



Photon Unity Networking 2 2.12

[Main Page](#)[Related Pages](#)[Modules](#)[Classes](#)

General Documentation

Brief overview of Photon, subscriptions, hosting options and how to start.

Table of Contents

↓ [Photon](#)

Photon Unity Networking - First steps

When you import PUN, the "Wizard" window will pop up. If not, find it in the Window menu as "Photon Unity Networking". In the Wizard, either enter your email address to register for the **Photon** Cloud, enter the AppId of an existing account or skip this step for the time being.

The Wizard creates a configuration in the project, named: PhotonServerSettings.

PUN consists of quite a few files, however most functionality is concentrated into: **Photon.Pun.PhotonNetwork**. This class contains all functions and variables typically needed. If you ever have custom requirements, you can always modify the source files - this plugin is just an implementation of **Photon** after all.

To learn how this API works, visit the [online documentation for PUN](#)

Photon

Photon Unity Networking (PUN) always connects to a dedicated **Photon** server, which provides matchmaking, load balancing and in-room communication for players.

Behind the scenes PUN uses more than one server: A "Name Server" acts as point of entry and provides a list of regional "Master Servers". A Master Server keeps track of rooms and provides the Matchmaking, while several "Game Servers" run the actual rooms (matches).

Exit Games Cloud

The Exit Games Cloud provides hosted and load balanced **Photon** servers for you, fully managed by Exit Games. Free trials are available and subscription costs for commercial use are competitively low.

The Public Cloud service runs a fixed logic, so the clients need to be authoritative.

Clients are separated by “application id” (identifies your game title) and a “game version”. Changing the game version helps separate players with new and old client builds.

Subscriptions bought in Asset Store

If you bought a package with **Photon** Cloud Subscription in the Asset Store:

- Register a Photon Cloud Account [at this link](#)
- Create an App and get your AppID from the [Dashboard](#)
- Send a Mail to: developer@photonengine.com
- With:
 - Your Name and Company (if applicable)
 - Invoice/Purchase ID from the Asset Store
 - Photon Cloud AppID

Photon Server SDK

As alternative to the **Photon** Cloud service, you can run your own server and develop server side logic on top of our "Load Balancing" C# solution. This gives you full control of the server logic.

The Photon Server SDK can be downloaded [at this link](#)

Read about how to start the server [here](#).

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Network Simulation GUI

Simple GUI element to control the built-in network condition simulation.

The Photon client library can simulate network conditions for lag (message delay) and loss, which can be a good tool for developer when testing with a local server or on near perfect network conditions.

To use it, add the component

Photon.Pun.UtilityScripts.PhotonLagSimulationGui to an enabled GameObject in your scene. At runtime, the top left of the screen shows the current roundtrip time (RTT) and the controls for network simulation:

- RTT: The roundtrip time is the average of milliseconds until a message was acknowledged by the server. The variance value (behind the +/-) shows how stable the rtt is (a lower value being better).
- "Sim" toggle: Enables and disables the simulation. A sudden, big change of network conditions might result in disconnects.
- "Lag" slider: Adds a fixed delay to all outgoing and incoming messages. In milliseconds.
- "Jit" slider: Adds a random delay of "up to X milliseconds" per message.
- "Loss" slider: Drops the set percentage of messages. You can expect less than 2% drop in the internet today.



Network Statistics GUI

The PhotonStatsGui is a simple GUI component to track and show network-metrics at runtime.

Usage

Just add the **Photon.Pun.UtilityScripts.PhotonStatsGui** component to any active GameObject in the hierarchy. A window appears (at runtime) and shows the message count.

A few toggles let you configure the window:

- buttons: Show buttons for "stats on", "reset stats" and "to log"
- traffic: Show lower level network traffic (bytes per direction)
- health: Show timing of sending, dispatches and their longest gaps

Message Statistics

The top most values showns are counter for "messages". Any operation, response and event are counted. Shown are the total count of outgoing, incoming and the sum of those messages as total and as average for the timespan that is tracked.

Traffic Statistics

These are the byte and packet counters. Anything that leaves or arrives via network is counted here. Even if there are few messages, they could be huge by accident and still cause less powerful clients to drop connection. You also see that there are packages sent when you don't send messages. They keeps the connection alive.

Health Statistics

The block beginning with "longest delta between" is about the performance of your client. We measure how much time passed between consecutive calls of send and dispatch. Usually they should be called ten times per second. If these values go beyond one second, you should check why Update() calls are delayed.

Button "Reset"

This resets the stats but keeps tracking them. This is useful to track message counts for different situations.

Button "To Log"

Pressing this simply logs the current stat values. This can be useful to have a overview how things evolved or just as reference.

Button "Stats On" (Enabling Traffic Stats)

The Photon library can track various network statistics but usually this feature is turned off. The PhotonStatsGui will enable the tracking and show those values.

The "stats on" toggle in the Gui controls if traffic stats are collected at all. The "Traffic Stats On" checkbox in the Inspector is the same value.



Public API Module

The Public API module rounds up the most commonly used classes of PUN.

The classes which are most commonly used, are grouped into a Public API module, which is only a documentation structure. Classes like **Photon.Pun.PhotonNetwork** and **Photon.Pun.MonoBehaviourPunCallbacks** are good entry points to learn how to code with PUN.

Typically, classes for internal use are not public but there are a few exceptions to this where access may be of use, if you know what you're doing.

Open the Public API module



Photon Unity Networking 2 2.12

[Main Page](#)[Related Pages](#)[Modules](#)[Classes](#)

Modules

Here is a list of all modules:

Public API	Groups the most important classes that you need to understand early on
Optional Gui Elements	Useful GUI elements for PUN
Callbacks	Callback Interfaces

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

[Main Page](#)[Related Pages](#)[Modules](#)[Classes](#)[Classes](#) | [Enumerations](#) | [Functions](#)

Public API

Groups the most important classes that you need to understand early on. [More...](#)

Classes

class **PhotonNetwork**

The main class to use the **PhotonNetwork** plugin. This class is static. [More...](#)

class **PhotonView**

A **PhotonView** identifies an object across the network (viewID) and configures how the controlling client updates remote instances. [More...](#)

struct **PhotonMessageInfo**

Container class for info about a particular message, RPC or update. [More...](#)

class **PhotonStream**

This container is used in OnPhotonSerializeView() to either provide incoming data of a **PhotonView** or for you to provide it. [More...](#)

Enumerations

enum **ClientState**

State values for a client, which handles switching **Photon** server types, some operations, etc. [More...](#)

enum **PunLogLevel**

Used to define the level of logging output created by the PUN classes. Either log errors, info (some more) or full. [More...](#)

enum **RpcTarget**

Enum of "target" options for RPCs. These define which remote clients get your RPC call. [More...](#)

Functions

void **OnPhotonSerializeView** (PhotonStream stream,
PhotonMessageInfo info)

Called by PUN several times per second, so that your script
can write and read synchronization data for the **PhotonView**.
[More...](#)

Detailed Description

Groups the most important classes that you need to understand early on.

Enumeration Type Documentation

enum ClientState

strong

State values for a client, which handles switching **Photon** server types, some operations, etc.

Enumerator	
PeerCreated	Peer is created but not used yet.
Authenticating	Transition state while connecting to a server. On the Photon Cloud this sends the AppId and AuthenticationValues (UserID).
Authenticated	Not Used.
JoiningLobby	The client sent an OpJoinLobby and if this was done on the Master Server, it will result in. Depending on the lobby, it gets room listings.
JoinedLobby	The client is in a lobby, connected to the MasterServer. Depending on the lobby, it gets room listings.
DisconnectingFromMasterServer	Transition from MasterServer to GameServer.
ConnectingToGameServer	Transition to GameServer (client authenticates and joins/creates a room).
ConnectedToGameServer	Connected to GameServer (going to auth and join game).

Joining	Transition state while joining or creating a room on GameServer.
Joined	The client entered a room. The CurrentRoom and Players are known and you can now raise events.
Leaving	Transition state when leaving a room.
DisconnectingFromGameServer	Transition from GameServer to MasterServer (after leaving a room/game).
ConnectingToMasterServer	Connecting to MasterServer (includes sending authentication values).
Disconnecting	The client disconnects (from any server). This leads to state Disconnected.
Disconnected	The client is no longer connected (to any server). Connect to MasterServer to go on.
ConnectedToMasterServer	Connected to MasterServer. You might use matchmaking or join a lobby now.
ConnectingToNameServer	Client connects to the NameServer. This process includes low level connecting and setting up encryption. When done, state becomes ConnectedToNameServer.
ConnectedToNameServer	Client is connected to the NameServer and established encryption already. You should call OpGetRegions or ConnectToRegionMaster.

DisconnectingFromNameServer

Clients disconnects (specifically) from the NameServer (usually to connect to the MasterServer).

enum PunLogLevel

strong

Used to define the level of logging output created by the PUN classes. Either log errors, info (some more) or full.

Enumerator

ErrorsOnly	Show only errors. Minimal output. Note: Some might be "runtime errors" which you have to expect.
Informational	Logs some of the workflow, calls and results.
Full	Every available log call gets into the console/log. Only use for debugging.

enum RpcTarget

strong

Enum of "target" options for RPCs. These define which remote clients get your RPC call.

Enumerator

All	Sends the RPC to everyone else and executes it immediately on this client. Player who join later will not execute this RPC.
Others	Sends the RPC to everyone else. This client does not execute the RPC. Player who join later will not execute this RPC.
MasterClient	Sends the RPC to MasterClient only. Careful: The MasterClient might disconnect before it executes the RPC and that might cause dropped RPCs.

AllBuffered	Sends the RPC to everyone else and executes it immediately on this client. New players get the RPC when they join as it's buffered (until this client leaves).
OthersBuffered	Sends the RPC to everyone. This client does not execute the RPC. New players get the RPC when they join as it's buffered (until this client leaves).
AllViaServer	<p>Sends the RPC to everyone (including this client) through the server.</p> <p>This client executes the RPC like any other when it received it from the server. Benefit: The server's order of sending the RPCs is the same on all clients.</p>
AllBufferedViaServer	<p>Sends the RPC to everyone (including this client) through the server and buffers it for players joining later.</p> <p>This client executes the RPC like any other when it received it from the server. Benefit: The server's order of sending the RPCs is the same on all clients.</p>

Function Documentation

```
void OnPhotonSerializeView ( PhotonStream      stream,  
                             PhotonMessageInfo info  
                             )
```

Called by PUN several times per second, so that your script can write and read synchronization data for the **PhotonView**.

This method will be called in scripts that are assigned as Observed component of a **PhotonView**.

PhotonNetwork.SerializationRate affects how often this method is called.

PhotonNetwork.SendRate affects how often packages are sent by this client.

Implementing this method, you can customize which data a **PhotonView** regularly synchronizes. Your code defines what is being sent (content) and how your data is used by receiving clients.

Unlike other callbacks, *OnPhotonSerializeView only gets called when it is assigned to a **PhotonView** as PhotonView.observed script.*

To make use of this method, the **PhotonStream** is essential. It will be in "writing" mode" on the client that controls a PhotonView (PhotonStream.IsWriting == true) and in "reading mode" on the remote clients that just receive that the controlling client sends.

If you skip writing any value into the stream, PUN will skip the update. Used carefully, this can conserve bandwidth and messages (which have a limit per room/second).

Note that OnPhotonSerializeView is not called on remote clients when the sender does not send any update. This can't be used as "x-times per second Update()".

Implemented in **PhotonAnimatorView**, **CullingHandler**, **PhotonTransformViewClassic**, **PhotonTransformView**, **PhotonRigidbodyView**, **PhotonRigidbody2DView**, and **SmoothSyncMovement**.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonNetwork		

PhotonNetwork Class Reference

Public API

[Classes](#) | [Static Public Member Functions](#) | [Public Attributes](#) | [Static Public Attributes](#) | [Properties](#) | [List of all members](#)

The main class to use the **PhotonNetwork** plugin. This class is static.
[More...](#)

Static Public Member Functions

static bool **ConnectUsingSettings ()**
Connect to **Photon** as configured in the PhotonServerSettings file. [More...](#)

static bool **ConnectToMaster** (string masterServerAddress, int port, string appId)
Connect to a **Photon** Master Server by address, port, appId. [More...](#)

static bool **ConnectToBestCloudServer ()**
Connect to the **Photon** Cloud region with the lowest ping (on platforms that support Unity's Ping). [More...](#)

static bool **ConnectToRegion** (string region)
Connects to the **Photon** Cloud region of choice. [More...](#)

static void **Disconnect ()**
Makes this client disconnect from the photon server, a process that leaves any room and calls OnDisconnected on completion. [More...](#)

static bool **Reconnect ()**
Can be used to reconnect to the master server after a disconnect. [More...](#)

static void **NetworkStatisticsReset ()**
Resets the traffic stats and re-enables them. [More...](#)

static string **NetworkStatisticsToString ()**
Only available when
NetworkStatisticsEnabled was used
to gather some stats. [More...](#)

static int **GetPing ()**
The current roundtrip time to the
photon server. [More...](#)

static void **FetchServerTimestamp ()**
Refreshes the server timestamp
(async operation, takes a roundtrip).
[More...](#)

static void **SendAllOutgoingCommands ()**
Can be used to immediately send the
RPCs and Instantiates just called, so
they are on their way to the other
players. [More...](#)

static bool **CloseConnection (Player
kickPlayer)**
Request a client to disconnect
(KICK). Only the master client can do
this [More...](#)

static bool **SetMasterClient (Player
masterClientPlayer)**
Asks the server to assign another
player as Master Client of your
current room. [More...](#)

static bool **JoinRandomRoom ()**
Joins a random room that matches
the filter. Will callback:
OnJoinedRoom or
OnJoinRandomFailed. [More...](#)

static bool **JoinRandomRoom** (Hashtable expectedCustomRoomProperties, byte expectedMaxPlayers)
Joins a random room that matches the filter. Will callback:
OnJoinedRoom or
OnJoinRandomFailed. [More...](#)

static bool **JoinRandomRoom** (Hashtable expectedCustomRoomProperties, byte expectedMaxPlayers, **MatchmakingMode** matchingType, **TypedLobby** typedLobby, string sqlLobbyFilter, string[] expectedUsers=null)
Joins a random room that matches the filter. Will callback:
OnJoinedRoom or
OnJoinRandomFailed. [More...](#)

static bool **CreateRoom** (string roomName, **RoomOptions** roomOptions=null, **TypedLobby** typedLobby=null, string[] expectedUsers=null)
Creates a new room. Will callback:
OnCreatedRoom and
OnJoinedRoom or
OnCreateRoomFailed. [More...](#)

static bool **JoinOrCreateRoom** (string roomName, **RoomOptions** roomOptions, **TypedLobby** typedLobby, string[] expectedUsers=null)
Joins a specific room by name and creates it on demand. Will callback:
OnJoinedRoom or
OnJoinRoomFailed. [More...](#)

static bool **JoinRoom** (string roomName, string[] expectedUsers=null)
Joins a room by name. Will callback: OnJoinedRoom or OnJoinRoomFailed. [More...](#)

static bool **RejoinRoom** (string roomName)
Rejoins a room by roomName (using the userID internally to return). Will callback: OnJoinedRoom or OnJoinRoomFailed. [More...](#)

static bool **ReconnectAndRejoin** ()
When the client lost connection during gameplay, this method attempts to reconnect and rejoin the room. [More...](#)

static bool **LeaveRoom** (bool becomeInactive=true)
Leave the current room and return to the Master Server where you can join or create rooms (see remarks). [More...](#)

static bool **JoinLobby** ()
On MasterServer this joins the default lobby which list rooms currently in use. [More...](#)

static bool **JoinLobby (TypedLobby typedLobby)**
On a Master Server you can join a lobby to get lists of available rooms. [More...](#)

static bool **LeaveLobby** ()
Leave a lobby to stop getting

updates about available rooms.
[More...](#)

static bool **FindFriends** (string[] friendsToFind)
Requests the rooms and online status for a list of friends and saves the result in PhotonNetwork.Friends.
[More...](#)

static bool **GetCustomRoomList** (TypedLobby typedLobby, string sqlLobbyFilter)
Fetches a custom list of games from the server, matching a SQL-like "where" clause, then triggers OnRoomListUpdate callback. [More...](#)

static void **SetPlayerCustomProperties** (Hashtable customProperties)
Sets this (local) player's properties and synchronizes them to the other players (don't modify them directly).
[More...](#)

static void **RemovePlayerCustomProperties** (string[] customPropertiesToDelete)
Locally removes Custom Properties of "this" player. Important: This does not synchronize the change! Useful when you switch rooms. [More...](#)

static bool **RaiseEvent** (byte eventCode, object eventContent, **RaiseEventOptions** raiseEventOptions, **SendOptions** sendOptions)
Sends fully customizable events in a room. Events consist of at least an EventCode (0..199) and can have content. [More...](#)

static bool **AllocateViewID (PhotonView view)**
Allocates a viewID for the current/local player. [More...](#)

static bool **AllocateSceneViewID (PhotonView view)**
Enables the Master Client to allocate a viewID that is valid for scene objects. [More...](#)

static GameObject **Instantiate** (string prefabName, Vector3 position, Quaternion rotation, byte group=0, object[] data=null)

static GameObject **InstantiateSceneObject** (string prefabName, Vector3 position, Quaternion rotation, byte group=0, object[] data=null)

static void **Destroy (PhotonView targetView)**
Network-Destroy the GameObject associated with the **PhotonView**, unless the **PhotonView** is static or not under this client's control. [More...](#)

static void **Destroy (GameObject targetGo)**
Network-Destroy the GameObject, unless it is static or not under this client's control. [More...](#)

static void **DestroyPlayerObjects (Player targetPlayer)**
Network-Destroy all GameObjects, PhotonViews and their RPCs of targetPlayer. Can only be called on local player (for "self") or Master Client (for anyone). [More...](#)

static void **DestroyPlayerObjects** (int
targetPlayerId)
Network-Destroy all GameObjects,
PhotonViews and their RPCs of this
player (by ID). Can only be called on
local player (for "self") or Master
Client (for anyone). [More...](#)

static void **DestroyAll** ()
Network-Destroy all GameObjects,
PhotonViews and their RPCs in the
room. Removes anything buffered
from the server. Can only be called
by Master Client (for anyone). [More...](#)

static void **RemoveRPCs (Player targetPlayer)**
Remove all buffered RPCs from
server that were sent by targetPlayer.
Can only be called on local player
(for "self") or Master Client (for
anyone). [More...](#)

static void **RemoveRPCs (PhotonView
targetPhotonView)**
Remove all buffered RPCs from
server that were sent via
targetPhotonView. The Master Client
and the owner of the
targetPhotonView may call this.
[More...](#)

static HashSet< GameObject > **FindGameObjectsWithComponent**
(Type type)
Finds the GameObjects with
Components of a specific type (using
FindObjectOfType). [More...](#)

static void **SetInterestGroups** (byte group, bool enabled)
Enable/disable receiving events from a given Interest Group. [More...](#)

static void **LoadLevel** (int levelNumber)
This method wraps loading a level asynchronously and pausing network messages during the process. [More...](#)

static void **LoadLevel** (string levelName)
This method wraps loading a level asynchronously and pausing network messages during the process. [More...](#)

static bool **WebRpc** (string name, object parameters, bool sendAuthCookie=false)
This operation makes **Photon** call your custom web-service by name (path) with the given parameters. [More...](#)

static void **AddCallbackTarget** (object target)
Registers an object for callbacks for the implemented callback-interfaces. [More...](#)

static void **RemoveCallbackTarget** (object target)
Removes the target object from callbacks for its implemented callback-interfaces. [More...](#)

static void **DestroyPlayerObjects** (int playerId, bool localOnly)

