constants with:

- EpicGames_Logging_SetLogLevel

EpicGames Logging Level Constant	Description
`EpicGames_LOG_Off`	---
`EpicGames_LOG_Fatal`	---
`EpicGames_LOG_Error`	---
`EpicGames_LOG_Warning`	---
`EpicGames_LOG_Info`	---
`EpicGames_LOG_Verbose`	---
`EpicGames_LOG_VeryVerbose`	---

ExternalAccountInfo

An external account information is represented by a struct and contains data about a single account. This struct is returned by the following function:

- EpicGames_Connect_CopyProductUserInfo

The ***status*** member present in the struct can be represented by one of the following values:

- `EpicGames_Success` if the information is available and was correctly returned;
- `EpicGames_InvalidParameters` (extension internal error, should never be returned);
- `EpicGames_NotFound` if the achievement definition is not found;
- `EpicGames_Invalid_ProductUserID` if any of the userid options are incorrect;

Member	Type	Description
status	EResult	The result value of the task
status_message	string	Text representation of the ***status*** code
DisplayName	string	Display name, can be null if not set.
userID	string	The Product User ID of the target user.
AccountId	string	External account ID. May be set to an empty string if the AccountIdType of another user belongs to different account system than the local user's authenticated account. The availability of this field is dependent on account system specifics.
AccountIdType	ExternalAccountType	The identity provider that owns the external account.

LastLoginTime	int64	The POSIX timestamp for the time the user last logged in.

EpicGames_Auth_VerifyIdToken

Verify a given ID token for authenticity and validity.

This is an asynchronous function that will trigger the Social Async Event when the task is finished.

EXTERNAL A wrapper around EOS_Auth_VerifyIdToken

Syntax:

```
EpicGames_Auth_VerifyIdToken(accountID, JsonWebToken)
```

Argument	Type	Description
accountID	string	The Epic Account ID of the local user who is being verify
JsonWebToken	string	The ID token to verify.

Returns:

```
real
```

Triggers:

```
Asynchronous Social Event
```

async_load Contents		
Key	Type	Description

type	string	The string `"EpicGames_Auth_VerifyIdToken"`
status	EResult	The status code for the operation. `EpicGames_Success` indicates that the operation succeeded; other codes indicate errors
status_message	string	Text representation of the ***status*** code
identifier	real	The asynchronous listener ID.

Example:

```
identifier = EpicGames_Auth_VerifyIdToken(accountID, JsonWebToken)
```

The code sample above keeps an handle that can be used inside an Async Social event.

```
if(async_load[? "type"] == "EpicGames_Auth_VerifyIdToken")
if(async_load[? "identifier"] = identifier)
{
    if (async_load[? "status"] == EpicGames_Success)
    {
        show_debug_message(async_load[? "type"] + " succeeded!");
    }
    else
    {
         show_debug_message(async_load[? "type"] + " failed: " + async_load[?
"status_message"])
    }
}
```

The code above matches the response against the correct event **type** and logs the success of the task.