Peter Gustafson writing sample
Epic Games
May 24, 2021

Player Data Storage Interface

The Player Data Storage Interface enables developers using Epic Online Services (EOS) to save encrypted game-specific data to cloud servers.

- Stored data is accessible on any device.

- All file formats are supported.

- Use cases include saved games and replay data.

Access
To access the Player Data Storage Interface, get an EOS_HPlayerDataStorage handle through the [Platform Interface](https://dev.epicgames.com/docs/services/en-US/Interfaces/Platform/) function `EOS_Platform_GetPlayerDataStorageInterface`.

- Check that your `EOS_HPlatform` handle is ticking in order for appropriate callbacks to trigger when operations complete.

- Initialize the EOS Platform with the following extra parameters:

Parameter	Description
`EncryptionKey`	The 64-character 256-bit key.
`CacheDirectory`	Full path to temporary local directory storage.

File Management
Below are details about EOS file names:

- EOS example file path:
```
Directory0/Directory1/DirectoryN/Filename.Extension
```
- `Directory` and `Extension` are optional.
- File names are limited to 64 characters.
- Alphanumeric ASCII characters are supported and the ones below in column A:

A	Description
!	Exclamation mark
-	Hyphen
_	Underscore
+	Plus sign
.	Period
'	Apostrophe

|( )|Parentheses |

Query All Files
Retrieve all files by making a call to `EOS_PlayerDataStorage_QueryFileList` with the following parameters:

Parameter	Description
`ApiVersion`	Set to `EOS_PLAYERDATASTORAGE_QUERYFILELISTOPTIONS_API_LATEST`. |The local user Epic account ID.
`LocalUserId`	The `EOS_ProductUserId` of the local user requesting file data.
`Options`	Pointer to `EOS_PlayerDataStorage_QueryFileListOptions`.
`QueryFileListOptions`	Contains properties related to which user is querying files.
`ClientData`	Optional pointer to help clients track this request, that is returned in the completion callback.
`CompletionCallback`	This function is called when the query operation completes.
`ProductUserId`	The name of the file being queried.
`CopyFileMetadataOptions`	Object containing properties related to which user is requesting metadata, and for which filename .
`OutMetadata`	A copy of the `FileMetadata` structure will be set if successful. This data must be released by calling `EOS_PlayerDataStorage_FileMetadata_Release`.

EOS runs your callback using `callbackEOS_PlayerDataStorage_OnQueryFileListCompleteCallback` and `EOS_PlayerDataStorage_QueryFileListCallbackInfo`.

Query Single File
To query a single file name make a call to `EOS_PlayerDataStorage_QueryFile` with the parameters below:

Parameter	Description
`ApiVersion`	Set to `EOS_PLAYERDATASTORAGE_QUERYFILEOPTIONS_API_LATEST`. |The local user Epic account ID.
`LocalUserId`	The `EOS_ProductUserId` of the local user requesting file data.
`Options`	Pointer to `EOS_PlayerDataStorage_QueryFileOptions`.

EOS runs your callback using `EOS_PlayerDataStorage_OnQueryFileCompleteCallback` and `EOS_PlayerDataStorage_QueryFileListCallbackInfo`, then the file is sent to your cache.

Cached Files

All your cached files are accessible by using commands.

Find One File

Once EOS has retrieved and cached the files, you find one file by calling `EOS_PlayerDataStorage_CopyFileMetadataByFilename`.

Sometimes `EOS_PlayerDataStorage_GetFileMetadataCount` returns multiple files in your cache. Follow these steps to resolve:

1. Re-check that the `ResultCode` in `EOS_PlayerDataStorage_QueryFileCallbackInfo` has a value of `EOS_Success`.
2. Then make a call to `EOS_PlayerDataStorage_CopyFileMetadataByFilename` to access the one file.

Find Number of Files

Make a call to EOS_PlayerDataStorage_GetFileMetadataCount with the parameters below:

Parameter	Description
`ApiVersion`	Set to `EOS_PLAYERDATASTORAGE_FILEMETADATA_API_LATEST`.
`FileSizeBytes`	File size in bytes.
`MD5Hash`	The MD5 Hash of the entire file, in hex digits (hash of encrypted file).
`Filename`	The file name.
`LastModifiedTime`	Date and time the file was saved. POSIX/Unix format (Epoch time).
`CopyFileMetadataOptions`	Object containing properties related to which user is requesting metadata and index.
`OutMetadata`	A copy of the `FileMetadata` structure will be set if successful. This data must be released by calling `EOS_PlayerDataStorage_FileMetadata_Release`.

