


Stats Interface

Upload stats and data to online services and complete stats queries.



 Learn to use this **Beta** feature, but use caution when shipping with it.

The **Online Services Stats Interface** is used to upload stats and data to online services and complete stats queries. Stats Interface functionality is also used by other interfaces that rely on user gameplay statistics such as the Online Services' Achievements and Leaderboards Interfaces.

API Overview

The following table provides a high-level description of the functions included in the Stats Interface.

Function	Description
Update	

Function	Description
<code>UpdateStats</code>	Upload stats to the platform
Query	
<code>QueryStats</code>	Query the stats of a user and cache the result in the interface.
<code>BatchQueryStats</code>	Query the stats of a group of users and cache the results in the interface.
Get	
<code>GetCachedStats</code>	Retrieve the cached user stats stored after a call to <code>QueryStats</code> or <code>BatchQueryStats</code> .
Event Listening	
<code>OnStatsUpdated</code>	An event will fire as a result of changes to user stats.

Configuration

You can use the Stats Interface with either a corresponding platform backend or a `StatsNull` implementation. To use the Stats Interface, you must first configure the Stats Interface in your `DefaultEngine.ini` file:

DefaultEngine.ini

```
1 [OnlineServices.Stats]
2 +StatDefinitions=(Name=<STAT_NAME>, Id=<ID_NUMBER>, ModifyMethod=<METHOD>,
   DefaultValue="<TYPE>:<DEFAULT_VALUE>")
```

 Copy full snippet

Stat Definitions consist of the following fields:

- **Name**: The name of the stat.
 - This is the name that will be used to update and query stats with **UpdateStats** and **QueryStats** respectively.
- **Id**: The ID of the stat.
 - This is the corresponding configured stat ID in the platform portal.
- **ModifyMethod**: Method prescribing how the stat will be updated.
 - For non-**StatsNull** implementations, the Modify Method is configured in the platform portal.
 - The Modify Method is used by the Achievements Interface on all implementations when using Title Managed achievements to determine whether an achievement meets the prescribed unlock rules.
- **DefaultValue**: The type and default value of the stat.
 - This prescribes the initial value of the stat.

To unlock achievements and update leaderboards with stats, you must specify corresponding stats in the Achievements and Leaderboards config sections of **DefaultEngine.ini**.

Configuration Example

Here is a configuration example for the Online Services Stats interface:

DefaultEngine.ini

```

1 [OnlineServices.Stats]
2 +StatDefinitions=(Name=Stat_Use_Largest, Id=0, ModifyMethod=Largest,
   DefaultValue="Int64:0")
3 +StatDefinitions=(Name=Stat_Use_Smallest, Id=1, ModifyMethod=Smallest,
   DefaultValue="Int64:999")
4 +StatDefinitions=(Name=Stat_Use_Set, Id=2, ModifyMethod=Set,
   DefaultValue="Int64:0")
5 +StatDefinitions=(Name=Stat_Use_Sum, Id=3, ModifyMethod=Sum,
   DefaultValue="Int64:0")
6 +StatDefinitions=(Name=Stat_Type_Bool, Id=4, ModifyMethod=Set,
   DefaultValue="Bool:True")
7 +StatDefinitions=(Name=Stat_Type_Double, Id=5, ModifyMethod=Smallest,
   DefaultValue="Double:9999.999")

```

 Copy full snippet

Examples

This section contains a variety of code examples that guide you on how to:

- [Query Stats](#)
- [Get Cached Stats](#)
- [Listen for an event](#)
- [Execute a Console Command](#)

Query Stats


```
1 UE::Online::IOnlineServicesPtr OnlineServices = UE::Online::GetServices();
2 UE::Online::IStatsPtr Stats = OnlineServices->GetStatsInterface();
3
4 UE::Online::FQueryStats::Params Params;
5 Params.LocalAccountId = LocalAccountId;
6 Params.TargetAccountId = TargetAccountId;
7 Params.StatNames = {"StatA", "StatB"};
8
9 // See Note below Walkthrough for more information about this OnComplete
call
10 Stats->QueryStats(MoveTemp(Params)).OnComplete([](const
    UE::Online::TOnlineResult<FQueryStats>& Result)
11 {
12     if (Result.IsError())
13     {
14         const UE::Online::FOnlineError OnlineError = Result.GetErrorValue();
15         // Process OnlineError
16         return;
17     }
18     const UE::Online::FQueryStats::Result QueriedStats = Result.GetOkValue();
19     // Process QueriedStats
20 });
```

 Copy full snippet

Walkthrough

1. Use the default online services by calling `GetServices` with no parameters specified:

```
UE::Online::IOnlineServicesPtr OnlineServices = UE::Online::GetServices
```

 Copy full snippet

2. Access the Stats Interface for the default online services:

```
UE::Online::IStatsPtr Stats = OnlineServices->GetStatsInterface();
```

 Copy full snippet

3. Instantiate the parameters necessary to query the `StatNames` of the `TargetAccountId`:

```
1 UE::Online::FQueryStats::Params Params;  
2 Params.LocalAccountId = LocalAccountId;  
3 Params.TargetAccountId = TargetAccountId;  
4 Params.StatNames = {"StatA", "StatB"};
```