Destroys all Instantiates and RPCs locally and (if not localOnly) sends EvDestroy(player) and clears related events in the server buffer. [More...](#)

static void **DestroyAll** (bool localOnly)

static bool **LocalCleanPhotonView**
(PhotonView view)

static PhotonView **GetPhotonView** (int viewID)

static void **RegisterPhotonView** (PhotonView netView)

static void **OpCleanActorRpcBuffer** (int actorNumber)
Removes the RPCs of someone else (to be used as master). This won't clean any local caches. It just tells the server to forget a player's RPCs and instantiates. [More...](#)

static void **OpRemoveCompleteCacheOfPlayer** (int actorNumber)
Instead removing RPCs or Instantiates, this removed everything cached by the actor. [More...](#)

static void **OpRemoveCompleteCache** ()

static void **CleanRpcBufferIfMine** (PhotonView view)

static void **OpCleanRpcBuffer** (PhotonView view)
Cleans server RPCs for **PhotonView** (without any further checks). [More...](#)

static void **RemoveRPCsInGroup** (int group)
Remove all buffered RPCs from server that were sent in the targetGroup, if this is the Master Client or if this controls the individual **PhotonView**. [More...](#)

static void **SetLevelPrefix** (byte prefix)
Sets level prefix for PhotonViews instantiated later on. Don't set it if you need only one! [More...](#)

static void **SetInterestGroups** (byte[] disableGroups, byte[] enableGroups)
Enable/disable receiving on given Interest Groups (applied to PhotonViews). [More...](#)

static void **SetSendingEnabled** (byte group, bool enabled)
Enable/disable sending on given group (applied to PhotonViews) [More...](#)

static void **SetSendingEnabled** (byte[] disableGroups, byte[] enableGroups)
Enable/disable sending on given groups (applied to PhotonViews) [More...](#)

Public Attributes

const string **PunVersion** = "2.12"
Version number of PUN. Used in the AppVersion, which separates your playerbase in matchmaking. [More...](#)

const int **SyncViewId** = 0

const int **SyncCompressed** = 1

const int **SyncNullValues** = 2

const int **SyncFirstValue** = 3

Static Public Attributes

static **LoadBalancingClient** **NetworkingClient**
The **LoadBalancingClient** is part of **Photon** and manages multiple servers and states for PUN. [More...](#)

static readonly int **MAX_VIEW_IDS** = 1000
The maximum number of assigned PhotonView IDs. See the **General Documentation** topic "Limitation of PhotonView IDs" for more information. [More...](#)

static **ServerSettings** **PhotonServerSettings** =
(**ServerSettings**)Resources.Load(PhotonTypeOf(**ServerSettings**))
Serialized server settings, written by the server. [ConnectUsingSettings](#). [More...](#)

static **ConnectMethod** **ConnectMethod** = **ConnectMethod**.NotC
Tracks, which Connect method was called.

static **PunLogLevel** **LogLevel** = **PunLogLevel**.ErrorsOnly
Controls how verbose PUN is. [More...](#)

static float **PrecisionForVectorSynchronization** = 0.0001f
The minimum difference that a Vector2 or Vector3 (or rotation) needs to change before we send it to OnSerialize/ObservingComponent. [More...](#)

static float **PrecisionForQuaternionSynchronization** = 0.0001f
The minimum angle that a rotation needs to change before we send it to **PhotonView**'s OnSerialize/ObservingComponent. [More...](#)

static float **PrecisionForFloatSynchronization** = 0.0001f
The minimum difference between floats before we send it to **PhotonView**'s OnSerialize/ObservingComponent. [More...](#)

static bool **UseRpcMonoBehaviourCache**

While enabled, the MonoBehaviours on w
avoiding costly GetComponent<MonoBe

static int **ObjectsInOneUpdate** = 10

Defines how many OnPhotonSerialize()-c
message. [More...](#)

Properties

static string **GameVersion** [get, set]
Version number of your game. Setting this updates the AppVersion, which separates your playerbase in matchmaking. [More...](#)

static string **AppVersion** [get]
Sent to **Photon** Server to specify the "Virtual AppId". [More...](#)

static string **ServerAddress** [get]
Currently used server address (no matter if master or game server). [More...](#)

static string **CloudRegion** [get]
Currently used Cloud Region (if any). As long as the client is not on a Master Server or Game Server, the region is not yet defined. [More...](#)

static string **BestRegionSummaryInPreferences** [get, set]
Used to store and access the "Best Region Summary" in the Player Preferences. [More...](#)

static bool **IsConnected** [get]
False until you connected to **Photon** initially. True in offline mode, while connected to any server and even while switching servers. [More...](#)

static bool **IsConnectedAndReady** [get]
A refined version of connected which is true only if your connection to the server

ready to accept operations like join, leave etc. [More...](#)

static **ClientState** **NetworkClientState** [get]

Directly provides the network-level client state, unless in OfflineMode. [More...](#)

static **ServerConnection** **Server** [get]

The server (type) this client is currently connected or connecting to. [More...](#)

static **AuthenticationValues** **AuthValues** [get, set]

A user's authentication values used during connect. [More...](#)

static **TypedLobby** **CurrentLobby** [get]

The lobby that will be used when PUN joins a lobby or creates a game. This is defined when joining a lobby or creating rooms [More...](#)

static **Room** **CurrentRoom** [get]

Get the room we're currently in (also when in OfflineMode). Null if we aren't in any room. [More...](#)

static **Player** **LocalPlayer** [get]

This client's Player instance is always available, unless the app shuts down. [More...](#)

static string **NickName** [get, set]

Set to synchronize the player's nickname with everyone in the room(s) you enter. This sets PhotonNetwork.player.NickName. [More...](#)

static **Player[]** **PlayerList** [get]

A sorted copy of the players-list of the current room. This is using Linq, so better cache this value. Update when players join / leave. [More...](#)

static **Player[]** **PlayerListOthers** [get]

A sorted copy of the players-list of the current room, excluding this client. This is using Linq, so better cache this value. Update when players join / leave. [More...](#)

static bool **OfflineMode** [get, set]

Offline mode can be set to re-use your multiplayer code in singleplayer game modes. When this is on **PhotonNetwork** will not create any connections and there is near to no overhead. Mostly useful for reusing RPC's and PhotonNetwork.Instantiate [More...](#)

static bool **AutomaticallySyncScene** [get, set]

Defines if all clients in a room should automatically load the same level as the Master Client. [More...](#)

static bool **EnableLobbyStatistics** [get]

If enabled, the client will get a list of available lobbies from the Master Server. [More...](#)

static bool **InLobby** [get]

True while this client is in a lobby. [More...](#)

static int **SendRate** [get, set]

Defines how many times per second

PhotonNetwork should send a package. If you change this, do not forget to also change 'SerializationRate'. [More...](#)

static int **SerializationRate** [get, set]
Defines how many times per second OnPhotonSerialize should be called on PhotonViews. [More...](#)

static bool **IsMessageQueueRunning** [get, set]
Can be used to pause dispatching of incoming events (RPCs, Instantiates and anything else incoming). [More...](#)

static double **Time** [get]
Photon network time, synched with the server. [More...](#)

static int **ServerTimestamp** [get]
The current server's millisecond timestamp. [More...](#)

static float **KeepAliveInBackground** [get, set]
Defines how many seconds PUN keeps the connection after Unity's OnApplicationPause(true) call. Default: 6 seconds. [More...](#)

static float **BackgroundTimeout** [get, set]

static bool **IsMasterClient** [get]
Are we the master client? [More...](#)

static **Player** **MasterClient** [get]
The Master Client of the current room or null (outside of rooms). [More...](#)

static bool **InRoom** [get]
Is true while being in a room
(NetworkClientState ==
ClientState.Joined). [More...](#)

static int **CountOfPlayersOnMaster** [get]
The count of players currently looking for
room (available on MasterServer in 5sec
intervals). [More...](#)

static int **CountOfPlayersInRooms** [get]
Count of users currently playing your app
in some room (sent every 5sec by Maste
Server). Use
PhotonNetwork.PlayerList.Length or
PhotonNetwork.CurrentRoom.PlayerCou
to get the count of players in the room
you're in! [More...](#)

static int **CountOfPlayers** [get]
The count of players currently using this
application (available on MasterServer in
5sec intervals). [More...](#)

static int **CountOfRooms** [get]
The count of rooms currently in use
(available on MasterServer in 5sec
intervals). [More...](#)

static bool **NetworkStatisticsEnabled** [get, set]
Enables or disables the collection of
statistics about this client's traffic. [More...](#)

static int **ResentReliableCommands** [get]
Count of commands that got repeated
(due to local repeat-timing before an ACK

was received). [More...](#)

static bool **CrcCheckEnabled** [get, set]
Crc checks can be useful to detect and avoid issues with broken datagrams. Can be enabled while not connected. [More...](#)

static int **PacketLossByCrcCheck** [get]
If CrcCheckEnabled, this counts the incoming packages that don't have a valid CRC checksum and got rejected. [More...](#)

static int **MaxResendsBeforeDisconnect** [get, set]
Defines the number of times a reliable message can be resent before not getting an ACK for it will trigger a disconnect. Default: 5. [More...](#)

static int **QuickResends** [get, set]
In case of network loss, reliable message can be repeated quickly up to 3 times. [More...](#)

static bool **UseAlternativeUdpPorts** [get, set]
Switch to alternative ports for a UDP connection to the Public Cloud. [More...](#)

static **PhotonView[]** **PhotonViews** [get]
Gets the photon views. [More...](#)

static **IPunPrefabPool** **PrefabPool** [get, set]
An Object Pool can be used to keep and reuse instantiated object instances. Replaces Unity's default Instantiate and Destroy methods. [More...](#)

static float **LevelLoadingProgress** [get]
Represents the scene loading progress
when using **LoadLevel()**. [More...](#)

Detailed Description

The main class to use the **PhotonNetwork** plugin. This class is static.

Member Function Documentation

static void AddCallbackTarget (object **target)**

static

Registers an object for callbacks for the implemented callback-interfaces.

The covered callback interfaces are: IConnectionCallbacks, IMatchmakingCallbacks, ILobbyCallbacks, IInRoomCallbacks, IOnEventCallback and IWebRpcCallback.

See: [.Net Callbacks](#)

Parameters

target The object that registers to get callbacks from PUN's LoadBalancingClient.

static bool AllocateSceneViewID (PhotonView **view)**

static

Enables the Master Client to allocate a viewID that is valid for scene objects.

Returns

static bool AllocateViewID (PhotonView **view)**

static

Allocates a viewID for the current/local player.

Returns

static bool CloseConnection (Player **kickPlayer)**

static

Request a client to disconnect (KICK). Only the master client can do this

Only the target player gets this event. That player will disconnect automatically, which is what the others will notice, too.

Parameters

kickPlayer The Player to kick.

static bool ConnectToBestCloudServer ()

static

Connect to the **Photon** Cloud region with the lowest ping (on platforms that support Unity's Ping).

Will save the result of pinging all cloud servers in PlayerPrefs. Calling this the first time can take +-2 seconds. The ping result can be overridden via PhotonNetwork.OverrideBestCloudServer(..) This call can take up to 2 seconds if it is the first time you are using this, all cloud servers will be pinged to check for the best region.

The PUN Setup Wizard stores your appId in a settings file and applies a server address/port. To connect to the **Photon** Cloud, a valid AppId must be in the settings file (shown in the **Photon** Cloud Dashboard). <https://dashboard.photonengine.com>

Connecting to the **Photon** Cloud might fail due to:

- Invalid AppId
- Network issues
- Invalid region
- Subscription CCU limit reached
- etc.

In general check out the DisconnectCause from the **IConnectionCallbacks.OnDisconnected** callback.

Returns

If this client is going to connect to cloud server based on ping.

Even if true, this does not guarantee a connection but the attempt is being made.

```
static bool
ConnectToMaster          ( string masterServerAddress,
                           int      port,
                           string   appId
                           )
```

static

Connect to a **Photon** Master Server by address, port, appId.

To connect to the **Photon** Cloud, a valid AppId must be in the settings file (shown in the **Photon** Cloud Dashboard).

<https://dashboard.photonengine.com>

Connecting to the **Photon** Cloud might fail due to:

- Invalid AppId
- Network issues
- Invalid region
- Subscription CCU limit reached
- etc.

In general check out the DisconnectCause from the **ILogicCallbacks.OnDisconnected** callback.

Parameters

masterServerAddress	The server's address (either your own or Photon Cloud address).
port	The server's port to connect to.
appId	Your application ID (Photon Cloud provides you with a GUID for your game).

```
static bool ConnectToRegion ( string region )
```

static

Connects to the **Photon** Cloud region of choice.

static bool ConnectUsingSettings ()

static

Connect to **Photon** as configured in the PhotonServerSettings file.

Implement IConnectionCallbacks, to make your game logic aware of state changes. Especially, IConnectionCallbacks.ConnectedToMasterServer is useful to react when the client can do matchmaking.

This method will disable OfflineMode (which won't destroy any instantiated GOs) and it will set IsMessageQueueRunning to true.

Your **Photon** configuration is created by the PUN Wizard and contains the AppId, region for **Photon** Cloud games, the server address among other things.

To ignore the settings file, set the relevant values and connect by calling ConnectToMaster, ConnectToRegion.

To connect to the **Photon** Cloud, a valid AppId must be in the settings file (shown in the **Photon** Cloud Dashboard).

<https://dashboard.photonengine.com>

Connecting to the **Photon** Cloud might fail due to:

- Invalid AppId
- Network issues
- Invalid region
- Subscription CCU limit reached
- etc.

In general check out the DisconnectCause from the **IConnectionCallbacks.OnDisconnected** callback.

static bool

```

CreateRoom      ( string      roomName,
                  RoomOptions roomOptions = null,
                  TypedLobby  typedLobby = null,
                  string[]    expectedUsers = null
                  )
static

```

Creates a new room. Will callback: OnCreatedRoom and OnJoinedRoom or OnCreateRoomFailed.

When successful, this calls the callbacks OnCreatedRoom and OnJoinedRoom (the latter, cause you join as first player). In all error cases, OnCreateRoomFailed gets called.

Creating a room will fail if the room name is already in use or when the RoomOptions clashing with one another. Check the EnterRoomParams reference for the various room creation options.

If you don't want to create a unique room-name, pass null or "" as name and the server will assign a roomName (a GUID as string).

This method can only be called while the client is connected to a Master Server so you should implement the callback OnConnectedToMaster. Check the return value to make sure the operation will be called on the server. Note: There will be no callbacks if this method returned false.

More about PUN matchmaking: <https://doc.photonengine.com/en-us/pun/v2/lobby-and-matchmaking/matchmaking-and-lobby>

Parameters

- roomName** Unique name of the room to create. Pass null or "" to make the server generate a name.
- roomOptions** Common options for the room like MaxPlayers, initial custom room properties and similar. See RoomOptions type..
- typedLobby** If null, the room is automatically created in the currently used lobby (which is "default" when you didn't join one explicitly).
- expectedUsers** Optional list of users (by UserId) who are

expected to join this game and who you want to block a slot for.

Returns

If the operation got queued and will be sent.

static void Destroy (PhotonView **targetView)**

static

Network-Destroy the GameObject associated with the **PhotonView**, unless the **PhotonView** is static or not under this client's control.

Destroying a networked GameObject while in a Room includes:

- Removal of the Instantiate call from the server's room buffer.
- Removing RPCs buffered for PhotonViews that got created indirectly with the PhotonNetwork.Instantiate call.
- Sending a message to other clients to remove the GameObject also (affected by network lag).

Usually, when you leave a room, the GOs get destroyed automatically. If you have to destroy a GO while not in a room, the Destroy is only done locally.

Destroying networked objects works only if they got created with PhotonNetwork.Instantiate(). Objects loaded with a scene are ignored, no matter if they have **PhotonView** components.

The GameObject must be under this client's control:

- Instantiated and owned by this client.
- Instantiated objects of players who left the room are controlled by the Master Client.
- Scene-owned game objects are controlled by the Master Client.
- GameObject can be destroyed while client is not in a room.

Returns

Nothing. Check error debug log for any issues.

static void Destroy (GameObject **targetGo)**

static

Network-Destroy the GameObject, unless it is static or not under this client's control.

Destroying a networked GameObject includes:

- Removal of the Instantiate call from the server's room buffer.
- Removing RPCs buffered for PhotonViews that got created indirectly with the PhotonNetwork.Instantiate call.
- Sending a message to other clients to remove the GameObject also (affected by network lag).

Usually, when you leave a room, the GOs get destroyed automatically. If you have to destroy a GO while not in a room, the Destroy is only done locally.

Destroying networked objects works only if they got created with PhotonNetwork.Instantiate(). Objects loaded with a scene are ignored, no matter if they have **PhotonView** components.

The GameObject must be under this client's control:

- Instantiated and owned by this client.
- Instantiated objects of players who left the room are controlled by the Master Client.
- Scene-owned game objects are controlled by the Master Client.
- GameObject can be destroyed while client is not in a room.

Returns

Nothing. Check error debug log for any issues.

static void DestroyAll ()

static

Network-Destroy all GameObjects, PhotonViews and their RPCs in the room. Removes anything buffered from the server. Can only be called by Master Client (for anyone).

Can only be called by Master Client (for anyone). Unlike the Destroy

methods, this will remove anything from the server's room buffer. If your game buffers anything beyond Instantiate and RPC calls, that will be cleaned as well from server.

Destroying all includes:

- Remove anything from the server's room buffer (Instantiate, RPCs, anything buffered).
- Sending a message to other clients to destroy everything locally, too (affected by network lag).

Destroying networked objects works only if they got created with `PhotonNetwork.Instantiate()`. Objects loaded with a scene are ignored, no matter if they have **PhotonView** components.