Access & Modify Files

EOS supports reading, writing, and deleting files from the cloud asynchronously. Reading and writing are streaming operations, and EOS provides handles that the game can use to check their progress, as well as throttle or interrupt them.

Read a File

To read a file from the cloud (or your local cache) make a call to `EOS_PlayerDataStorage_ReadFile` with the parameters below:

| Parameter |Description |

|--|--|
`ApiVersion`	Set to `EOS_PLAYERDATASTORAGE_READFILEOPTIONS_API_LATEST`.
`LocalUserId`	The Account ID of the local user reading the requested file.
`Options`	Pointer to `EOS_PlayerDataStorage_ReadFileOptions`.
`Filename`	The name of the file.
`ReadChunkLengthBytes`	The maximum amount of data to read in a single `EOS_PlayerDataStorage_OnReadFileDataCallback` call.
`ReadFileDataCallback`	Callback function that handles data.
`FileTransferProgressCallback`	Optional callback of transfer progress and called at least once if set.
`ReadOptions`	Object containing properties related to which user is opening the file, what the file's name is, and related mechanisms for copying the data .
`ClientData`	Optional pointer to help clients track the request that is returned in the callbacks.
`CompletionCallback`	Function called when the read operation completes.

A valid Player Data Storage File Request handle if successful, or NULL otherwise. Data contained in the completion callback will have more detailed information about issues with the request in failure cases. This handle must be released when it is no longer needed

The return is `EOS_HPlayerDataStorageFileTransferRequest` which is a handle you use to check the progress of the transfer, get the name of the file, or cancel the transfer.

- To poll the current state make a call to:

```
EOS_PlayerDataStorageFileTransferRequest_GetFileRequestState
```

- Check the name of the file being transferred by calling:

```
EOS_PlayerDataStorageFileTransferRequest_GetFileRequestState
```

- Cancel a request by making a call to:

```
EOS_PlayerDataStorageFileTransferRequest_CancelRequest
```

When downloading multiple files in succession, query the file list first, cache the metadata, and then process your downloads. This avoids having to perform multiple queries and wait for the results between file downloads.

Write a File

Make a call to `EOS_PlayerDataStorage_WriteFile` with the parameters below:

Parameter	Description
`ApiVersion`	Set to `EOS_PLAYERDATASTORAGE_WRITEFILEOPTIONS_API_LATEST`.
`LocalUserId`	The Account ID of the local user reading the requested file.
`Options`	Pointer to `EOS_PlayerDataStorage_ReadFileOptions`.
`Filename`	The name of the file to write.
`ChunkLengthBytes`	The maximum amount of data to write to the file in a single tick.
`WriteFileDataCallback`	The callback that provides chunks of data to EOS.
`FileTransferProgressCallback`	Optional callback of `EOS_PlayerDataStorage_OnFileTransferProgressCallback` of transfer progress; called at least once if set.
`WriteOptions`	Object containing properties of user writing the file, what the file's name is, and related mechanisms for writing the data.
`ClientData`	Optional pointer to help clients track the request and callbacks.
`CompletionCallback`	Called when the write operation completes.

Return
- The return is `EOS_HPlayerDataStorageFileTransferRequest` which is a handle you use to check the progress of the transfer.

After the Transfer
- After the file transfers, EOS runs your `EOS_PlayerDataStorage_OnWriteFileCompleteCallback` callback with `EOS_PlayerDataStorage_WriteFileCallbackInfo` parameter. This parameter indicates success or failure and includes the name of the file. In order to avoid data loss, ensure that this callback runs before letting the user quit the game or shut down the console.

