Friends Interface

Overview of the friend interface

Playing games with your friends and meeting new players online is an important part of many online services. The **Friends Interface** contains features to manage a user's list of social contacts, including adding, removing, and blocking other users.

Managing Friends

Friends lists are stored on the online service's servers, and can change during a session as friends are added or removed, join and leave games and sessions, or log in and out of the service. As a result, managing these lists involves querying the server for the latest information, then caching that information and using it in your game.

Retrieving Friends Lists

The first step in dealing with a user's friends lists is generally to call `ReadFriendsList`, which retrieves the most up-to-date version of a named friends list belonging to a specified local user. Valid list names can be found in the `EFriendsList` enumerated type, and can be converted into strings by the provided `ToString` function. Because it queries a remote

machine, the `ReadFriendsList` is asynchronous and will call a delegate of type `FOnReadFriendsListComplete` when complete.

> On success, this call caches the list so that it can be examined later without repeatedly querying the remote machine, or requiring developers to write their own caching code. It also updates presence status data for the users on the list. The data returned by `FOnReadFriendsListComplete` only contains information regarding the success or failure of the `ReadFriendsList` operation.

Examining Friends Lists

After a successful call to `ReadFriendsList` retrieves and caches a list, developers can use `GetFriendsList` to get a copy of the list itself, or `GetFriend` to retrieve an individual friend from the list. In addition, a known user's `FUniqueNetId` can be passed to the `IsFriend` function to check a specified list for that user.

> Friends lists can change at any time, both from in-game events like meeting new players, and from out-of-game events like the user modifying the account from a separate system. Calling `ReadFriendsList` to keep the cached lists up-to-date should be considered.

Inviting a Friend

The `SendInvite` function will send an invitation to another user, identified by that user's `FUniqueNetId`. Upon acceptance, the user will be added to the specified list by the online service. A delegate of type `FOnSendInviteComplete` will be called when this operation finishes, but this only means that the invitation has been sent (or has failed to send), not that the intended recipient has received or responded to it. Some online services may have custom user interfaces for sending invitations, and these may open automatically when `SendInvite` is called.

All `SendInvite` calls will eventually trigger the `FOnSendInviteComplete` delegate. This includes cases where an external UI is opened and the user cancels it.

Accepting or Rejecting an Invitation

When an invitation from another user has arrived, a delegate of type `FOnInviteReceivedDelegate` will be called, containing the `FUniqueNetId` of the sender and the recipient. The invited user can then call `AcceptInvite` or `RejectInvite` to respond, specifying the name of the list where the new friend should appear. `AcceptInvite` uses a delegate of type `FOnAcceptInviteComplete` to communicate the results of the operation, while `RejectInvite` uses an `FOnRejectInviteComplete` delegate.

Deleting a Friends List

The online service can be instructed to delete a friends list through the asynchronous `DeleteFriendsList` function. When complete, a delegate of type `FOnDeleteFriendsListComplete` will be called.

Deleting a Friend

To remove a friend from a list belonging to a local user, call the `DeleteFriend` function. When the operation is complete, a delegate of type `FOnDeleteFriendComplete` will be called. It may be possible for a friend to exist in multiple lists on some online services; if this is the case, this function will only remove the friend from the specified list.

Dealing with Players Met Online

Online services often keep a separate list of players that a user has encountered recently, such as in a public gaming session, but has not added to the friends list or blocked. Like friends lists, the list of recently-met players is handled by querying the online service and then caching the list.

Retrieving the List of Recently-Met Players

The `QueryRecentPlayers` makes an asynchronous call to the online service, calling a delegate of type `FOnQueryRecentPlayersComplete` upon completion. If successful, the Friends Interface will cache the list locally.

Examining Recently-Met Players

Once a successful call to `QueryRecentPlayers` has retrieved the list of recently-met players, the `GetRecentPlayers` function will return the cached array. The individual elements of the array contain the user's data as well as a function to tell when the player was last seen online.

Managing the Block List

Many online services provide the ability for a user to prevent specific other users from contacting or playing with them through the service. The Friends Interface can retrieve and cache a list of blocked users, as well as utilize the online service's block and unblock features.

Listing Currently-Blocked Users

To retrieve the list of blocked users, call the `QueryBlockedPlayers` function. This function is asynchronous, and will call a delegate of type `FOnQueryBlockedPlayersComplete` when it completes. In addition, any changes made to the list of blocked users will result in notification through an `FOnBlockListChange` delegate.

Blocking and Unblocking Users

The `BlockPlayer` and `UnblockPlayer` functions will make asynchronous calls to the online service, requesting that a specific player, identified by an `FUniqueNetId`, be blocked or unblocked by a local player. When these operations complete, they will return success or failure information through delegates of type `FOnBlockedPlayerComplete` and `FOnUnblockedPlayerComplete`, respectively.