Returns

Nothing. Check error debug log for any issues.

```
static void DestroyPlayerObjects ( int   playerId,  
                                  bool  localOnly  
                                )
```

static

Destroys all Instantiates and RPCs locally and (if not `localOnly`) sends `EvDestroy(player)` and clears related events in the server buffer.

```
static void DestroyPlayerObjects ( Player targetPlayer )
```

static

Network-Destroy all GameObjects, PhotonViews and their RPCs of `targetPlayer`. Can only be called on local player (for "self") or Master Client (for anyone).

Destroying a networked GameObject includes:

- Removal of the Instantiate call from the server's room buffer.
- Removing RPCs buffered for PhotonViews that got created indirectly with the `PhotonNetwork.Instantiate` call.

- Sending a message to other clients to remove the GameObject also (affected by network lag).

Destroying networked objects works only if they got created with `PhotonNetwork.Instantiate()`. Objects loaded with a scene are ignored, no matter if they have **PhotonView** components.

Returns

Nothing. Check error debug log for any issues.

static void DestroyPlayerObjects (int **targetPlayerId)**

static

Network-Destroy all GameObjects, PhotonViews and their RPCs of this player (by ID). Can only be called on local player (for "self") or Master Client (for anyone).

Destroying a networked GameObject includes:

- Removal of the Instantiate call from the server's room buffer.
- Removing RPCs buffered for PhotonViews that got created indirectly with the `PhotonNetwork.Instantiate` call.
- Sending a message to other clients to remove the GameObject also (affected by network lag).

Destroying networked objects works only if they got created with `PhotonNetwork.Instantiate()`. Objects loaded with a scene are ignored, no matter if they have **PhotonView** components.

Returns

Nothing. Check error debug log for any issues.

static void Disconnect ()

static

Makes this client disconnect from the photon server, a process that leaves any room and calls `OnDisconnected` on completion.

When you disconnect, the client will send a "disconnecting" message to the server. This speeds up leave/disconnect messages

for players in the same room as you (otherwise the server would timeout this client's connection). When used in OfflineMode, the state-change and event-call OnDisconnected are immediate. Offline mode is set to false as well. Once disconnected, the client can connect again. Use ConnectUsingSettings.

static void FetchServerTimestamp ()

static

Refreshes the server timestamp (async operation, takes a roundtrip).

Can be useful if a bad connection made the timestamp unusable or imprecise.

static bool FindFriends (string[] friendsToFind)

static

Requests the rooms and online status for a list of friends and saves the result in PhotonNetwork.Friends.

Works only on Master Server to find the rooms played by a selected list of users.

The result will be stored in PhotonNetwork.Friends when available. That list is initialized on first use of OpFindFriends (before that, it is null). To refresh the list, call FindFriends again (in 5 seconds or 10 or 20).

Users identify themselves by setting a unique userId in the **PhotonNetwork.AuthValues**. See remarks of AuthenticationValues for info about how this is set and used.

The list of friends must be fetched from some other source (not provided by **Photon**).

Internal: The server response includes 2 arrays of info (each index matching a friend from the request):

ParameterCode.FindFriendsResponseOnlineList = bool[] of online states **ParameterCode.FindFriendsResponseRoomIdList** =

string[] of room names (empty string if not in a room)

Parameters

friendsToFind Array of friend (make sure to use unique NickName or AuthValues).

Returns

If the operation could be sent (requires connection, only one request is allowed at any time). Always false in offline mode.

static HashSet<GameObject>

FindGameObjectsWithComponent

(Type **type**)

static

Finds the GameObjects with Components of a specific type (using FindObjectsOfType).

Parameters

type Type must be a Component

Returns

HashSet with GameObjects that have a specific type of Component.

static bool

GetCustomRoomList

(TypedLobby **typedLobby**,
string **sqlLobbyFilter**
)

static

Fetches a custom list of games from the server, matching a SQL-like "where" clause, then triggers OnRoomListUpdate callback.

Operation is only available for lobbies of type SqlLobby. This is an async request.

Note: You don't have to join a lobby to query it. Rooms need to be "attached" to a lobby, which can be done via the typedLobby parameter in CreateRoom, JoinOrCreateRoom, etc..

When done, OnRoomListUpdate gets called.

https://doc.photonengine.com/en-us/pun/v2/lobby-and-matchmaking/matchmaking-and-lobby/#sql_lobby_type

Parameters

typedLobby The lobby to query. Has to be of type SqlLobby.
sqlLobbyFilter The sql query statement.

Returns

If the operation could be sent (has to be connected).

static int GetPing ()

static

The current roundtrip time to the photon server.

Returns

Roundtrip time (to server and back).

static bool JoinLobby ()

static

On MasterServer this joins the default lobby which list rooms currently in use.

The room list is sent and refreshed by the server using **ILobbyCallbacks.OnRoomListUpdate**.

Per room you should check if it's full or not before joining. **Photon** also lists rooms that are full, unless you close and hide them (room.open = false and room.visible = false).

In best case, you make your clients join random games, as described here: <https://doc.photonengine.com/en-us/pun/v2/lobby-and-matchmaking/matchmaking-and-lobby>

You can show your current players and room count without joining a lobby (but you must be on the master server). Use: CountOfPlayers,

CountOfPlayersOnMaster, CountOfPlayersInRooms and CountOfRooms.

You can use more than one lobby to keep the room lists shorter. See **JoinLobby(TypedLobby lobby)**. When creating new rooms, they will be "attached" to the currently used lobby or the default lobby.

You can use JoinRandomRoom without being in a lobby!

static bool JoinLobby (TypedLobby typedLobby)

static

On a Master Server you can join a lobby to get lists of available rooms.

The room list is sent and refreshed by the server using **ILobbyCallbacks.OnRoomListUpdate**.

Any client can "make up" any lobby on the fly. Splitting rooms into multiple lobbies will keep each list shorter. However, having too many lists might ruin the matchmaking experience.

In best case, you create a limited number of lobbies. For example, create a lobby per game-mode: "koth" for king of the hill and "ffa" for free for all, etc.

There is no listing of lobbies at the moment.

Sql-typed lobbies offer a different filtering model for random matchmaking. This might be more suited for skillbased-games. However, you will also need to follow the conventions for naming filterable properties in sql-lobbies! Both is explained in the matchmaking doc linked below.

In best case, you make your clients join random games, as described here: <https://doc.photonengine.com/en-us/realtime/current/reference/matchmaking-and-lobby>

Per room you should check if it's full or not before joining. **Photon** does list rooms that are full, unless you close and hide them (room.open = false and room.visible = false).

You can show your games current players and room count without joining a lobby (but you must be on the master server). Use: CountOfPlayers, CountOfPlayersOnMaster, CountOfPlayersInRooms and CountOfRooms.

When creating new rooms, they will be "attached" to the currently used lobby or the default lobby.

You can use JoinRandomRoom without being in a lobby!

Parameters

typedLobby A typed lobby to join (must have name and type).

```
static bool
JoinOrCreateRoom ( string      roomName,
                   RoomOptions roomOptions,
                   TypedLobby  typedLobby,
                   string[]    expectedUsers = null
                 )
```

static

Joins a specific room by name and creates it on demand. Will callback: OnJoinedRoom or OnJoinRoomFailed.

Useful when players make up a room name to meet in: All involved clients call the same method and whoever is first, also creates the room.

When successful, the client will enter the specified room. The client which creates the room, will callback both OnCreatedRoom and OnJoinedRoom. Clients that join an existing room will only callback OnJoinedRoom. In all error cases, OnJoinRoomFailed gets called.

Joining a room will fail, if the room is full, closed or when the user already is present in the room (checked by userId).

To return to a room, use OpRejoinRoom.

This method can only be called while the client is connected to a

Master Server so you should implement the callback `OnConnectedToMaster`. Check the return value to make sure the operation will be called on the server. Note: There will be no callbacks if this method returned false.

If you set room properties in `roomOptions`, they get ignored when the room is existing already. This avoids changing the room properties by late joining players.

You can define an array of `expectedUsers`, to block player slots in the room for these users. The corresponding feature in **Photon** is called "Slot Reservation" and can be found in the doc pages.

More about PUN matchmaking: <https://doc.photonengine.com/en-us/pun/v2/lobby-and-matchmaking/matchmaking-and-lobby>

Parameters

roomName	Name of the room to join. Must be non null.
roomOptions	Options for the room, in case it does not exist yet. Else these values are ignored.
typedLobby	Lobby you want a new room to be listed in. Ignored if the room was existing and got joined.
expectedUsers	Optional list of users (by <code>UserId</code>) who are expected to join this game and who you want to block a slot for.

Returns

If the operation got queued and will be sent.

static bool JoinRandomRoom ()

static

Joins a random room that matches the filter. Will callback: `OnJoinedRoom` or `OnJoinRandomFailed`.

Used for random matchmaking. You can join any room or one with specific properties defined in `opJoinRandomRoomParams`.

This operation fails if no rooms are fitting or available (all full, closed,

in another lobby or not visible). It may also fail when actually joining the room which was found. Rooms may close, become full or empty anytime.

This method can only be called while the client is connected to a Master Server so you should implement the callback `OnConnectedToMaster`. Check the return value to make sure the operation will be called on the server. Note: There will be no callbacks if this method returned false.

More about PUN matchmaking: <https://doc.photonengine.com/en-us/pun/v2/lobby-and-matchmaking/matchmaking-and-lobby>

```
static bool
JoinRandomRoom ( Hashtable expectedCustomRoomProperties,
                 byte        expectedMaxPlayers
                 )
```

Joins a random room that matches the filter. Will callback: `OnJoinedRoom` or `OnJoinRandomFailed`.

Used for random matchmaking. You can join any room or one with specific properties defined in `opJoinRandomRoomParams`.

This operation fails if no rooms are fitting or available (all full, closed, in another lobby or not visible). It may also fail when actually joining the room which was found. Rooms may close, become full or empty anytime.

This method can only be called while the client is connected to a Master Server so you should implement the callback `OnConnectedToMaster`. Check the return value to make sure the operation will be called on the server. Note: There will be no callbacks if this method returned false.

More about PUN matchmaking: <https://doc.photonengine.com/en-us/pun/v2/lobby-and-matchmaking/matchmaking-and-lobby>

Parameters

expectedCustomRoomProperties Filters for rooms that match the custom properties (string keys)

expectedMaxPlayers

and values). To ignore, pass r
Filters for a particular maxplay
setting. Use 0 to accept any
maxPlayer value.

Returns

If the operation got queued and will be sent.

static bool

```
JoinRandomRoom ( Hashtable  
                 byte  
                 MatchmakingMode  
                 TypedLobby  
                 string  
                 string[]  
                 )  
                 expectedCustomRoomProp  
                 expectedMaxPlayers,  
                 matchingType,  
                 typedLobby,  
                 sqlLobbyFilter,  
                 expectedUsers = null
```

Joins a random room that matches the filter. Will callback: OnJoinedRo
OnJoinRandomFailed.

Used for random matchmaking. You can join any room or one with spec
defined in opJoinRandomRoomParams.

This operation fails if no rooms are fitting or available (all full, closed, in
lobby or not visible). It may also fail when actually joining the room whic
Rooms may close, become full or empty anytime.

This method can only be called while the client is connected to a Maste
you should implement the callback OnConnectedToMaster. Check the r
make sure the operation will be called on the server. Note: There will be
if this method returned false.

More about PUN matchmaking: <https://doc.photonengine.com/en-us/pu-and-matchmaking/matchmaking-and-lobby>

Parameters

expectedCustomRoomProperties Filters for rooms that match th
properties (string keys and va

expectedMaxPlayers

ignore, pass null.

matchingType

Filters for a particular maxplay
Use 0 to accept any maxPlaye
Selects one of the available m
algorithms. See Matchmaking
for options.

typedLobby

The lobby in which you want t
room. Pass null, to use the de
This does not join that lobby a
sets the lobby property.

sqlLobbyFilter
expectedUsers

A filter-string for SQL-typed lo
Optional list of users (by User
expected to join this game and
want to block a slot for.

Returns

If the operation got queued and will be sent.

```
static bool JoinRoom ( string  roomName,  
                      string[] expectedUsers = null  
                      )
```

static

Joins a room by name. Will callback: OnJoinedRoom or OnJoinRoomFailed.

Useful when using lobbies or when players follow friends or invite each other.

When successful, the client will enter the specified room and callback via OnJoinedRoom. In all error cases, OnJoinRoomFailed gets called.

Joining a room will fail if the room is full, closed, not existing or when the user already is present in the room (checked by userId).

To return to a room, use OpRejoinRoom. When players invite each other and it's unclear who's first to respond, use OpJoinOrCreateRoom instead.

This method can only be called while the client is connected to a Master Server so you should implement the callback `OnConnectedToMaster`. Check the return value to make sure the operation will be called on the server. Note: There will be no callbacks if this method returned false.

More about PUN matchmaking: <https://doc.photonengine.com/en-us/pun/v2/lobby-and-matchmaking/matchmaking-and-lobby>

`OnJoinRoomFailed` `OnJoinedRoom`

Parameters

roomName Unique name of the room to join.
expectedUsers Optional list of users (by `UserId`) who are expected to join this game and who you want to block a slot for.

Returns

If the operation got queued and will be sent.

static bool LeaveLobby ()

static

Leave a lobby to stop getting updates about available rooms.

This does not reset `PhotonNetwork.lobby`! This allows you to join this particular lobby later easily.

The values `CountOfPlayers`, `CountOfPlayersOnMaster`, `CountOfPlayersInRooms` and `CountOfRooms` are received even without being in a lobby.

You can use `JoinRandomRoom` without being in a lobby.

static bool LeaveRoom (bool **becomeInactive = true)**

static

Leave the current room and return to the Master Server where you can join or create rooms (see remarks).

This will clean up all (network) GameObjects with a **PhotonView**, unless you changed `autoCleanUp` to false. Returns to the Master Server.

In `OfflineMode`, the local "fake" room gets cleaned up and `OnLeftRoom` gets called immediately.

In a room with `playerTTL < 0`, `LeaveRoom` just turns a client inactive. The player stays in the room's player list and can return later on. Setting `becomeInactive` to false deliberately, means to "abandon" the room, despite the `playerTTL` allowing you to come back.

In a room with `playerTTL == 0`, `become inactive` has no effect (clients are removed from the room right away).

Parameters

becomeInactive If this client becomes inactive in a room with `playerTTL < 0`. Defaults to true.

static void LoadLevel (int levelNumber)

static

This method wraps loading a level asynchronously and pausing network messages during the process.

While loading levels in a networked game, it makes sense to not dispatch messages received by other players. `LoadLevel` takes care of that by setting **`PhotonNetwork.IsMessageQueueRunning`** = false until the scene loaded.

To sync the loaded level in a room, set **`PhotonNetwork AutomaticallySyncScene`** to true. The Master Client of a room will then sync the loaded level with every other player in the room. Note that this works only for a single active scene and that reloading the scene is not supported. The Master Client will actually reload a scene but other clients won't.

You should make sure you don't fire RPCs before you load another scene (which doesn't contain the same GameObjects and PhotonViews).

LoadLevel uses SceneManager.LoadSceneAsync().

Check the progress of the LevelLoading using **PhotonNetwork.LevelLoadingProgress**.

Calling LoadLevel before the previous scene finished loading is not recommended. If AutomaticallySyncScene is enabled, PUN cancels the previous load (and prevent that from becoming the active scene). If AutomaticallySyncScene is off, the previous scene loading can finish. In both cases, a new scene is loaded locally.

Parameters

levelNumber Build-index number of the level to load. When using level numbers, make sure they are identical on all clients.

static void LoadLevel (string **levelName)**

static

This method wraps loading a level asynchronously and pausing network messages during the process.

While loading levels in a networked game, it makes sense to not dispatch messages received by other players. LoadLevel takes care of that by setting **PhotonNetwork.IsMessageQueueRunning** = false until the scene loaded.

To sync the loaded level in a room, set **PhotonNetwork.AutomaticallySyncScene** to true. The Master Client of a room will then sync the loaded level with every other player in the room. Note that this works only for a single active scene and that reloading the scene is not supported. The Master Client will actually reload a scene but other clients won't.

You should make sure you don't fire RPCs before you load another scene (which doesn't contain the same GameObjects and PhotonViews).

LoadLevel uses SceneManager.LoadSceneAsync().

Check the progress of the LevelLoading using **PhotonNetwork.LevelLoadingProgress**.

Calling LoadLevel before the previous scene finished loading is not recommended. If AutomaticallySyncScene is enabled, PUN cancels the previous load (and prevent that from becoming the active scene). If AutomaticallySyncScene is off, the previous scene loading can finish. In both cases, a new scene is loaded locally.

Parameters

levelName Name of the level to load. Make sure it's available to all clients in the same room.

static void NetworkStatisticsReset ()

static

Resets the traffic stats and re-enables them.

static string NetworkStatisticsToString ()

static

Only available when NetworkStatisticsEnabled was used to gather some stats.

Returns

A string with vital networking statistics.

static void OpCleanActorRpcBuffer (int **actorNumber)**

static

Removes the RPCs of someone else (to be used as master). This won't clean any local caches. It just tells the server to forget a player's RPCs and instantiates.

Parameters

actorNumber

```
static void OpCleanRpcBuffer ( PhotonView view )
```

static

Cleans server RPCs for **PhotonView** (without any further checks).

```
static void  
OpRemoveCompleteCacheOfPlayer    ( int actorNumber )
```

static

Instead removing RPCs or Instantiates, this removed everything cached by the actor.

Parameters

actorNumber

```
static bool  
RaiseEvent    ( byte  
               object  
               RaiseEventOptions  
               SendOptions  
               )  
               eventCode,  
               eventContent,  
               raiseEventOptions,  
               sendOptions
```

static

Sends fully customizable events in a room. Events consist of at least an EventCode (0..199) and can have content.

To receive events, implement **IOonEventCallback** in any class and register it via **PhotonNetwork.AddCallbackTarget**. See **IOonEventCallback.OnEvent**.

The eventContent is optional. If set, eventContent must be a "serializable type", something that the client can turn into a byte[] basically. Most basic types and arrays of them are supported, including Unity's Vector2, Vector3, Quaternion. Transforms are not supported.

You can turn a class into a "serializable type" by following the example in CustomTypes.cs.

The `RaiseEventOptions` have some (less intuitive) combination rules: If you set `targetActors` (an array of `Player.ID` values), the `receivers` parameter gets ignored. When using event caching, the `targetActors`, `receivers` and `interestGroup` can't be used. Buffered events go to all. When using `cachingOption removeFromRoomCache`, the `eventCode` and `content` are actually not sent but used as filter.