 - To poll the current state make a call to:

```
EOS_PlayerDataStorageFileTransferRequest_GetFileRequestState
```

- Check the name of the file being transferred by calling:

```
EOS_PlayerDataStorageFileTransferRequest_GetFilename
```

 - Cancel a request by making a call to:

```
EOS_PlayerDataStorageFileTransferRequest_CancelRequest
```

Copy a File

To copy a file make a call to `EOS_PlayerDataStorage_DuplicateFile` by using the parameters below:

Parameter	Description
`ApiVersion`	Set to `EOS_PLAYERDATASTORAGE_DUPLICATEFILEOPTIONS_API_LATEST`.
`LocalUserId`	The Account ID of the local user reading the requested file.
`Options`	Pointer to `EOS_PlayerDataStorage_DuplicateFileOptions`.
`SourceFilename`	The name of the file to copy.
`DestinationFilename`	The name of the duplicate file.
`DuplicateOptions`	Object containing properties which user is duplicating the file and the source and destination file names.
`ClientData`	Optional pointer to help clients track request that is returned in the callback.
`CompletionCallback`	This function is called when the duplicate operation completes.

 - Upon completion, EOS will run your `EOS_PlayerDataStorage_OnDuplicateFileCompleteCallback` function with an `EOS_PlayerDataStorage_DuplicateFileCallbackInfo` parameter.
 - File duplication fails when the user doesn't own the original file, or if the user is out of storage space.
 - When `DestinationFilename` identifies a file that already exists, that file is replaced with the duplicate file.

Delete a File

Delete a file by calling `EOS_PlayerDataStorage_DeleteFileOptions` with the parameters below:

Parameter	Description
`ApiVersion`	Set to `EOS_PLAYERDATASTORAGE_DELETEFILEOPTIONS_API_LATEST`.
`LocalUserId`	The Account ID of the local user reading the requested file.
`Options`	Pointer to `EOS_PlayerDataStorage_DeleteFileOptions`.
`Filename`	The name of the file to write.
`DeleteOptions`	Object containing properties which user is

deleting the file and the name of the file.|
| `ClientData` | Optional pointer to help clients track request that is returned in the callback.|
| `CompletionCallback` | This function is called when the delete operation completes.|

Upon completion, EOS will run your callback (of type `EOS_PlayerDataStorage_OnDeleteFileCompleteCallback`) and pass a parameter of type `EOS_PlayerDataStorage_DeleteFileCallbackInfo` to it.

Data Caching and Encryption

The Player Data Storage Interface caches file data and metadata in the `CacheDirectory` folder on the client system.

- EOS uses this data instead of streaming from the cloud, for both read and write operations, using MD5 checksum testing to prevent data corruption.
- The data monitors when the locally cached file is identical to the version on the cloud, in which case read and write operations can complete in a single callback cycle.
- Any successful read or write operation updates the cache, while deletion clears the cached file and removes its metadata.
- EOS does not clear the cache upon shutdown, enabling reuse of cached files in future sessions.

Player Data Storage files are stored with encryption using the key you provide during EOS Platform initialization. The encryption key itself is not stored on the cloud, preventing both Epic Games and malicious third parties from using the key to access clients' data. Encryption keys are automatically modified for each user, so different users can't decrypt each other's data.

Usage Limitations

The Player Data Storage Interface enables developers using EOS to save encrypted, user-specific, game-specific data to cloud servers. Data you store through the interface is accessible to the user on any device where they can login.

The service applies limits to both storage amounts and usage rates. Throttling enforces usage rate limitations, and the service can delete files to enforce storage space limitations. For example, uploading a file larger than the maximum individual file size results in the service automatically deleting that file.

Below are limitations per minute:

| Usage Type | Limitation |

Read or write	1000
Maximum individual file size	200 MB
Auto-delete file size	400 MB
Maximum individual folder size	400 MB
Maximum files per user	1000

- In the event a user attempts to upload past either storage space limit, the upload operation fails with an error.
- Transferring too much data at once will not cause an error, but the user is prevented from uploading additional files until that file is deleted.
- Files that exceed the auto-delete file size are deleted following the upload.

Player Data Storage vs EGS Cloud Saves

Epic offers two different cloud save services:

1. Launcher Cloud Saves are tied to games within the Epic Games Launcher.

2. Cloud Saves from the Launcher are handled entirely on the Epic side once configured.

EOS Player Data Storage requires the player to be online, and is driven game-side using the EOS APIs.