 Copy full snippet

4. Handle the `QueryStats.OnComplete` callback by processing the error or the queried stats:

```
1 Stats->QueryStats(MoveTemp(Params)).OnComplete([](const  
    UE::Online::TOnlineResult<FQueryStats>& Result)  
2 {  
3     if (Result.IsError())  
4     {  
5         const UE::Online::FOnlineError OnlineError = Result.GetErrorValue();  
6         // Process OnlineError  
7         return;  
8     }  
9     const UE::Online::FQueryStats::Result QueriedStats =  
        Result.GetOkValue();  
10    // Process QueriedStats  
11    });
```

 Copy full snippet

To bind to a member function, always prefer to use a UObject-derived class or a class that inherits from `TSharedFromThis` and use

```
.OnComplete(this, &MyClass::OnQueryStatsComplete)
```



Copy full snippet

This automatically selects `CreateUObject`, `CreateThreadSafeSP`, or `CreateSP`. The safest delegate creation call will be used. For more information, refer to the [Callback Format](#) section of the Online Services Overview page.

Get Cached Stats

```
1 UE::Online::IOnlineServicesPtr OnlineServices = UE::Online::GetServices();
2 UE::Online::IStatsPtr Stats = OnlineServices->GetStatsInterface();
3
4 UE::Online::TOnlineResult<FGetCachedStats> CachedStats = Stats-
  >GetCachedStats({});
5 if (CachedStats.IsError())
6 {
7   UE::Online::FOnlineError OnlineError = CachedStats.GetErrorValue();
8   // Process OnlineError
9   return;
10 }
11
12 UE::Online::FGetCachedStats::Result& CachedStatsData =
  CachedStats.GetOkValue();
13 // Process CachedStatsData
```

Copy full snippet

Walkthrough

1. Use the default online services by calling `GetServices` with no parameters specified and access the Stats Interface:

```

1 UE::Online::IOnlineServicesPtr OnlineServices =
  UE::Online::GetServices();
2 UE::Online::IStatsPtr Stats = OnlineServices->GetStatsInterface();

```


 Copy full snippet

2. Retrieve the cached stats through the Stats Interface with `Stats->GetCachedStats`:

```

UE::Online::TOnlineResult<UE::Online::FGetCachedStats> CachedStats = Stats->GetCachedStats();

```

 Copy full snippet

3. Handle the `CachedStats` by processing the error or the cached stats data:

```

1 if (CachedStats.IsError())
2 {
3   UE::Online::FOnlineError OnlineError = CachedStats.GetErrorValue();
4   // Process OnlineError
5   return;
6
7 }
8 UE::Online::FGetCachedStats::Result& CachedStatsData =
  CachedStats.GetOkValue();
9 // Process CachedStatsData

```

 Copy full snippet


Listen for an Event

Event listening is handled differently than synchronous and asynchronous functions. An `FOnlineEventDelegateHandle` is created to handle the result of the `OnStatsUpdated` event, then `Unbind` must be called in your shutdown code to ensure proper destruction.

Walkthrough

1. Declare an event handle in your class for the Stat interface.

```
UE::Online::FOnlineEventDelegateHandle StatEventHandle;
```

 Copy full snippet


2. In your init code, initialize the default online services, access the Stats interface, and process the stats when an event happens.

```
1 UE::Online::IOnlineServicesPtr OnlineServices =  
  UE::Online::GetServices();  
2 UE::Online::IStatsPtr Stats = OnlineServices->GetStatsInterface();  
3 StatEventHandle = Stats->OnStatsUpdated().Add([](const  
  UE::Online::FStatsUpdated& StatsUpdated)  
4 {  
5   // custom logic inside this lambda  
6 });
```

 Copy full snippet

3. Ensure that you unbind the event handler in your shutdown code.

```
StatEventHandle.Unbind();
```

 Copy full snippet

Execute a Console Command

For the general command-line syntax to run an async interface with a console command, refer to the [Online Services Overview](#) documentation.

Example

To run the `QueryStats` function, execute the following console command:

```
OnlineServices Index=0 Stats QueryStats 0 0 ["StatA", "StatB"]
```

 Copy full snippet

This command calls `QueryStats` from the Stats Interface with the default online services for the zeroth local user. In particular, the above command queries the default online services for `StatA` and `StatB` of this user.

Reset Stats Data

During development and testing, the `ResetStats` function resets all provided player stats for the current title. Although policies vary across online services, you should not expect this function to work outside a testing environment. Be sure to remove any code that uses `ResetStats` from shipping builds, or use compile-time logic to mask the code like this:

```
1 #if !UE_BUILD_SHIPPING
2 // Code block with call to ResetStats
3 #endif
```

 Copy full snippet

More Information

Header File

Consult the `Stats.h` header file directly for more information as needed. The Stats Interface header file `Stats.h` is located in the directory:

```
Engine\Plugins\Online\OnlineServices\Source\OnlineServicesInterface\Public\Onlin
```

 Copy full snippet

For instructions on how to obtain the UE source code, refer to our documentation on [Downloading Unreal Engine Source Code](#).

Function Parameters and Return Types

Refer to the [Functions](#) section of the [Online Services Overview](#) page for an explanation of function parameters and return types, including how to pass parameters and processing the results when functions return.