Parameters

eventCode	A byte identifying the type of event. You might want to use a code per action or to signal which content can be expected. Allowed: 0..199.
eventContent	Some serializable object like string, byte, integer, float (etc) and arrays of those. Hashtables with byte keys are good to send variable content.
raiseEventOptions	Allows more complex usage of events. If null, <code>RaiseEventOptions.Default</code> will be used (which is fine).
sendOptions	Send options for reliable, encryption etc..

Returns

False if event could not be sent.

static bool Reconnect ()

static

Can be used to reconnect to the master server after a disconnect.

After losing connection, you can use this to connect a client to the region Master Server again. Cache the room name you're in and use `RejoinRoom(roomname)` to return to a game. Common use case: Press the Lock Button on a iOS device and you get disconnected immediately.

static bool ReconnectAndRejoin ()

static

When the client lost connection during gameplay, this method attempts to reconnect and rejoin the room.

This method re-connects directly to the game server which was hosting the room PUN was in before. If the room was shut down in the meantime, PUN will call `OnJoinRoomFailed` and return this client to the Master Server.

Check the return value, if this client will attempt a reconnect and rejoin (if the conditions are met). If `ReconnectAndRejoin` returns false, you can still attempt a Reconnect and Rejoin.

Similar to **PhotonNetwork.RejoinRoom**, this requires you to use unique IDs per player (the UserID).

Rejoining room will not send any player properties. Instead client will receive up-to-date ones from server. If you want to set new player properties, do it once rejoined.

Returns

False, if there is no known room or game server to return to. Then, this client does not attempt the `ReconnectAndRejoin`.

static bool RejoinRoom (string **roomName)**

static

Rejoins a room by roomName (using the userID internally to return). Will callback: `OnJoinedRoom` or `OnJoinRoomFailed`.

After losing connection, you might be able to return to a room and continue playing, if the client is reconnecting fast enough. Use **Reconnect()** and this method. Cache the room name you're in and use `RejoinRoom(roomname)` to return to a game.

Note: To be able to Rejoin any room, you need to use UserIDs! You also need to set **RoomOptions.PlayerTtl**.

Important: Instantiate() and use of RPCs is not yet supported. The ownership rules of PhotonViews prevent a seamless return to a

game, if you use PhotonViews. Use Custom Properties and RaiseEvent with event caching instead.

Common use case: Press the Lock Button on a iOS device and you get disconnected immediately.

Rejoining room will not send any player properties. Instead client will receive up-to-date ones from server. If you want to set new player properties, do it once rejoined.

static void RemoveCallbackTarget (object **target)**

static

Removes the target object from callbacks for its implemented callback-interfaces.

The covered callback interfaces are: IConnectionCallbacks, IMatchmakingCallbacks, ILobbyCallbacks, IInRoomCallbacks, IOnEventCallback and IWebRpcCallback.

See: [.Net Callbacks](#)

Parameters

target The object that unregisters from getting callbacks.

static void

RemovePlayerCustomProperties (string[] **customPropertiesToDelete)**

Locally removes Custom Properties of "this" player. Important: This does not synchronize the change! Useful when you switch rooms.

Use this method with care. It can create inconsistencies of state between clients. This only changes the player.customProperties locally. This can be used to remove your Custom Properties between games (let's say they store which turn you made, kills, etc).

SetPlayerCustomProperties() syncs and can be used to set values to a room. That can be considered "removed" while in a room.

If `customPropertiesToDelete` is null or has 0 entries, all Custom Property deleted (replaced with a new Hashtable). If you specify keys to remove, be removed from the Hashtable but other keys are unaffected.

Parameters

`customPropertiesToDelete` List of Custom Property keys to remove remarks.

static void RemoveRPCs (Player `targetPlayer`)

static

Remove all buffered RPCs from server that were sent by `targetPlayer`. Can only be called on local player (for "self") or Master Client (for anyone).

This method requires either:

- This is the `targetPlayer`'s client.
- This client is the Master Client (can remove any Player's RPCs).

If the `targetPlayer` calls RPCs at the same time that this is called, network lag will determine if those get buffered or cleared like the rest.

Parameters

`targetPlayer` This player's buffered RPCs get removed from server buffer.

static void RemoveRPCs (PhotonView `targetPhotonView`)

static

Remove all buffered RPCs from server that were sent via `targetPhotonView`. The Master Client and the owner of the `targetPhotonView` may call this.

This method requires either:

- The `targetPhotonView` is owned by this client (Instantiated by it).
- This client is the Master Client (can remove any **PhotonView's**

RPCs).

Parameters

targetPhotonView RPCs buffered for this **PhotonView** get removed from server buffer.

static void RemoveRPCsInGroup (int **group)**

static

Remove all buffered RPCs from server that were sent in the targetGroup, if this is the Master Client or if this controls the individual **PhotonView**.

This method requires either:

- This client is the Master Client (can remove any RPCs per group).
- Any other client: each **PhotonView** is checked if it is under this client's control. Only those RPCs are removed.

Parameters

group Interest group that gets all RPCs removed.

static void SendAllOutgoingCommands ()

static

Can be used to immediately send the RPCs and Instantiates just called, so they are on their way to the other players.

This could be useful if you do a RPC to load a level and then load it yourself. While loading, no RPCs are sent to others, so this would delay the "load" RPC. You can send the RPC to "others", use this method, disable the message queue (by IsMessageQueueRunning) and then load.

static void SetInterestGroups (byte[] **disableGroups,
byte[] **enableGroups**
)**

static

Enable/disable receiving on given Interest Groups (applied to PhotonViews).

A client can tell the server which Interest Groups it's interested in. The server will only forward events for those Interest Groups to that client (saving bandwidth and performance).

See: <https://doc.photonengine.com/en-us/pun/v2/gameplay/interestgroups>

See: <https://doc.photonengine.com/en-us/pun/v2/demos-and-tutorials/package-demos/culling-demo>

Parameters

disableGroups The interest groups to disable (or null).

enableGroups The interest groups to enable (or null).

```
static void SetInterestGroups ( byte group,  
                               bool enabled  
                               )
```

static

Enable/disable receiving events from a given Interest Group.

A client can tell the server which Interest Groups it's interested in. The server will only forward events for those Interest Groups to that client (saving bandwidth and performance).

See: <https://doc.photonengine.com/en-us/pun/v2/gameplay/interestgroups>

See: <https://doc.photonengine.com/en-us/pun/v2/demos-and-tutorials/package-demos/culling-demo>

Parameters

group The interest group to affect.

enabled Sets if receiving from group to enabled (or not).

static void SetLevelPrefix (byte **prefix)**

static

Sets level prefix for PhotonViews instantiated later on. Don't set it if you need only one!

Important: If you don't use multiple level prefixes, simply don't set this value. The default value is optimized out of the traffic.

This won't affect existing PhotonViews (they can't be changed yet for existing PhotonViews).

Messages sent with a different level prefix will be received but not executed. This affects RPCs, Instantiates and synchronization.

Be aware that PUN never resets this value, you'll have to do so yourself.

Parameters

prefix Max value is short.MaxValue = 255

static bool SetMasterClient (Player **masterClientPlayer)**

static

Asks the server to assign another player as Master Client of your current room.

RPCs and RaiseEvent have the option to send messages only to the Master Client of a room. SetMasterClient affects which client gets those messages.

This method calls an operation on the server to set a new Master Client, which takes a roundtrip. In case of success, this client and the others get the new Master Client from the server.

SetMasterClient tells the server which current Master Client should be replaced with the new one. It will fail, if anything switches the Master Client moments earlier. There is no callback for this error. All clients should get the new Master Client assigned by the server anyways.

See also: **PhotonNetwork.MasterClient**

On v3 servers: The **ReceiverGroup.MasterClient** (usable in RPCs) is not affected by this (still points to lowest player.ID in room). Avoid using this enum value (and send to a specific player instead).

If the current Master Client leaves, PUN will detect a new one by "lowest player ID". Implement OnMasterClientSwitched to get a callback in this case. The PUN-selected Master Client might assign a new one.

Make sure you don't create an endless loop of Master-assigning! When selecting a custom Master Client, all clients should point to the same player, no matter who actually assigns this player.

Locally the Master Client is immediately switched, while remote clients get an event. This means the game is temporarily without Master Client like when a current Master Client leaves.

When switching the Master Client manually, keep in mind that this user might leave and not do it's work, just like any Master Client.

Parameters

masterClientPlayer The player to become the next Master Client.

Returns

False when this operation couldn't be done. Must be in a room (not in OfflineMode).

static void

SetPlayerCustomProperties (Hashtable **customProperties)**

static

Sets this (local) player's properties and synchronizes them to the other players (don't modify them directly).

While in a room, your properties are synced with the other players. CreateRoom, JoinRoom and JoinRandomRoom will all apply your player's custom properties when you enter the room. The whole

Hashtable will get sent. Minimize the traffic by setting only updated key/values.

If the Hashtable is null, the custom properties will be cleared. Custom properties are never cleared automatically, so they carry over to the next room, if you don't change them.

Don't set properties by modifying
PhotonNetwork.player.customProperties!

Parameters

customProperties Only string-typed keys will be used from this hashtable. If null, custom properties are all deleted.

```
static void SetSendingEnabled ( byte group,  
                               bool enabled  
                               )
```

static

Enable/disable sending on given group (applied to PhotonViews)

This does not interact with the **Photon** server-side. It's just a client-side setting to suppress updates, should they be sent to one of the blocked groups.

This setting is not particularly useful, as it means that updates literally never reach the server or anyone else. Use with care.

Parameters

group The interest group to affect.

enabled Sets if sending to group is enabled (or not).

```
static void SetSendingEnabled ( byte[] disableGroups,  
                               byte[] enableGroups  
                               )
```

static

Enable/disable sending on given groups (applied to PhotonViews)

This does not interact with the **Photon** server-side. It's just a client-side setting to suppress updates, should they be sent to one of the blocked groups.

This setting is not particularly useful, as it means that updates literally never reach the server or anyone else. Use with care.

Parameters

enableGroups The interest groups to enable sending on (or null).

disableGroups The interest groups to disable sending on (or null).

```
static bool WebRpc ( string name,
                    object parameters,
                    bool  sendAuthCookie = false
                    )
```

This operation makes **Photon** call your custom web-service by name (r with the given parameters.

This is a server-side feature which must be setup in the **Photon** Cloud Dashboard prior to use. <https://doc.photonengine.com/en-us/pun/v2/gameplay/web-extensions/webrpc> The Parameters will be converted into JSON format, so make sure your parameters are compat

See **Photon.Realtime.IWebRpcCallback.OnWebRpcResponse** on h get a response.

It's important to understand that the OperationResponse only tells if the WebRPC could be called. The content of the response contains any va your web-service sent and the error/success code. In case the web-ser failed, an error code and a debug message are usually inside the OperationResponse.

The class `WebRpcResponse` is a helper-class that extracts the most valuable content from the WebRPC response.

Example callback implementation:

```
public void OnWebRpcResponse(OperationResponse response)
{
    WebRpcResponse webResponse = new WebRpcResponse(operationResponse);
    if (webResponse.ReturnCode != 0) { //...
    }

    switch (webResponse.Name) { //...
    }
    // and so on
}
```

Member Data Documentation

**ConnectMethod ConnectMethod =
ConnectMethod.NotCalled**

static

Tracks, which Connect method was called last.

ConnectToMaster sets this to ConnectToMaster. ConnectToRegion sets this to ConnectToRegion. ConnectToBestCloudServer sets this to ConnectToBest. **PhotonNetwork.ConnectUsingSettings** will call either ConnectToMaster, ConnectToRegion or ConnectToBest, depending on the settings.

PunLogLevel LogLevel = PunLogLevel.ErrorsOnly

static

Controls how verbose PUN is.

readonly int MAX_VIEW_IDS = 1000

static

The maximum number of assigned PhotonViews *per player* (or scene). See the **General Documentation** topic "Limitations" on how to raise this limitation.

LoadBalancingClient NetworkingClient

static

The LoadBalancingClient is part of **Photon Realtime** and wraps up multiple servers and states for PUN.

int ObjectsInOneUpdate = 10

static

Defines how many OnPhotonSerialize()-calls might get summarized in one message.

A low number increases overhead, a high number might mean fragmentation.

**ServerSettings PhotonServerSettings =
(ServerSettings)Resources.Load(PhotonNetwork.ServerSettingsFi
typeof(ServerSettings))**

Serialized server settings, written by the Setup Wizard for use in ConnectUsingSettings.

float PrecisionForFloatSynchronization = 0.01f

static

The minimum difference between floats before we send it via a **PhotonView's** OnSerialize/ObservingComponent.

float PrecisionForQuaternionSynchronization = 1.0f

static

The minimum angle that a rotation needs to change before we send it via a **PhotonView's** OnSerialize/ObservingComponent.

float PrecisionForVectorSynchronization = 0.000099f

static

The minimum difference that a Vector2 or Vector3(e.g. a transforms rotation) needs to change before we send it via a **PhotonView's** OnSerialize/ObservingComponent.

Note that this is the sqrMagnitude. E.g. to send only after a 0.01 change on the Y-axis, we use $0.01f * 0.01f = 0.0001f$. As a remedy against float inaccuracy we use 0.000099f instead of 0.0001f.

const string PunVersion = "2.12"

Version number of PUN. Used in the AppVersion, which separates your playerbase in matchmaking.

bool UseRpcMonoBehaviourCache

static

While enabled, the MonoBehaviours on which we call RPCs are cached, avoiding costly `GetComponent<MonoBehaviour>()` calls.

RPCs are called on the MonoBehaviours of a target **PhotonView**. Those have to be found via `GetComponent`.

When set this to true, the list of MonoBehaviours gets cached in each **PhotonView**. You can use `photonView.RefreshRpcMonoBehaviourCache()` to manually refresh a **PhotonView**'s list of MonoBehaviours on demand (when a new MonoBehaviour gets added to a networked GameObject, e.g.).

Property Documentation

string AppVersion

static get

Sent to **Photon** Server to specify the "Virtual AppId".

Sent with the operation Authenticate. When using PUN, you should set the GameVersion or use **ConnectUsingSettings()**.

AuthenticationValues AuthValues

static

A user's authentication values used during connect.

Set these before calling Connect if you want custom authentication. The values set the userId, if and how that userId gets verified (server-side),

If authentication fails for any values, PUN will call your implementation of **OnCustomAuthenticationFailed(string debugMessage)**. See **Photon.Realtime.IConnectionCallbacks.OnCustomAuthenticationFailed**

bool AutomaticallySyncScene

static get set

Defines if all clients in a room should automatically load the same level as the Master Client.

When enabled, clients load the same scene that is active on the Master Client. When a client joins a room, the scene gets loaded even before the callback **OnJoinedRoom** gets called.

To synchronize the loaded level, the Master Client should use **PhotonNetwork.LoadLevel**, which notifies the other clients before starting to load the scene. If the Master Client loads a level directly via Unity's API, PUN will notify the other players after the scene

loading completed (using SceneManager.sceneLoaded).

Internally, a Custom Room Property is set for the loaded scene. On change, clients use LoadLevel if they are not in the same scene.

Note that this works only for a single active scene and that reloading the scene is not supported. The Master Client will actually reload a scene but other clients won't. To get everyone to reload, the game can send an RPC or event to trigger the loading.

string BestRegionSummaryInPreferences

static get set

Used to store and access the "Best Region Summary" in the Player Preferences.

string CloudRegion

static get

Currently used Cloud Region (if any). As long as the client is not on a Master Server or Game Server, the region is not yet defined.

int CountOfPlayers

static get

The count of players currently using this application (available on MasterServer in 5sec intervals).

int CountOfPlayersInRooms

static get

Count of users currently playing your app in some room (sent every 5sec by Master Server). Use PhotonNetwork.PlayerList.Length or PhotonNetwork.CurrentRoom.PlayerCount to get the count of players in the room you're in!

int CountOfPlayersOnMaster

static get

The count of players currently looking for a room (available on MasterServer in 5sec intervals).

int CountOfRooms

static get

The count of rooms currently in use (available on MasterServer in 5sec intervals).

bool CrcCheckEnabled

static get set

Crc checks can be useful to detect and avoid issues with broken datagrams. Can be enabled while not connected.

TypedLobby CurrentLobby

static get

The lobby that will be used when PUN joins a lobby or creates a game. This is defined when joining a lobby or creating rooms

The default lobby uses an empty string as name. So when you connect or leave a room, PUN automatically gets you into a lobby again.

Check **PhotonNetwork.InLobby** if the client is in a lobby.
(masterServerAndLobby)

Room CurrentRoom

static get

Get the room we're currently in (also when in OfflineMode). Null if we aren't in any room.

LoadBalancing Client is not aware of the **Photon** Offline Mode, so never use PhotonNetwork.NetworkingClient.CurrentRoom will be null if you are using OffLine Mode, while

PhotonNetwork.CurrentRoom will be set when **offlineMode** is true

bool EnableLobbyStatistics

static get

If enabled, the client will get a list of available lobbies from the Master Server.

Set this value before the client connects to the Master Server. While connected to the Master Server, a change has no effect.

Implement **OptionalInfoCallbacks.OnLobbyStatisticsUpdate**, to get the list of used lobbies.

The lobby statistics can be useful if your title dynamically uses lobbies, depending (e.g.) on current player activity or such. In this case, getting a list of available lobbies, their room-count and player-count can be useful info.

ConnectUsingSettings sets this to the **PhotonServerSettings** value.

string GameVersion

static get set

Version number of your game. Setting this updates the **AppVersion**, which separates your playerbase in matchmaking.

In PUN, the **GameVersion** is only one component of the **LoadBalancingClient.AppVersion**. Setting the **GameVersion** will also set the **LoadBalancingClient.AppVersion** to: `value+'_'+PhotonNetwork.PunVersion`.

The **AppVersion** is used to split your playerbase as needed. One AppId may have various **AppVersions** and each is a separate set of users for matchmaking.

The **AppVersion** gets sent in the "Authenticate" step. This means you can set the **GameVersion** right after calling **ConnectUsingSettings** (e.g.) and the new value will be used on the server. Once the client is connected, authentication is done and the

value won't be sent to the server anymore.

bool InLobby

static get

True while this client is in a lobby.

Implement `IPunCallbacks.OnRoomListUpdate()` for a notification when the list of rooms becomes available or updated.

You are automatically leaving any lobby when you join a room! Lobbies only exist on the Master Server (whereas rooms are handled by Game Servers).

bool InRoom

static get

Is true while being in a room (`NetworkClientState == ClientState.Joined`).

Aside from polling this value, game logic should implement `IMatchmakingCallbacks` in some class and react when that gets called.

Many actions can only be executed in a room, like `Instantiate` or `Leave`, etc.

