Purpose of CryLobby

CryLobby is a collection of modules abstracting away platform specific Lobby and Matchmaking services. Crytek currently implemented support for the following services:

- Playstation Network
- Xbox Live
- LAN

Starting from CryEngine 5.2, CryLobby is decoupled from the engine into *CryExtensions/CryLobby* folder and is completely optional for building and running the engine. You can control the CryLobby instantiation by using *InitializeEngineModule(...)* from the game. For more information, please see *CGame::Init()* from GameSDK sample:

```
if (!gEnv->pSystem->InitializeEngineModule("CryLobby", "EngineModule_CryLobby", false))
{
 CryWarning(VALIDATOR_MODULE_GAME, VALIDATOR_ERROR, "Error creating Lobby System!");
}
```

Interfaces

- **ICryLobby**: Central access point for lobby services, retrievable via *gEnv->pNetwork->GetLobby()*.
- **ICryMatchmaking**: Handles session based tasks, such as Creating, Searching or Joining rooms.
- **ICryVoice**: Wraps up VOIP; currently used primarily for muting player groups.
- **ICryStats**: Handles persistent stats and leaderboard types of operations.
- **ICryReward**: Handles achievements.
- **ICryTCPService**: Basic TCP implementation (used for telemetry).
- **ICryFriends**: Used to specify Friends list management such as list friends or send game invite.
- **ICryLobbyUI**: Access to xmb - show gamercards, show friends list, set rich presence.

CryLobby API

A service is either eCLS_LAN or eCLS_Online. At present you can only have 1 of each. LAN only provides the *ICryMatchMaking* interface.

Online provides a more complete set. In general, all lobby functions expect a callback and return a task identifier.

CryLobby Configuration

Due to differences in third party functionality, an additional callback may be executed at any time.

CryLobbyConfigurationCallback is used to provide details about your game that are not exposed via the normal API.

CryLobby Usage

Events

Game can register as being interested in events. For example, the Player connected to LIVE, ethernet cable removed, game invite received.

```
gEnv->pNetwork->GetLobby()->RegisterEventInterest(eCLSE_OnlineState, callback, userArg);
```

Callback

```
typedef void (*CryLobbyEventCallback)(UCryLobbyEventData eventData, void *userParam);
```

Most functions in CryLobby and subsystems return an error code.

- eCLE_Success = 0
- eCLE_TooManyTasks
- eCLE_InvalidSession

- ...

These error codes also returned in callback functions. A full list can be found in ICryLobby.h

Cancelling tasks

This happens for example when your listener class is being destroyed.

```
gEnv->pNetwork->GetLobby()->CancelTask(taskId);
```

A taskId is an output parameter in any lobby function that also takes a callback.

It is recommended to avoid wherever possible, especially on certain tasks such as *SessionCreate*, *SessionJoin* or *SessionDelete*. This may not stop the task from running (could be too late to abort), but it will stop the callback from firing.

CryMatchMaking

Manages sessions

Restrictions:

- Maximum of 4 sessions (MAX_MATCHMAKING_SESSIONS)
- Up to 64 players (MAX_LOBBY_CONNECTIONS)

Connections limit is an overall limit which is shared between sessions.

Most functions require a session handle, this is returned by SessionCreate or SessionJoin.

SessionCreate

```
virtual ECryLobbyError SessionCreate(uint32* users, int numUsers, uint32 flags, SCrySessionData* data,
CryLobbyTaskID* taskID, CryMatchmakingSessionCreateCallback cb, void* cbArg) = 0;
```

The user pointer along with the numUsers count parameters passes an array of user indexes and correspond to the pad index.

The flags parameter can be one of the following:

- CRYSESSION_CREATE_FLAG_SEARCHABLE
- CRYSESSION_CREATE_FLAG_INVITABLE
- CRYSESSION_CREATE_FLAG_NUB
- CRYSESSION_CREATE_FLAG_MIGRATABLE
- CRYSESSION_LOCAL_FLAG_CAN_SEND_SERVER_PING

The Data argument represents the advertised/synched values which can be changed using SessionUpdate.

SessionJoin

```
virtual ECryLobbyError SessionJoin(uint32* users, int numUsers, uint32 flags, CrySessionID id,
CryLobbyTaskID* taskID, CryMatchmakingSessionJoinCallback cb, void* cbArg) = 0;
```

The function is similar to SessionCreate. It takes a CrySessionID as argument. Provided by either a search callback, an invite callback or from a game packet in another session.

SessionDelete

```
virtual ECryLobbyError SessionDelete(CrySessionHandle h, CryLobbyTaskID* taskID,
CryMatchmakingCallback cb, void* cbArg) = 0;
```

CrySessionHandle is provided by either SessionCreate or SessionJoin callback.

User Packets

Send arbitrary packets. Doesn't use a fixed format.

Here is an example:

```
CCryLobbyPacket packet;
Uint32 bufferSize = CryLobbyPacketReliableHeaderSize + CryLobbyPacketUINT8Size;

if (packet.CreateWriteBuffer(bufferSize))
{
  packet.StartWrite(type, reliable);
  packet.WriteUINT8(3);
}
```

```
gEnv->pNetwork->GetLobby()->GetMatchMaking()->SendToAllClients(&packet, sessionHandle);
```

Also look at *SendToClient* and *SendToServer*.

Events

Handled through same mechanism as CryLobbyEvents.

```
gEnv->pNetwork->GetLobby()->RegisterEventInterest(eCLSE_UserPacket, cb, this);
```

MatchMaking events will have a sessionHandle as part of the data in the callback, check this against the current session since you don't register for events on a specific session.