A client can join a room in offline mode. In that case, don't use **`LoadBalancingClient.InRoom`**, which does not cover offline mode.

bool IsConnected

static get

False until you connected to **Photon** initially. True in offline mode, while connected to any server and even while switching servers.

bool IsConnectedAndReady

static get

A refined version of `connected` which is true only if your connection to the server is ready to accept operations like join, leave, etc.

bool IsMasterClient

static get

Are we the master client?

bool IsMessageQueueRunning

static get set

Can be used to pause dispatching of incoming events (RPCs, Instantiates and anything else incoming).

While `IsMessageQueueRunning == false`, the `OnPhotonSerializeView` calls are not done and nothing is sent by a client. Also, incoming messages will be queued until you re-activate the message queue.

This can be useful if you first want to load a level, then go on receiving data of PhotonViews and RPCs. The client will go on receiving and sending acknowledgements for incoming packages and your RPCs/Events. This adds "lag" and can cause issues when the pause is longer, as all incoming messages are just queued.

float KeepAliveInBackground

static get set

Defines how many seconds PUN keeps the connection after Unity's `OnApplicationPause(true)` call. Default: 60 seconds.

It's best practice to disconnect inactive apps/connections after a while but to also allow users to take calls, etc.. We think a reasonable background timeout is 60 seconds.

To handle the timeout, implement: `OnDisconnected()`, as usual. Your application will "notice" the background disconnect when it becomes active again (running the `Update()` loop).

If you need to separate this case from others, you need to track if the app was in the background (there is no special callback by PUN).

Info: PUN is running a "fallback thread" to send ACKs to the server, even when Unity is not calling Update() regularly. This helps keeping the connection while loading scenes and assets and when the app is in the background.

Note: Some platforms (e.g. iOS) don't allow to keep a connection while the app is in background. In those cases, this value does not change anything, the app immediately loses connection in background.

Unity's OnApplicationPause() callback is broken in some exports (Android) of some Unity versions. Make sure OnApplicationPause() gets the callbacks you expect on the platform you target! Check PhotonHandler.OnApplicationPause(bool pause) to see the implementation.

float LevelLoadingProgress

static get

Represents the scene loading progress when using **LoadLevel()**.

The value is 0 if the app never loaded a scene with **LoadLevel()**. During async scene loading, the value is between 0 and 1. Once any scene completed loading, it stays at 1 (signaling "done").

The level loading progress. Ranges from 0 to 1.

Player LocalPlayer

static get

This client's Player instance is always available, unless the app shuts down.

Useful (e.g.) to set the Custom Player Properties or the NickName for this client anytime. When the client joins a room, the Custom Properties and other values are synced.

Player MasterClient

static get

The Master Client of the current room or null (outside of rooms).

Can be used as "authoritative" client/player to make descisions, run AI or other.

If the current Master Client leaves the room (leave/disconnect), the server will quickly assign someone else. If the current Master Client times out (closed app, lost connection, etc), messages sent to this client are effectively lost for the others! A timeout can take 10 seconds in which no Master Client is active.

Implement the method `IPunCallbacks.OnMasterClientSwitched` to be called when the Master Client switched.

Use **PhotonNetwork.SetMasterClient**, to switch manually to some other player / client.

With `OfflineMode == true`, this always returns the `PhotonNetwork.player`.

int MaxResendsBeforeDisconnect

static get set

Defines the number of times a reliable message can be resent before not getting an ACK for it will trigger a disconnect. Default: 5.

Less resends mean quicker disconnects, while more can lead to much more lag without helping. Min: 3. Max: 10.

ClientState NetworkClientState

static get

Directly provides the network-level client state, unless in `OfflineMode`.

In context of PUN, you should usually use `IsConnected` or

IsConnectedAndReady.

This is the lower level connection state. Keep in mind that PUN uses more than one server, so the client may become Disconnected, even though it's just switching servers.

While OfflineMode is true, this is ClientState.Joined (after create/join) or ConnectedToMasterServer in all other cases.

bool NetworkStatisticsEnabled

static get set

Enables or disables the collection of statistics about this client's traffic.

If you encounter issues with clients, the traffic stats are a good starting point to find solutions. Only with enabled stats, you can use GetVitalStats

string NickName

static get set

Set to synchronize the player's nickname with everyone in the room(s) you enter. This sets PhotonNetwork.player.NickName.

The NickName is just a nickname and does not have to be unique or backed up with some account.

Set the value any time (e.g. before you connect) and it will be available to everyone you play with.

Access the names of players by: **Player.NickName**.

PhotonNetwork.PlayerListOthers is a list of other players - each contains the NickName the remote player set.

bool OfflineMode

static get set

Offline mode can be set to re-use your multiplayer code in singleplayer game modes. When this is on **PhotonNetwork** will not create any connections and there is near to no overhead. Mostly

usefull for reusing RPC's and PhotonNetwork.Instantiate

int PacketLossByCrcCheck

static get

If CrcCheckEnabled, this counts the incoming packages that don't have a valid CRC checksum and got rejected.

PhotonView [] PhotonViews

static get

Gets the photon views.

This is an expensive operation as it returns a copy of the internal list.

The photon views.

Player [] PlayerList

static get

A sorted copy of the players-list of the current room. This is using Linq, so better cache this value. Update when players join / leave.

Player [] PlayerListOthers

static get

A sorted copy of the players-list of the current room, excluding this client. This is using Linq, so better cache this value. Update when players join / leave.

IPunPrefabPool PrefabPool

static get set

An Object Pool can be used to keep and reuse instantiated object instances. Replaces Unity's default Instantiate and Destroy methods.

Defaults to the **DefaultPool** type. To use a GameObject pool, implement **IPunPrefabPool** and assign it here. Prefabs are

identified by name.

int QuickResends

static get set

In case of network loss, reliable messages can be repeated quickly up to 3 times.

When reliable messages get lost more than once, subsequent repeats are delayed a bit to allow the network to recover. With this option, the repeats 2 and 3 can be sped up. This can help avoid timeouts but also it increases the speed in which gaps are closed.

When you set this, increase

PhotonNetwork.MaxResendsBeforeDisconnect to 6 or 7.

int ResentReliableCommands

static get

Count of commands that got repeated (due to local repeat-timing before an ACK was received).

If this value increases a lot, there is a good chance that a timeout disconnect will happen due to bad conditions.

int SendRate

static get set

Defines how many times per second **PhotonNetwork** should send a package. If you change this, do not forget to also change 'SerializationRate'.

Less packages are less overhead but more delay. Setting the SendRate to 50 will create up to 50 packages per second (which is a lot!). Keep your target platform in mind: mobile networks are slower and less reliable.

int SerializationRate

static get set

Defines how many times per second OnPhotonSerialize should be called on PhotonViews.

Choose this value in relation to **PhotonNetwork.SendRate**. OnPhotonSerialize will create updates and messages to be sent. A lower rate takes up less performance but will cause more lag.

ServerConnection Server

static get

The server (type) this client is currently connected or connecting to.

Photon uses 3 different roles of servers: Name Server, Master Server and Game Server.

string ServerAddress

static get

Currently used server address (no matter if master or game server).

int ServerTimestamp

static get

The current server's millisecond timestamp.

This can be useful to sync actions and events on all clients in one room. The timestamp is based on the server's Environment.TickCount.

It will overflow from a positive to a negative value every so often, so be careful to use only time-differences to check the Time delta when things happen.

This is the basis for **PhotonNetwork.Time**.

double Time

static get

Photon network time, synched with the server.

v1.55

This time value depends on the server's Environment.TickCount. It is different per server but inside a Room, all clients should have the same value (Rooms are on one server only).

This is not a DateTime!

Use this value with care:

It can start with any positive value.

It will "wrap around" from 4294967.295 to 0!

bool UseAlternativeUdpPorts

static get set

Switch to alternative ports for a UDP connection to the Public Cloud.

This should be used when a customer has issues with connection stability. Some players reported better connectivity for Steam games. The effect might vary, which is why the alternative ports are not the new default.

The alternative (server) ports are 27000 up to 27003.

The values are applied by replacing any incoming server-address string accordingly. You only need to set this to true though.

This value does not affect TCP or WebSocket connections.



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy	Class Members	
Photon	Pun	PhotonView				

PhotonView Class Reference

Public API

[Public Member Functions](#) |
[Static Public Member Functions](#) |
[Public Attributes](#) | [Properties](#) |
[List of all members](#)

A **PhotonView** identifies an object across the network (viewID) and configures how the controlling client updates remote instances. [More...](#)

Inherits MonoBehaviour.

Public Member Functions

void **RequestOwnership** ()
Depending on the **PhotonView**'s OwnershipTransfer setting, any client can request to become owner of the **PhotonView**. [More...](#)

void **TransferOwnership** (Player newOwner)
Transfers the ownership of this **PhotonView** (and GameObject) to another player. [More...](#)

void **TransferOwnership** (int newOwnerId)
Transfers the ownership of this **PhotonView** (and GameObject) to another player. [More...](#)

void **SerializeView** (PhotonStream stream, PhotonMessageInfo info)

void **DeserializeView** (PhotonStream stream, PhotonMessageInfo info)

void **RefreshRpcMonoBehaviourCache** ()
Can be used to refresh the list of MonoBehaviours on this GameObject while **PhotonNetwork.UseRpcMonoBehaviourCache** is true. [More...](#)

void **RPC** (string methodName, **RpcTarget** target, params object[] parameters)
Call a RPC method of this GameObject on remote clients of this room (or on all, including this client). [More...](#)

void **RpcSecure** (string methodName, **RpcTarget** target, bool encrypt, params object[] parameters)
Call a RPC method of this GameObject on remote clients of this room (or on all, including this client).

[More...](#)

void **RPC** (string methodName, **Player** targetPlayer, params object[] parameters)
Call a RPC method of this GameObject on remote clients of this room (or on all, including this client).
[More...](#)

void **RpcSecure** (string methodName, **Player** targetPlayer, bool encrypt, params object[] parameters)
Call a RPC method of this GameObject on remote clients of this room (or on all, including this client).
[More...](#)

override string **ToString** ()

Static Public Member Functions

static **PhotonView** **Get** (Component component)

static **PhotonView** **Get** (GameObject gameObj)

static **PhotonView** **Find** (int viewID)

Public Attributes

byte **Group** = 0

bool **OwnershipWasTransferred**

Flag to check if ownership of this photonView was set during the lifecycle. Used for checking when joining late if event with mismatched owner and sender needs addressing. [More...](#)

int **prefixField** = -1

ViewSynchronization **Synchronization**

OwnershipOption **OwnershipTransfer** = **OwnershipOption.Fixed**

Defines if ownership of this **PhotonView** is fixed, can be requested or simply taken. [More...](#)

List< Component > **ObservedComponents**

int **InstantiationId**

Properties

int **Prefix** [get, set]

object[] **InstantiationData** [get, set]

This is the **InstantiationData** that was passed when calling **PhotonNetwork.Instantiate*** (if that was used to spawn this prefab) [More...](#)

int **ViewID** [get, set]

The ID of the **PhotonView**. Identifies it in a networked game (per room). [More...](#)

bool **IsSceneView** [get]

True if the **PhotonView** was loaded with the scene (game object) or instantiated with **InstantiateSceneObject**. [More...](#)

Player Owner [get]

The owner of a **PhotonView** is the player who created the **GameObject** with that view. Objects in the scene don't have an owner. [More...](#)

int **OwnerActorNr** [get, set]

Player Controller [get]

int **ControllerActorNr** [get]

bool **IsOwnerActive** [get]

int **CreatorActorNr** [get]

bool **IsMine** [get]

True if the **PhotonView** is "mine" and can be controlled by this client. [More...](#)

Detailed Description

A **PhotonView** identifies an object across the network (viewID) and configures how the controlling client updates remote instances.

Member Function Documentation

void RefreshRpcMonoBehaviourCache ()

Can be used to refresh the list of MonoBehaviours on this **GameObject** while **PhotonNetwork.UseRpcMonoBehaviourCache** is true.

Set **PhotonNetwork.UseRpcMonoBehaviourCache** to true to enable the caching. Uses this.**GetComponent<MonoBehaviour>()** to get a list of MonoBehaviours to call RPCs on (potentially).

While **PhotonNetwork.UseRpcMonoBehaviourCache** is false, this method has no effect, because the list is refreshed when a RPC gets called.

void RequestOwnership ()

Depending on the **PhotonView's** OwnershipTransfer setting, any client can request to become owner of the **PhotonView**.

Requesting ownership can give you control over a **PhotonView**, if the OwnershipTransfer setting allows that. The current owner might have to implement **IPunCallbacks.OnOwnershipRequest** to react to the ownership request.

The owner/controller of a **PhotonView** is also the client which sends position updates of the **GameObject**.

```
void RPC ( string          methodName,  
           RpcTarget        target,  
           params object[] parameters  
           )
```

Call a RPC method of this GameObject on remote clients of this room (or on all, including this client).

Remote Procedure Calls are an essential tool in making multiplayer games with PUN. It enables you to make every client in a room call a specific method.

RPC calls can target "All" or the "Others". Usually, the target "All" gets executed locally immediately after sending the RPC. The "*ViaServer" options send the RPC to the server and execute it on this client when it's sent back. Of course, calls are affected by this client's lag and that of remote clients.

Each call automatically is routed to the same **PhotonView** (and GameObject) that was used on the originating client.

See: Remote Procedure Calls.

Parameters

- methodName** The name of a fitting method that was has the RPC attribute.
- target** The group of targets and the way the RPC gets sent.
- parameters** The parameters that the RPC method has (must fit this call!).

```
void RPC ( string      methodName,  
           Player      targetPlayer,  
           params object[] parameters  
           )
```

Call a RPC method of this GameObject on remote clients of this room (or on all, including this client).

Remote Procedure Calls are an essential tool in making multiplayer games with PUN. It enables you to make every client in a room call a specific method.

This method allows you to make an RPC calls on a specific player's client. Of course, calls are affected by this client's lag and that of remote clients.

Each call automatically is routed to the same **PhotonView** (and **GameObject**) that was used on the originating client.

See: Remote Procedure Calls.

Parameters

- methodName** The name of a fitting method that was has the RPC attribute.
- targetPlayer** The group of targets and the way the RPC gets sent.
- parameters** The parameters that the RPC method has (must fit this call!).

```
void RpcSecure ( string      methodName,  
                 RpcTarget  target,  
                 bool       encrypt,  
                 params object[] parameters  
               )
```

Call a RPC method of this **GameObject** on remote clients of this room (or on all, including this client).

Remote Procedure Calls are an essential tool in making multiplayer games with PUN. It enables you to make every client in a room call a specific method.

RPC calls can target "All" or the "Others". Usually, the target "All" gets executed locally immediately after sending the RPC. The "*ViaServer" options send the RPC to the server and execute it on this client when it's sent back. Of course, calls are affected by this client's lag and that of remote clients.

Each call automatically is routed to the same **PhotonView** (and **GameObject**) that was used on the originating client.

See: Remote Procedure Calls.

param name="methodName">The name of a fitting method that was has the RPC attribute.

param name="target">The group of targets and the way the RPC gets sent.

param name="encrypt">

param name="parameters">The parameters that the RPC method has (must fit this call!).

```
void RpcSecure ( string      methodName,  
                 Player      targetPlayer,  
                 bool         encrypt,  
                 params object[] parameters  
               )
```

Call a RPC method of this GameObject on remote clients of this room (or on all, including this client).

Remote Procedure Calls are an essential tool in making multiplayer games with PUN. It enables you to make every client in a room call a specific method.

This method allows you to make an RPC calls on a specific player's client. Of course, calls are affected by this client's lag and that of remote clients.

Each call automatically is routed to the same **PhotonView** (and GameObject) that was used on the originating client.

See: Remote Procedure Calls.

param name="methodName">The name of a fitting method that was has the RPC attribute.

param name="targetPlayer">The group of targets and the way the

RPC gets sent.

param name="encrypt">

param name="parameters">The parameters that the RPC method has (must fit this call!).

void TransferOwnership (Player **newOwner)**

Transfers the ownership of this **PhotonView** (and GameObject) to another player.

The owner/controller of a **PhotonView** is also the client which sends position updates of the GameObject.

void TransferOwnership (int **newOwnerId)**

Transfers the ownership of this **PhotonView** (and GameObject) to another player.

The owner/controller of a **PhotonView** is also the client which sends position updates of the GameObject.

Member Data Documentation

OwnershipOption OwnershipTransfer = OwnershipOption.Fixed

Defines if ownership of this **PhotonView** is fixed, can be requested or simply taken.

Note that you can't edit this value at runtime. The options are described in enum `OwnershipOption`. The current owner has to implement `IPunCallbacks.OnOwnershipRequest` to react to the ownership request.

bool OwnershipWasTransferred

Flag to check if ownership of this `PhotonView` was set during the lifecycle. Used for checking when joining late if event with mismatched owner and sender needs addressing.

true if owner ship was transfered; otherwise, false.

Property Documentation

object [] InstantiationData

get set

This is the InstantiationData that was passed when calling PhotonNetwork.Instantiate* (if that was used to spawn this prefab)

bool IsMine

get

True if the **PhotonView** is "mine" and can be controlled by this client.

PUN has an ownership concept that defines who can control and destroy each **PhotonView**. True in case the owner matches the local Player. True if this is a scene photonview on the Master client.

bool IsSceneView

get

True if the **PhotonView** was loaded with the scene (game object) or instantiated with InstantiateSceneObject.

Scene objects are not owned by a particular player but belong to the scene. Thus they don't get destroyed when their creator leaves the game and the current Master Client can control them (whoever that is). The ownerId is 0 (player IDs are 1 and up).

Player Owner

get

The owner of a **PhotonView** is the player who created the GameObject with that view. Objects in the scene don't have an owner.

The owner/controller of a **PhotonView** is also the client which sends position updates of the GameObject.

Ownership can be transferred to another player with **PhotonView.TransferOwnership** or any player can request ownership by calling the **PhotonView's** RequestOwnership method. The current owner has to implement IPunCallbacks.OnOwnershipRequest to react to the ownership request.

int ViewID

get set

The ID of the **PhotonView**. Identifies it in a networked game (per room).

See: Network Instantiation



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonMessageInfo		

PhotonMessageInfo Struct Reference

Public API

[Public Member Functions](#) | [Public Attributes](#) |
[Properties](#) | [List of all members](#)

Container class for info about a particular message, RPC or update.
[More...](#)

Public Member Functions

PhotonMessageInfo (**Player** player, int timestamp,
PhotonView view)

override string **ToString** ()

Public Attributes

readonly **Player** **Sender**

The sender of a message / event. May be null. [More...](#)

readonly **PhotonView** **photonView**

Properties

double **timestamp** [get]

double **SentServerTime** [get]

int **SentServerTimestamp** [get]

Detailed Description

Container class for info about a particular message, RPC or update.

Member Data Documentation

readonly Player Sender
The sender of a message / event. May be null.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonStream		

[Public Member Functions](#) | [Properties](#) |

[List of all members](#)

PhotonStream Class Reference

Public API

This container is used in `OnPhotonSerializeView()` to either provide incoming data of a **PhotonView** or for you to provide it. [More...](#)

Public Member Functions

PhotonStream (bool write, object[] incomingData)
Creates a stream and initializes it. Used by PUN internally.
[More...](#)

void **SetReadStream** (object[] incomingData, byte pos=0)

object **ReceiveNext** ()
Read next piece of data from the stream when IsReading is true. [More...](#)

object **PeekNext** ()
Read next piece of data from the stream without advancing the "current" item. [More...](#)

void **SendNext** (object obj)
Add another piece of data to send it when IsWriting is true.
[More...](#)

bool **CopyToListAndClear** (List< object > target)

object[] **ToArray** ()
Turns the stream into a new object[]. [More...](#)

void **Serialize** (ref bool myBool)
Will read or write the value, depending on the stream's IsWriting value. [More...](#)

void **Serialize** (ref int myInt)
Will read or write the value, depending on the stream's IsWriting value. [More...](#)

void **Serialize** (ref string value)
Will read or write the value, depending on the stream's IsWriting value. [More...](#)

void **Serialize** (ref char value)
Will read or write the value, depending on the stream's IsWriting value. [More...](#)

void **Serialize** (ref short value)
Will read or write the value, depending on the stream's IsWriting value. [More...](#)

void **Serialize** (ref float obj)
Will read or write the value, depending on the stream's IsWriting value. [More...](#)

void **Serialize** (ref **Player** obj)
Will read or write the value, depending on the stream's IsWriting value. [More...](#)

void **Serialize** (ref Vector3 obj)
Will read or write the value, depending on the stream's IsWriting value. [More...](#)

void **Serialize** (ref Vector2 obj)
Will read or write the value, depending on the stream's IsWriting value. [More...](#)

void **Serialize** (ref Quaternion obj)
Will read or write the value, depending on the stream's IsWriting value. [More...](#)

Properties

bool **IsWriting** [get]

If true, this client should add data to the stream to send it.

[More...](#)

bool **IsReading** [get]

If true, this client should read data send by another client.

[More...](#)

int **Count** [get]

Count of items in the stream. [More...](#)

Detailed Description

This container is used in `OnPhotonSerializeView()` to either provide incoming data of a **PhotonView** or for you to provide it.

The `IsWriting` property will be true if this client is the "owner" of the **PhotonView** (and thus the `GameObject`). Add data to the stream and it's sent via the server to the other players in a room. On the receiving side, `IsWriting` is false and the data should be read.

Send as few data as possible to keep connection quality up. An empty **PhotonStream** will not be sent.

Use either **Serialize()** for reading and writing or **SendNext()** and **ReceiveNext()**. The latter two are just explicit read and write methods but do about the same work as **Serialize()**. It's a matter of preference which methods you use.

Constructor & Destructor Documentation

```
PhotonStream ( bool    write,  
               object[] incomingData  
               )
```

Creates a stream and initializes it. Used by PUN internally.

Member Function Documentation

object PeekNext ()

Read next piece of data from the stream without advancing the "current" item.

object ReceiveNext ()

Read next piece of data from the stream when IsReading is true.

void SendNext (object **obj)**

Add another piece of data to send it when IsWriting is true.

void Serialize (ref bool **myBool)**

Will read or write the value, depending on the stream's IsWriting value.

void Serialize (ref int **myInt)**

Will read or write the value, depending on the stream's IsWriting value.

void Serialize (ref string **value)**

Will read or write the value, depending on the stream's IsWriting

value.

void Serialize (ref char *value*)

Will read or write the value, depending on the stream's IsWriting value.

void Serialize (ref short *value*)

Will read or write the value, depending on the stream's IsWriting value.

void Serialize (ref float *obj*)

Will read or write the value, depending on the stream's IsWriting value.

void Serialize (ref Player *obj*)

Will read or write the value, depending on the stream's IsWriting value.

void Serialize (ref Vector3 *obj*)

Will read or write the value, depending on the stream's IsWriting value.

void Serialize (ref Vector2 *obj*)

Will read or write the value, depending on the stream's IsWriting value.

void Serialize (ref Quaternion **obj)**

Will read or write the value, depending on the stream's IsWriting value.

object [] ToArray ()

Turns the stream into a new object[].

Property Documentation

int Count

get

Count of items in the stream.

bool IsReading

get

If true, this client should read data send by another client.

bool IsWriting

get

If true, this client should add data to the stream to send it.



Photon Unity Networking 2 2.12

[Main Page](#)[Related Pages](#)[Modules](#)[Classes](#)[Classes](#)

Optional Gui Elements

Useful GUI elements for PUN. [More...](#)

Classes

class **PhotonLagSimulationGui**

This MonoBehaviour is a basic GUI for the **Photon** client's network-simulation feature. It can modify lag (fixed delay), jitter (random lag) and packet loss. [More...](#)

class **PhotonStatsGui**

Basic GUI to show traffic and health statistics of the connection to **Photon**, toggled by shift+tab. [More...](#)

Detailed Description

Useful GUI elements for PUN.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	PhotonLagSimulationGui	

[Public Member Functions](#) | [Public Attributes](#) |

[Properties](#) | [List of all members](#)

PhotonLagSimulationGui Class Reference

Optional Gui Elements

This MonoBehaviour is a basic GUI for the **Photon** client's network-simulation feature. It can modify lag (fixed delay), jitter (random lag) and packet loss. [More...](#)

Inherits MonoBehaviour.

Public Member Functions

void **Start** ()

void **OnGUI** ()

Public Attributes

Rect **WindowRect** = new Rect(0, 100, 120, 100)
Positioning rect for window. [More...](#)

int **WindowId** = 101
Unity GUI Window ID (must be unique or will cause issues).
[More...](#)

bool **Visible** = true
Shows or hides GUI (does not affect settings). [More...](#)

Properties

PhotonPeer **Peer** [get, set]

The peer currently in use (to set the network simulation). [More...](#)

Detailed Description

This MonoBehaviour is a basic GUI for the **Photon** client's network-simulation feature. It can modify lag (fixed delay), jitter (random lag) and packet loss.

Member Data Documentation

bool Visible = true

Shows or hides GUI (does not affect settings).

int WindowId = 101

Unity GUI Window ID (must be unique or will cause issues).

Rect WindowRect = new Rect(0, 100, 120, 100)

Positioning rect for window.

Property Documentation

PhotonPeer Peer

[get](#) [set](#)

The peer currently in use (to set the network simulation).

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	PhotonStatsGui			

[Public Member Functions](#) | [Public Attributes](#) |

[List of all members](#)

PhotonStatsGui Class Reference

Optional Gui Elements

Basic GUI to show traffic and health statistics of the connection to **Photon**, toggled by shift+tab. [More...](#)

Inherits MonoBehaviour.

Public Member Functions

void **Start** ()

void **Update** ()

Checks for shift+tab input combination (to toggle statsOn).
More...

void **OnGUI** ()

void **TrafficStatsWindow** (int windowID)

Public Attributes

bool **statsWindowOn** = true

Shows or hides GUI (does not affect if stats are collected).

[More...](#)

bool **statsOn** = true

Option to turn collecting stats on or off (used in **Update()**).

[More...](#)

bool **healthStatsVisible**

Shows additional "health" values of connection. [More...](#)

bool **trafficStatsOn**

Shows additional "lower level" traffic stats. [More...](#)

bool **buttonsOn**

Show buttons to control stats and reset them. [More...](#)

Rect **statsRect** = new Rect(0, 100, 200, 50)

Positioning rect for window. [More...](#)

int **WindowId** = 100

Unity GUI Window ID (must be unique or will cause issues).

[More...](#)

Detailed Description

Basic GUI to show traffic and health statistics of the connection to **Photon**, toggled by shift+tab.

The shown health values can help identify problems with connection losses or performance. Example: If the time delta between two consecutive `SendOutgoingCommands` calls is a second or more, chances rise for a disconnect being caused by this (because acknowledgements to the server need to be sent in due time).

Member Function Documentation

void Update ()

Checks for shift+tab input combination (to toggle statsOn).

Member Data Documentation

bool buttonsOn

Show buttons to control stats and reset them.

bool healthStatsVisible

Shows additional "health" values of connection.

bool statsOn = true

Option to turn collecting stats on or off (used in **Update()**).

Rect statsRect = new Rect(0, 100, 200, 50)

Positioning rect for window.

bool statsWindowOn = true

Shows or hides GUI (does not affect if stats are collected).

bool trafficStatsOn

Shows additional "lower level" traffic stats.

int WindowId = 100

Unity GUI Window ID (must be unique or will cause issues).

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

[Main Page](#)[Related Pages](#)[Modules](#)[Classes](#)[Classes](#)

Callbacks

Callback Interfaces. [More...](#)

Classes

interface **IConnectionCallbacks**
Collection of "organizational" callbacks for the **Realtime** Api to cover: Connection and Regions. [More...](#)

interface **ILobbyCallbacks**
Collection of "organizational" callbacks for the **Realtime** Api to cover the Lobby. [More...](#)

interface **IMatchmakingCallbacks**
Collection of "organizational" callbacks for the **Realtime** Api to cover Matchmaking. [More...](#)

interface **IInRoomCallbacks**
Collection of "in room" callbacks for the **Realtime** Api to cover: Players entering or leaving, property updates and Master Client switching. [More...](#)

interface **IONEventCallback**
Event callback for the **Realtime** Api. Covers events from the server and those sent by clients via OpRaiseEvent. [More...](#)

interface **IWebRpcCallback**
Interface for "WebRpc" callbacks for the **Realtime** Api. Currently includes only responses for Web RPCs. [More...](#)

interface **IPunObservable**
Defines the OnPhotonSerializeView method to make it easy to implement correctly for observable scripts. [More...](#)

interface **IPunOwnershipCallbacks**
This interface is used as definition of all callback methods of PUN, except OnPhotonSerializeView. Preferably, implement them individually. [More...](#)

interface **IPunInstantiateMagicCallback**

class **MonoBehaviourPunCallbacks**

This class provides a .photonView and all callbacks/events that PUN can call. Override the events/methods you want to use. [More...](#)

Detailed Description

Callback Interfaces.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	IConnectionCallbacks		

[Public Member Functions](#) | [List of all members](#)

IConnectionCallbacks Interface Reference

Callbacks

Collection of "organizational" callbacks for the **Realtime** Api to cover: Connection and Regions. [More...](#)

Inherited by **MonoBehaviourPunCallbacks**, **OnJoinedInstantiate**, **ConnectionCallbacksContainer**, and **SupportLogger**.

Public Member Functions

void **OnConnected** ()

Called to signal that the "low level connection" got established but before the client can call operation on the server. [More...](#)

void **OnConnectedToMaster** ()

Called when the client is connected to the Master Server and ready for matchmaking and other tasks. [More...](#)

void **OnDisconnected** (**DisconnectCause** cause)

Called after disconnecting from the **Photon** server. It could be a failure or an explicit disconnect call [More...](#)

void **OnRegionListReceived** (**RegionHandler** regionHandler)

Called when the Name Server provided a list of regions for your title. [More...](#)

void **OnCustomAuthenticationResponse** (Dictionary< string, object > data)

Called when your Custom Authentication service responds with additional data. [More...](#)

void **OnCustomAuthenticationFailed** (string debugMessage)

Called when the custom authentication failed. Followed by disconnect! [More...](#)

Detailed Description

Collection of "organizational" callbacks for the **Realtime** Api to cover: Connection and Regions.

Classes that implement this interface must be registered to get callbacks for various situations.

To register for callbacks, `PhotonNetwork.AddCallbackTarget(<Your Component implementing this interface>);` To stop getting callbacks, `PhotonNetwork.RemoveCallbackTarget(<Your Component implementing this interface>);`

You can also simply override `MonoBehaviourPunCallbacks` which will provide you with Magic Callbacks (like Unity would call `Start()`, `Update()` on a `MonoBehaviour`)

Member Function Documentation

void OnConnected ()

Called to signal that the "low level connection" got established but before the client can call operation on the server.

After the (low level transport) connection is established, the client will automatically send the Authentication operation, which needs to get a response before the client can call other operations.

Your logic should wait for either: `OnRegionListReceived` or `OnConnectedToMaster`.

This callback is useful to detect if the server can be reached at all (technically). Most often, it's enough to implement **`OnDisconnected(DisconnectCause cause)`** and check for the cause.

This is not called for transitions from the masterserver to game servers.

Implemented in **`ConnectionCallbacksContainer`**, **`MonoBehaviourPunCallbacks`**, **`SupportLogger`**, and **`OnJoinedInstantiate`**.

void OnConnectedToMaster ()

Called when the client is connected to the Master Server and ready for matchmaking and other tasks.

The list of available rooms won't become available unless you join a lobby via **`LoadBalancingClient.OpJoinLobby`**. You can join rooms and create them even without being in a lobby. The default lobby is used in that case.

Implemented in **ConnectionCallbacksContainer**, **MonoBehaviourPunCallbacks**, **SupportLogger**, **OnJoinedInstantiate**, and **ConnectAndJoinRandom**.

void OnCustomAuthenticationFailed (string **debugMessage)**

Called when the custom authentication failed. Followed by disconnect!

Custom Authentication can fail due to user-input, bad tokens/secrets. If authentication is successful, this method is not called. Implement **OnJoinedLobby()** or **OnConnectedToMaster()** (as usual).

During development of a game, it might also fail due to wrong configuration on the server side. In those cases, logging the **debugMessage** is very important.

Unless you setup a custom authentication service for your app (in the [Dashboard](#)), this won't be called!

Parameters

debugMessage Contains a debug message why authentication failed. This has to be fixed during development.

Implemented in **ConnectionCallbacksContainer**, **MonoBehaviourPunCallbacks**, **SupportLogger**, and **OnJoinedInstantiate**.

void OnCustomAuthenticationResponse (Dictionary< string, object > c

Called when your Custom Authentication service responds with additional data.

Custom Authentication services can include some custom data in their

response. When present, that data is made available in this callback as Dictionary. While the keys of your data have to be strings, the values can be either string or a number (in Json). You need to make extra sure, that the value type is the one you expect. Numbers become (currently) int64

Example: void OnCustomAuthenticationResponse(Dictionary<string, object> data) { ... }

<https://doc.photonengine.com/en-us/realtime/current/reference/custom-authentication>

Implemented in **ConnectionCallbacksContainer**, **MonoBehaviourPunCallbacks**, **SupportLogger**, and **OnJoinedInstantiate**.

void OnDisconnected (DisconnectCause **cause)**

Called after disconnecting from the **Photon** server. It could be a failure or an explicit disconnect call

The reason for this disconnect is provided as DisconnectCause.

Implemented in **ConnectionCallbacksContainer**, **MonoBehaviourPunCallbacks**, **SupportLogger**, **OnJoinedInstantiate**, and **ConnectAndJoinRandom**.

void OnRegionListReceived (RegionHandler **regionHandler)**

Called when the Name Server provided a list of regions for your title.

Check the **RegionHandler** class description, to make use of the provided values.

Parameters

regionHandler The currently used **RegionHandler**.

Implemented in **ConnectionCallbacksContainer**, **MonoBehaviourPunCallbacks**, **SupportLogger**, and

OnJoinedInstantiate.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	ILobbyCallbacks		

[Public Member Functions](#) | [List of all members](#)

ILobbyCallbacks Interface Reference

Callbacks

Collection of "organizational" callbacks for the **Realtime** Api to cover the Lobby. [More...](#)

Inherited by **MonoBehaviourPunCallbacks**, **OnJoinedInstantiate**, **LobbyCallbacksContainer**, and **SupportLogger**.

Public Member Functions

void **OnJoinedLobby** ()

Called on entering a lobby on the Master Server. The actual room-list updates will call OnRoomListUpdate. [More...](#)

void **OnLeftLobby** ()

Called after leaving a lobby. [More...](#)

void **OnRoomListUpdate** (List< **RoomInfo** > roomList)

Called for any update of the room-listing while in a lobby (InLobby) on the Master Server. [More...](#)

void **OnLobbyStatisticsUpdate** (List< **TypedLobbyInfo** > lobbyStatistics)

Called when the Master Server sent an update for the Lobby Statistics, updating PhotonNetwork.LobbyStatistics. [More...](#)

Detailed Description

Collection of "organizational" callbacks for the **Realtime** Api to cover the Lobby.

Classes that implement this interface must be registered to get callbacks for various situations.

To register for callbacks, `PhotonNetwork.AddCallbackTarget(<Your Component implementing this interface>);` To stop getting callbacks, `PhotonNetwork.RemoveCallbackTarget(<Your Component implementing this interface>);`

You can also simply override `MonoBehaviourPunCallbacks` which will provide you with Magic Callbacks (like Unity would call `Start()`, `Update()` on a `MonoBehaviour`)

Member Function Documentation

void OnJoinedLobby ()

Called on entering a lobby on the Master Server. The actual room-list updates will call OnRoomListUpdate.

While in the lobby, the roomlist is automatically updated in fixed intervals (which you can't modify in the public cloud). The room list gets available via OnRoomListUpdate.

Implemented in **MonoBehaviourPunCallbacks**, **SupportLogger**, **OnJoinedInstantiate**, and **ConnectAndJoinRandom**.

void OnLeftLobby ()

Called after leaving a lobby.

When you leave a lobby, OpCreateRoom and OpJoinRandomRoom automatically refer to the default lobby.

Implemented in **MonoBehaviourPunCallbacks**, **SupportLogger**, and **OnJoinedInstantiate**.

void OnLobbyStatisticsUpdate (List< TypedLobbyInfo > lobbyStatistics)

Called when the Master Server sent an update for the Lobby Statistics, updating PhotonNetwork.LobbyStatistics.

This callback has two preconditions: EnableLobbyStatistics must be set to true, before this client connects. And the client has to be connected to the Master Server, which is providing the info about lobbies.

Implemented in **MonoBehaviourPunCallbacks**, **SupportLogger**, and **OnJoinedInstantiate**.

void OnRoomListUpdate (List< RoomInfo > **roomList)**

Called for any update of the room-listing while in a lobby (InLobby) on the Master Server.

Each item is a **RoomInfo** which might include custom properties (provided you defined those as lobby-listed when creating a room). Not all types of lobbies provide a listing of rooms to the client. Some are silent and specialized for server-side matchmaking.

Implemented in **MonoBehaviourPunCallbacks**, **SupportLogger**, and **OnJoinedInstantiate**.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	IMatchmakingCallbacks		

[Public Member Functions](#) | [List of all members](#)

IMatchmakingCallbacks Interface Reference

Callbacks

Collection of "organizational" callbacks for the **Realtime** Api to cover Matchmaking. [More...](#)

Inherited by **MonoBehaviourPunCallbacks**, **PhotonHandler**, **OnJoinedInstantiate**, **MatchMakingCallbacksContainer**, and **SupportLogger**.

Public Member Functions

void **OnFriendListUpdate** (List< **FriendInfo** > friendList)
Called when the server sent the response to a FindFriends request. [More...](#)

void **OnCreatedRoom** ()
Called when this client created a room and entered it. **OnJoinedRoom()** will be called as well. [More...](#)

void **OnCreateRoomFailed** (short returnCode, string message)
Called when the server couldn't create a room (OpCreateRoom failed). [More...](#)

void **OnJoinedRoom** ()
Called when the **LoadBalancingClient** entered a room, no matter if this client created it or simply joined. [More...](#)

void **OnJoinRoomFailed** (short returnCode, string message)
Called when a previous OpJoinRoom call failed on the server. [More...](#)

void **OnJoinRandomFailed** (short returnCode, string message)
Called when a previous OpJoinRandom call failed on the server. [More...](#)

void **OnLeftRoom** ()
Called when the local user/client left a room, so the game's logic can clean up it's internal state. [More...](#)

Detailed Description

Collection of "organizational" callbacks for the **Realtime** Api to cover Matchmaking.

Classes that implement this interface must be registered to get callbacks for various situations.

To register for callbacks, `PhotonNetwork.AddCallbackTarget(<Your Component implementing this interface>);` To stop getting callbacks, `PhotonNetwork.RemoveCallbackTarget(<Your Component implementing this interface>);`

You can also simply override `MonoBehaviourPunCallbacks` which will provide you with Magic Callbacks (like Unity would call `Start()`, `Update()` on a `MonoBehaviour`)

Member Function Documentation

void OnCreatedRoom ()

Called when this client created a room and entered it. **OnJoinedRoom()** will be called as well.

This callback is only called on the client which created a room (see `OpCreateRoom`).

As any client might close (or drop connection) anytime, there is a chance that the creator of a room does not execute `OnCreatedRoom`.

If you need specific room properties or a "start signal", implement `OnMasterClientSwitched()` and make each new `MasterClient` check the room's state.

Implemented in **MatchMakingCallbacksContainer**, **MonoBehaviourPunCallbacks**, **SupportLogger**, and **OnJoinedInstantiate**.

void OnCreateRoomFailed (short **returnCode, string **message**)**

Called when the server couldn't create a room (`OpCreateRoom` failed).

Creating a room may fail for various reasons. Most often, the room already exists (roomname in use) or the **RoomOptions** clash and it's impossible to create the room.

When creating a room fails on a Game Server: The client will cache the failure internally and returns to the Master Server before it calls

the fail-callback. This way, the client is ready to find/create a room at the moment of the callback. In this case, the client skips calling `OnConnectedToMaster` but returning to the Master Server will still call `OnConnected`. Treat callbacks of `OnConnected` as pure information that the client could connect.

Parameters

returnCode Operation ReturnCode from the server.

message Debug message for the error.

Implemented in **MatchMakingCallbacksContainer**, **MonoBehaviourPunCallbacks**, **SupportLogger**, and **OnJoinedInstantiate**.

void OnFriendListUpdate (List< FriendInfo > friendList)

Called when the server sent the response to a FindFriends request.

After calling `OpFindFriends`, the Master Server will cache the friend list and send updates to the friend list. The friends includes the name, `userId`, online state and the room (if any) for each requested user/friend.

Use the `friendList` to update your UI and store it, if the UI should highlight changes.

Implemented in **MatchMakingCallbacksContainer**, **MonoBehaviourPunCallbacks**, **SupportLogger**, and **OnJoinedInstantiate**.

void OnJoinedRoom ()

Called when the **LoadBalancingClient** entered a room, no matter if this client created it or simply joined.

When this is called, you can access the existing players in **Room.Players**, their custom properties and

Room.CustomProperties.

In this callback, you could create player objects. For example in Unity, instantiate a prefab for the player.

If you want a match to be started "actively", enable the user to signal "ready" (using `OpRaiseEvent` or a Custom Property).

Implemented in **MatchMakingCallbacksContainer**, **MonoBehaviourPunCallbacks**, **SupportLogger**, **PlayerNumbering**, **ConnectAndJoinRandom**, **PunTeams**, and **OnJoinedInstantiate**.

```
void OnJoinRandomFailed ( short returnCode,  
                        string message  
                        )
```

Called when a previous `OpJoinRandom` call failed on the server.

The most common causes are that a room is full or does not exist (due to someone else being faster or closing the room).

This operation is only ever sent to the Master Server. Once a room is found by the Master Server, the client will head off to the designated Game Server and use the operation `Join` on the Game Server.

When using multiple lobbies (via `OpJoinLobby` or a **TypedLobby** parameter), another lobby might have more/fitting rooms.

Parameters

returnCode Operation ReturnCode from the server.

message Debug message for the error.

Implemented in **MatchMakingCallbacksContainer**, **MonoBehaviourPunCallbacks**, **SupportLogger**, **OnJoinedInstantiate**, and **ConnectAndJoinRandom**.


```
void OnJoinRoomFailed ( short returnCode,  
                        string message  
                      )
```

Called when a previous OpJoinRoom call failed on the server.

Joining a room may fail for various reasons. Most often, the room is full or does not exist anymore (due to someone else being faster or closing the room).

When joining a room fails on a Game Server: The client will cache the failure internally and returns to the Master Server before it calls the fail-callback. This way, the client is ready to find/create a room at the moment of the callback. In this case, the client skips calling OnConnectedToMaster but returning to the Master Server will still call OnConnected. Treat callbacks of OnConnected as pure information that the client could connect.

Parameters

returnCode Operation ReturnCode from the server.

message Debug message for the error.

Implemented in **MatchMakingCallbacksContainer**, **MonoBehaviourPunCallbacks**, **SupportLogger**, and **OnJoinedInstantiate**.

```
void OnLeftRoom ( )
```

Called when the local user/client left a room, so the game's logic can clean up it's internal state.

When leaving a room, the **LoadBalancingClient** will disconnect the Game Server and connect to the Master Server. This wraps up multiple internal actions.

Wait for the callback OnConnectedToMaster, before you use lobbies and join or create rooms.

Implemented in **MatchMakingCallbacksContainer**,
MonoBehaviourPunCallbacks, **SupportLogger**,
OnJoinedInstantiate, **PlayerNumbering**, and **PunTeams**.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	IInRoomCallbacks		

[Public Member Functions](#) | [List of all members](#)

IInRoomCallbacks Interface Reference

Callbacks

Collection of "in room" callbacks for the **Realtime** Api to cover: Players entering or leaving, property updates and Master Client switching.
[More...](#)

Inherited by **MonoBehaviourPunCallbacks**, PhotonHandler, InRoomCallbacksContainer, and **SupportLogger**.

Public Member Functions

void **OnPlayerEnteredRoom** (**Player** newPlayer)

Called when a remote player entered the room. This **Player** is already added to the playerlist. [More...](#)

void **OnPlayerLeftRoom** (**Player** otherPlayer)

Called when a remote player left the room or became inactive. Check otherPlayer.IsInactive. [More...](#)

void **OnRoomPropertiesUpdate** (Hashtable propertiesThatChanged)

Called when a room's custom properties changed. The propertiesThatChanged contains all that was set via **Room.SetCustomProperties**. [More...](#)

void **OnPlayerPropertiesUpdate** (**Player** targetPlayer, Hashtable changedProps)

Called when custom player-properties are changed. **Player** and the changed properties are passed as object[]. [More...](#)

void **OnMasterClientSwitched** (**Player** newMasterClient)

Called after switching to a new MasterClient when the current one leaves. [More...](#)

Detailed Description

Collection of "in room" callbacks for the **Realtime** Api to cover: Players entering or leaving, property updates and Master Client switching.

The callback to get events is in a separate interface:
IOnEventCallback.

To register for callbacks, PhotonNetwork.AddCallbackTarget(<Your Component implementing this interface>); To stop getting callbacks, PhotonNetwork.RemoveCallbackTarget(<Your Component implementing this interface>);

You can also simply override MonoBehaviourPunCallbacks which will provide you with Magic Callbacks (like Unity would call Start(), Update() on a MonoBehaviour)

Member Function Documentation

void OnMasterClientSwitched (Player **newMasterClient)**

Called after switching to a new MasterClient when the current one leaves.

This is not called when this client enters a room. The former MasterClient is still in the player list when this method get called.

Implemented in **MonoBehaviourPunCallbacks**, and **SupportLogger**.

void OnPlayerEnteredRoom (Player **newPlayer)**

Called when a remote player entered the room. This **Player** is already added to the playerlist.

If your game starts with a certain number of players, this callback can be useful to check the Room.playerCount and find out if you can start.

Implemented in **MonoBehaviourPunCallbacks**, **SupportLogger**, **PlayerNumbering**, and **PunTeams**.

void OnPlayerLeftRoom (Player **otherPlayer)**

Called when a remote player left the room or became inactive. Check otherPlayer.IsInactive.

If another player leaves the room or if the server detects a lost connection, this callback will be used to notify your game logic.

Depending on the room's setup, players may become inactive, which means they may return and retake their spot in the room. In such cases, the **Player** stays in the **Room.Players** dictionary.

If the player is not just inactive, it gets removed from the **Room.Players** dictionary, before the callback is called.

Implemented in **MonoBehaviourPunCallbacks**, **SupportLogger**, **PlayerNumbering**, and **PunTeams**.

```
void OnPlayerPropertiesUpdate ( Player      targetPlayer,  
                             Hashtable changedProps  
                             )
```

Called when custom player-properties are changed. **Player** and the changed properties are passed as object[].

Changing properties must be done by **Player.SetCustomProperties**, which causes this callback locally, too.

Parameters

targetPlayer Contains **Player** that changed.

changedProps Contains the properties that changed.

Implemented in **MonoBehaviourPunCallbacks**, **SupportLogger**, **PlayerNumbering**, and **PunTeams**.

```
void  
OnRoomPropertiesUpdate ( Hashtable propertiesThatChanged )
```

Called when a room's custom properties changed. The **propertiesThatChanged** contains all that was set via **Room.SetCustomProperties**.

Since v1.25 this method has one parameter: Hashtable **propertiesThatChanged**.

Changing properties must be done by **Room.SetCustomProperties**, which causes this callback locally, too.

Parameters

propertiesThatChanged

Implemented in **MonoBehaviourPunCallbacks**, **PunTurnManager**, **SupportLogger**, and **CountdownTimer**.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	IOnEventCallback		

[Public Member Functions](#) | [List of all members](#)

IOnEventCallback Interface Reference

Callbacks

Event callback for the **Realtime** Api. Covers events from the server and those sent by clients via `OpRaiseEvent`. [More...](#)

Inherited by **PunTurnManager**.

Public Member Functions

void **OnEvent** (EventData photonEvent)
Called for any incoming events. [More...](#)

Detailed Description

Event callback for the **Realtime** Api. Covers events from the server and those sent by clients via OpRaiseEvent.

Classes that implement this interface must be registered to get callbacks for various situations.

To register for callbacks, register the instance via:
LoadBalancingClient.EventReceived += instance. To stop getting callbacks, remove the instance via: -=.

Member Function Documentation

void OnEvent (EventArgs photonEvent)

Called for any incoming events.

To receive events, implement **IOnEventCallback** in any class and register it via **AddCallbackTarget** (either in **LoadBalancingClient** or **PhotonNetwork**).

With the **EventArgs.Sender** you can look up the **Player** who sent the event.

It is best practice to assign an **eventCode** for each different type of content and action, so the **Code** will be essential to read the incoming events.

Implemented in **PunTurnManager**.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	IWebRpcCallback		

[Public Member Functions](#) | [List of all members](#)

IWebRpcCallback Interface Reference

Callbacks

Interface for "WebRpc" callbacks for the **Realtime** Api. Currently includes only responses for Web RPCs. [More...](#)

Inherited by WebRpcCallbacksContainer.

Public Member Functions

void **OnWebRpcResponse** (OperationResponse response)
Called by PUN when the response to a WebRPC is available.
See PhotonNetwork.WebRPC. [More...](#)

Detailed Description

Interface for "WebRpc" callbacks for the **Realtime** Api. Currently includes only responses for Web RPCs.

Classes that implement this interface must be registered to get callbacks for various situations.

To register for callbacks, use the `LoadBalancingClient.WebRpcCallbackTargets` and `Add()` the instance. To stop getting callbacks, `Remove()` the instance.

Member Function Documentation

void OnWebRpcResponse (OperationResponse **response)**

Called by PUN when the response to a WebRPC is available. See PhotonNetwork.WebRPC.

Important: The response.ReturnCode is 0 if **Photon** was able to reach your web-service.

The content of the response is what your web-service sent. You can create a **WebRpcResponse** from it.

Example: **WebRpcResponse** webResponse = new WebRpcResponse(operationResponse);

Please note: Class OperationResponse is in a namespace which needs to be "used":

using ExitGames.Client.Photon; // includes OperationResponse (and other classes)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	IPunObservable		

[Public Member Functions](#) | [List of all members](#)

IPunObservable Interface Reference

Callbacks

Defines the OnPhotonSerializeView method to make it easy to implement correctly for observable scripts. [More...](#)

Inherited by **PhotonAnimatorView**, **PhotonRigidbody2DView**, **PhotonRigidbodyView**, **PhotonTransformView**, **PhotonTransformViewClassic**, **CullingHandler**, and **SmoothSyncMovement**.

Public Member Functions

void **OnPhotonSerializeView** (**PhotonStream** stream,
PhotonMessageInfo info)

Called by PUN several times per second, so that your script can write and read synchronization data for the **PhotonView**.
[More...](#)

Detailed Description

Defines the OnPhotonSerializeView method to make it easy to implement correctly for observable scripts.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	IPunOwnershipCallbacks		

[Public Member Functions](#) | [List of all members](#)

IPunOwnershipCallbacks Interface Reference

Callbacks

This interface is used as definition of all callback methods of PUN, except OnPhotonSerializeView. Preferably, implement them individually. [More...](#)

Public Member Functions

void **OnOwnershipRequest** (**PhotonView** targetView, **Player** requestingPlayer)

Called when another player requests ownership of a **PhotonView** from you (the current owner). [More...](#)

void **OnOwnershipTransferred** (**PhotonView** targetView, **Player** previousOwner)

Called when ownership of a **PhotonView** is transferred to another player. [More...](#)

Detailed Description

This interface is used as definition of all callback methods of PUN, except `OnPhotonSerializeView`. Preferably, implement them individually.

This interface is available for completeness, more than for actually implementing it in a game. You can implement each method individually in any `MonoBehaviour`, without implementing `IPunCallbacks`.

PUN calls all callbacks by name. Don't use implement callbacks with fully qualified name. Example: `IPunCallbacks.OnConnected` won't get called by Unity's `SendMessage()`.

PUN will call these methods on any script that implements them, analog to Unity's events and callbacks. The situation that triggers the call is described per method.

`OnPhotonSerializeView` is NOT called like these callbacks! It's usage frequency is much higher and it is implemented in: **`IPunObservable`**.

Member Function Documentation

```
void OnOwnershipRequest ( PhotonView targetView,  
                          Player      requestingPlayer  
                          )
```

Called when another player requests ownership of a **PhotonView** from you (the current owner).

The parameter viewAndPlayer contains:

PhotonView view = viewAndPlayer[0] as **PhotonView**;

Player requestingPlayer = viewAndPlayer[1] as Player;

Parameters

targetView **PhotonView** for which ownership gets requested.

requestingPlayer Player who requests ownership.

```
void OnOwnershipTransferred ( PhotonView targetView,  
                              Player      previousOwner  
                              )
```

Called when ownership of a **PhotonView** is transferred to another player.

The parameter viewAndPlayers contains:

PhotonView view = viewAndPlayers[0] as **PhotonView**;

Player newOwner = viewAndPlayers[1] as Player;

Player oldOwner = viewAndPlayers[2] as Player;

```
void OnOwnershipTransferred(object[] viewAndPlayers) {} //
```

Parameters

targetView **PhotonView** for which ownership changed.
previousOwner Player who was the previous owner (or null, if none).



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	IPunInstantiateMagicCallback		

[Public Member Functions](#) | [List of all members](#)

IPunInstantiateMagicCallback Interface Reference

Callbacks

Public Member Functions

void **OnPhotonInstantiate** (PhotonMessageInfo info)

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	MonoBehaviourPunCallbacks		

[Public Member Functions](#) | [List of all members](#)

MonoBehaviourPunCallbacks Class Reference

Callbacks

This class provides a .photonView and all callbacks/events that PUN can call. Override the events/methods you want to use. [More...](#)

Inherits **MonoBehaviourPun**, **IConnectionCallbacks**, **IMatchmakingCallbacks**, **IInRoomCallbacks**, and **ILobbyCallbacks**.

Inherited by **ConnectAndJoinRandom**, **CountdownTimer**, **PlayerNumbering**, **PunTeams**, and **PunTurnManager**.

Public Member Functions

virtual void **OnEnable** ()

virtual void **OnDisable** ()

virtual void **OnConnected** ()

Called to signal that the raw connection got established but before the client can call operation on the server.

[More...](#)

virtual void **OnLeftRoom** ()

Called when the local user/client left a room, so the game's logic can clean up it's internal state. [More...](#)

virtual void **OnMasterClientSwitched** (**Player** newMasterClient)

Called after switching to a new MasterClient when the current one leaves. [More...](#)

virtual void **OnCreateRoomFailed** (short returnCode, string message)

Called when the server couldn't create a room (OpCreateRoom failed). [More...](#)

virtual void **OnJoinRoomFailed** (short returnCode, string message)

Called when a previous OpJoinRoom call failed on the server. [More...](#)

virtual void **OnCreatedRoom** ()

Called when this client created a room and entered it. **OnJoinedRoom()** will be called as well. [More...](#)

virtual void **OnJoinedLobby** ()

Called on entering a lobby on the Master Server. The actual room-list updates will call OnRoomListUpdate. [More...](#)

virtual void **OnLeftLobby ()**
Called after leaving a lobby. [More...](#)

virtual void **OnDisconnected (DisconnectCause cause)**
Called after disconnecting from the **Photon** server. It could be a failure or intentional [More...](#)

virtual void **OnRegionListReceived (RegionHandler regionHandler)**
Called when the Name Server provided a list of regions for your title. [More...](#)

virtual void **OnRoomListUpdate (List< RoomInfo > roomList)**
Called for any update of the room-listing while in a lobby (InLobby) on the Master Server. [More...](#)

virtual void **OnJoinedRoom ()**
Called when the LoadBalancingClient entered a room, no matter if this client created it or simply joined. [More...](#)

virtual void **OnPlayerEnteredRoom (Player newPlayer)**
Called when a remote player entered the room. This Player is already added to the playerlist. [More...](#)

virtual void **OnPlayerLeftRoom (Player otherPlayer)**
Called when a remote player left the room or became inactive. Check otherPlayer.IsInactive. [More...](#)

virtual void **OnJoinRandomFailed** (short returnCode, string message)
Called when a previous OpJoinRandom call failed on the server. [More...](#)

virtual void **OnConnectedToMaster ()**
Called when the client is connected to the Master Server and ready for matchmaking and other tasks. [More...](#)

virtual void **OnRoomPropertiesUpdate** (Hashtable propertiesThatChanged)
Called when a room's custom properties changed. The propertiesThatChanged contains all that was set via **Room.SetCustomProperties**. [More...](#)

virtual void **OnPlayerPropertiesUpdate** (Player target, Hashtable changedProps)
Called when custom player-properties are changed. Player and the changed properties are passed as object[]. [More...](#)

virtual void **OnFriendListUpdate** (List< FriendInfo > friendList)
Called when the server sent the response to a FindFriends request. [More...](#)

virtual void **OnCustomAuthenticationResponse** (Dictionary< string, object > data)
Called when your Custom Authentication service responds with additional data. [More...](#)

virtual void **OnCustomAuthenticationFailed** (string debugMessage)
Called when the custom authentication failed. Followed by disconnect! [More...](#)

virtual void **OnWebRpcResponse** (OperationResponse response)

virtual void **OnLobbyStatisticsUpdate** (List< TypedLobbyInfo > lobbyStatistics)
Called when the Master Server sent an update for the Lobby Statistics, updating PhotonNetwork.LobbyStatistics. [More...](#)

Additional Inherited Members

‣ Properties inherited from **MonoBehaviourPun**

PhotonView **photonView** [get]

A cached reference to a **PhotonView** on this
GameObject. [More...](#)

Detailed Description

This class provides a .photonView and all callbacks/events that PUN can call. Override the events/methods you want to use.

By extending this class, you can implement individual methods as override.

Visual Studio and MonoDevelop should provide the list of methods when you begin typing "override". **Your implementation does not have to call "base.method()".**

This class implements all callback interfaces and extends **Photon.Pun.MonoBehaviourPun**.

Member Function Documentation

virtual void OnConnected ()

virtual

Called to signal that the raw connection got established but before the client can call operation on the server.

After the (low level transport) connection is established, the client will automatically send the Authentication operation, which needs to get a response before the client can call other operations.

Your logic should wait for either: OnRegionListReceived or OnConnectedToMaster.

This callback is useful to detect if the server can be reached at all (technically). Most often, it's enough to implement **OnDisconnected()**.

This is not called for transitions from the masterserver to game servers.

Implements **IConnectionCallbacks**.

virtual void OnConnectedToMaster ()

virtual

Called when the client is connected to the Master Server and ready for matchmaking and other tasks.

The list of available rooms won't become available unless you join a lobby via **LoadBalancingClient.OpJoinLobby**. You can join rooms and create them even without being in a lobby. The default lobby is used in that case.

Implements **IConnectionCallbacks**.

Reimplemented in **ConnectAndJoinRandom**.

virtual void OnCreatedRoom ()

virtual

Called when this client created a room and entered it.
OnJoinedRoom() will be called as well.

This callback is only called on the client which created a room (see **OpCreateRoom**).

As any client might close (or drop connection) anytime, there is a chance that the creator of a room does not execute **OnCreatedRoom**.

If you need specific room properties or a "start signal", implement **OnMasterClientSwitched()** and make each new **MasterClient** check the room's state.

Implements **IMatchmakingCallbacks**.

virtual void OnCreateRoomFailed (short **returnCode,
string **message**
)**

virtual

Called when the server couldn't create a room (**OpCreateRoom** failed).

The most common cause to fail creating a room, is when a title relies on fixed room-names and the room already exists.

Parameters

returnCode Operation ReturnCode from the server.

message Debug message for the error.

Implements **IMatchmakingCallbacks**.

virtual void

OnCustomAuthenticationFailed (string **debugMessage**) virtual

Called when the custom authentication failed. Followed by disconnect!

Custom Authentication can fail due to user-input, bad tokens/secrets. If authentication is successful, this method is not called. Implement **OnJoinedLobby()** or **OnConnectedToMaster()** (as usual).

During development of a game, it might also fail due to wrong configuration on the server side. In those cases, logging the debugMessage is very important.

Unless you setup a custom authentication service for your app (in the [Dashboard](#)), this won't be called!

Parameters

debugMessage Contains a debug message why authentication failed. This has to be fixed during development.

Implements **ICollectionCallbacks**.

virtual void

OnCustomAuthenticationResponse (Dictionary< string, object > c

Called when your Custom Authentication service responds with addition

Custom Authentication services can include some custom data in their response. When present, that data is made available in this callback as Dictionary<string, object>. Keys of your data have to be strings, the values can be either string or a number (in Json). You need to make extra sure, that the value type is the one you expect. Numbers become (currently) int64.

Example: void OnCustomAuthenticationResponse(Dictionary<string, object> data)
{ ... }

<https://doc.photonengine.com/en-us/realtime/current/reference/custom-authentication>

Implements **ICollectionCallbacks**.

virtual void OnDisconnected (DisconnectCause cause)

virtual

Called after disconnecting from the **Photon** server. It could be a failure or intentional

The reason for this disconnect is provided as DisconnectCause.

Implements **ICollectionCallbacks**.

Reimplemented in **ConnectAndJoinRandom**.

**virtual void
OnFriendListUpdate**

(List< FriendInfo > friendList)

virtual

Called when the server sent the response to a FindFriends request.

After calling OpFindFriends, the Master Server will cache the friend list and send updates to the friend list. The friends includes the name, userId, online state and the room (if any) for each requested user/friend.

Use the friendList to update your UI and store it, if the UI should highlight changes.

Implements **IMatchmakingCallbacks**.

virtual void OnJoinedLobby ()

virtual

Called on entering a lobby on the Master Server. The actual room-list updates will call OnRoomListUpdate.

While in the lobby, the roomlist is automatically updated in fixed intervals (which you can't modify in the public cloud). The room list gets available via `OnRoomListUpdate`.

Implements **`ILobbyCallbacks`**.

Reimplemented in **`ConnectAndJoinRandom`**.

virtual void OnJoinedRoom ()

virtual

Called when the `LoadBalancingClient` entered a room, no matter if this client created it or simply joined.

When this is called, you can access the existing players in **`Room.Players`**, their custom properties and **`Room.CustomProperties`**.

In this callback, you could create player objects. For example in Unity, instantiate a prefab for the player.

If you want a match to be started "actively", enable the user to signal "ready" (using `OpRaiseEvent` or a Custom Property).

Implements **`IMatchmakingCallbacks`**.

Reimplemented in **`PlayerNumbering`**, **`ConnectAndJoinRandom`**, and **`PunTeams`**.

virtual void OnJoinRandomFailed (short `returnCode`, string `message`)

virtual

Called when a previous `OpJoinRandom` call failed on the server.

The most common causes are that a room is full or does not exist (due to someone else being faster or closing the room).

When using multiple lobbies (via `OpJoinLobby` or a `TypedLobby`

parameter), another lobby might have more/fitting rooms.

Parameters

returnCode Operation ReturnCode from the server.

message Debug message for the error.

Implements **IMatchmakingCallbacks**.

Reimplemented in **ConnectAndJoinRandom**.

```
virtual void OnJoinRoomFailed ( short returnCode,  
                                string message  
                                )
```

virtual

Called when a previous OpJoinRoom call failed on the server.

The most common causes are that a room is full or does not exist (due to someone else being faster or closing the room).

Parameters

returnCode Operation ReturnCode from the server.

message Debug message for the error.

Implements **IMatchmakingCallbacks**.

```
virtual void OnLeftLobby ( )
```

virtual

Called after leaving a lobby.

When you leave a lobby, OpCreateRoom and OpJoinRandomRoom automatically refer to the default lobby.

Implements **ILobbyCallbacks**.

```
virtual void OnLeftRoom ( )
```

virtual

Called when the local user/client left a room, so the game's logic can clean up it's internal state.

When leaving a room, the LoadBalancingClient will disconnect the Game Server and connect to the Master Server. This wraps up multiple internal actions.

Wait for the callback OnConnectedToMaster, before you use lobbies and join or create rooms.

Implements **IMatchmakingCallbacks**.

Reimplemented in **PlayerNumbering**, and **PunTeams**.

virtual void

OnLobbyStatisticsUpdate (List< TypedLobbyInfo > lobbyStatistics

Called when the Master Server sent an update for the Lobby Statistics, updating PhotonNetwork.LobbyStatistics.

This callback has two preconditions: EnableLobbyStatistics must be set before this client connects. And the client has to be connected to the Master Server, which is providing the info about lobbies.

Implements **ILobbyCallbacks**.

virtual void

OnMasterClientSwitched (Player newMasterClient) virtual

Called after switching to a new MasterClient when the current one leaves.

This is not called when this client enters a room. The former MasterClient is still in the player list when this method get called.

Implements **IInRoomCallbacks**.

virtual void OnPlayerEnteredRoom (Player **newPlayer)**

virtual

Called when a remote player entered the room. This Player is already added to the playerlist.

If your game starts with a certain number of players, this callback can be useful to check the Room.playerCount and find out if you can start.

Implements **IInRoomCallbacks**.

Reimplemented in **PlayerNumbering**, and **PunTeams**.

virtual void OnPlayerLeftRoom (Player **otherPlayer)**

virtual

Called when a remote player left the room or became inactive. Check otherPlayer.IsInactive.

If another player leaves the room or if the server detects a lost connection, this callback will be used to notify your game logic.

Depending on the room's setup, players may become inactive, which means they may return and retake their spot in the room. In such cases, the Player stays in the **Room.Players** dictionary.

If the player is not just inactive, it gets removed from the **Room.Players** dictionary, before the callback is called.

Implements **IInRoomCallbacks**.

Reimplemented in **PlayerNumbering**, and **PunTeams**.

**virtual void
OnPlayerPropertiesUpdate (Player **target**,
Hashtable **changedProps**
)**

virtual

Called when custom player-properties are changed. Player and the changed properties are passed as object[].

Changing properties must be done by **Player.SetCustomProperties**, which causes this callback locally, too.

Parameters

targetPlayer Contains Player that changed.

changedProps Contains the properties that changed.

Implements **INRoomCallbacks**.

Reimplemented in **PlayerNumbering**, and **PunTeams**.

virtual void
OnRegionListReceived (**RegionHandler** **regionHandler**) virtual

Called when the Name Server provided a list of regions for your title.

Check the RegionHandler class description, to make use of the provided values.

Parameters

regionHandler The currently used RegionHandler.

Implements **IConnectionCallbacks**.

virtual void
OnRoomListUpdate (**List< RoomInfo >** **roomList**) virtual

Called for any update of the room-listing while in a lobby (InLobby) on the Master Server.

Each item is a RoomInfo which might include custom properties (provided you defined those as lobby-listed when creating a room).

Not all types of lobbies provide a listing of rooms to the client. Some are silent and specialized for server-side matchmaking.

Implements **ILobbyCallbacks**.

virtual void

OnRoomPropertiesUpdate (Hashtable **propertiesThatChanged)**

Called when a room's custom properties changed. The **propertiesThatChanged** contains all that was set via **Room.SetCustomProperties**.

Since v1.25 this method has one parameter: Hashtable **propertiesThatChanged**.

Changing properties must be done by **Room.SetCustomProperties**, which causes this callback locally, too.

Parameters

propertiesThatChanged

Implements **IInRoomCallbacks**.

Reimplemented in **PunTurnManager**, and **CountdownTimer**.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Package Functions				
All	Enumerations			

Here is a list of all documented namespace members with links to the namespaces they belong to:

- AuthModeOption : **Photon.Realtime**
- ChatDisconnectCause : **Photon.Chat**
- ChatState : **Photon.Chat**
- ClientState : **Photon.Realtime**
- ConnectMethod : **Photon.Pun**
- CustomAuthenticationType : **Photon.Chat** , **Photon.Realtime**
- DisconnectCause : **Photon.Realtime**
- EncryptionMode : **Photon.Realtime**
- EventCaching : **Photon.Realtime**
- JoinMode : **Photon.Realtime**
- LobbyType : **Photon.Realtime**
- MatchmakingMode : **Photon.Realtime**
- OwnershipOption : **Photon.Pun**
- PropertyTypeFlag : **Photon.Realtime**
- PunLogLevel : **Photon.Pun**
- ReceiverGroup : **Photon.Realtime**
- RpcTarget : **Photon.Pun**
- ServerConnection : **Photon.Realtime**



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Package Functions				
All	Enumerations			

- AuthModeOption : **Photon.Realtime**
- ChatDisconnectCause : **Photon.Chat**
- ChatState : **Photon.Chat**
- ClientState : **Photon.Realtime**
- ConnectMethod : **Photon.Pun**
- CustomAuthenticationType : **Photon.Chat** , **Photon.Realtime**
- DisconnectCause : **Photon.Realtime**
- EncryptionMode : **Photon.Realtime**
- EventCaching : **Photon.Realtime**
- JoinMode : **Photon.Realtime**
- LobbyType : **Photon.Realtime**
- MatchmakingMode : **Photon.Realtime**
- OwnershipOption : **Photon.Pun**
- PropertyTypeFlag : **Photon.Realtime**
- PunLogLevel : **Photon.Pun**
- ReceiverGroup : **Photon.Realtime**
- RpcTarget : **Photon.Pun**
- ServerConnection : **Photon.Realtime**



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

[detail level 1 2 3 4 5]	
▼ Photon	
▼ Chat	
AuthenticationValues	Container for user authentication values. You must set these values before you can use the chat system.
ChannelCreationOptions	
ChannelWellKnownProperties	
ChatChannel	A channel of communication. It is updated by ChatClient ONLY.
ChatClient	Central class of the Photon chat system. It handles channels and manages the chat system.
ChatEventCode	Wraps up internally used event codes. You don't have to use them directly.
ChatOperationCode	Wraps up codes for operations. You don't have to use them directly.
ChatParameterCode	Wraps up codes for parameters (used internally) used internally. You don't have to use them directly.
ChatPeer	Provides basic operation for the chat system. This internal class is used by ChatClient .
ChatUserStatus	Contains commonly used user status values.

🔗 CullingHandler

🔗 EventSystemSpawner

🔗 GraphicToggleIsOnTransition

🔗 IPunTurnManagerCallbacks

🔗 MoveByKeys

🔗 OnClickDestroy

🔗 OnClickInstantiate

🔗 OnClickRpc

🔗 OnEscapeQuit

🔗 OnJoinedInstantiate

🔗 OnPointerOverTooltip

🔗 OnStartDelete

🔗 PhotonLagSimulationGui

🔗 PhotonStatsGui

🔗 PlayerNumbering

Handles the network

Event system spawner
GameObject with an
StandaloneInputModule
additive scene loading
otherwise get a "Multi
is not supported" error
Use this on toggles to
transition on the text

Very basic component
WASD and Space.

Destroys the network
PhotonNetwork.Destroy
which calls Object.Destroy

Instantiates a network

This component will in
GameObject when in
that component's Gar
PhysicsRaycaster for

This component will c
escape key is pressed

This component will in
GameObject when a

Set focus to a given p
over

This component will c
attached to (in Start())

This MonoBehaviour
client's network-simul
(fixed delay), jitter (ra

Basic GUI to show tra
connection to **Photon**

Implements consist
with help of room proj

🔗 **PlayerNumberingExtensions**

🔗 **PointedAtGameObjectInfo**

🔗 **PunPlayerScores**

🔗 **PunTeams**

🔗 **PunTurnManager**

🔗 **ScoreExtensions**

🔗 **SmoothSyncMovement**

🔗 **StatesGui**

▶ 🔗 **TabViewManager**

🔗 **Tab**

🔗 **TabChangeEvent**

🔗 **TeamExtensions**

🔗 **TextButtonTransition**

🔗 **TextToggleIsOnTransition**

🔗 **TurnExtensions**

🔗 **DefaultPool**

Player.GetPlayerNum
Extension used for **Pl** class.

Display ViewId, **Own**
IsMine when clicked.

Scoring system for Pl
Implements teams in
player properties. Acc
extension.

Pun turnBased Game
Interface (**IPunTurnM**
typical turn flow and l
Extensions for Player
feature dedicated api















Smoothed out moven
Output detailed inform
states, using the old l
Tab view manager. H
and deactivation, and
Callback when a tab \

















Tab change event.

















Extension used for **Pl**
Wraps access to the j
Use this on Button tex
transition on the text a
button's behaviour.
















Use this on toggles te
transition on the text c







The default implemen
which actually Instant
GameObjects but poc

 InstantiateParameters	
 IPunInstantiateMagicCallback	
 IPunObservable	Defines the OnPhoton interface to make it easy to implement callbacks.
 IPunOwnershipCallbacks	This interface is used for the ownership methods of PUN, except for IPunObservable . Preferably, implement IPunObservable .
 IPunPrefabPool	Defines an interface for the PhotonNetwork.Instantiate method. See PhotonNetwork.Deserialize .
 MonoBehaviourPun	This class adds the IPunObservable interface and logging a warning when the networkView is null.
 MonoBehaviourPunCallbacks	This class provides a set of callbacks/events that you can implement. See the events/methods you can implement.
▶  PhotonAnimatorView	This class helps you to manage animations. Simply add it to your GameObject and make it a PhotonAnimatorView component. It is an observed component.
 PhotonMessageInfo	Container class for incoming RPC or update.
 PhotonNetwork	The main class to use for Photon. This class is static.
 PhotonRigidbody2DView	
 PhotonRigidbodyView	
 PhotonStream	This container is used to store data. You can either provide incoming data or you can provide it.
 PhotonStreamQueue	The PhotonStreamQueue stores states at higher frequency. See PhotonNetwork.Send for all those states at once. On the receiving end, see PhotonNetwork.Receive .

	then the stream will re in the same order and in.
 PhotonTransformView	
 PhotonTransformViewClassic	This class helps you t and scale of a GameC different options to m appear smooth, even couple of times per se component to your G the PhotonTransform list of observed comp
 PhotonTransformViewPositionControl	
 PhotonTransformViewPositionModel	
 PhotonTransformViewRotationControl	
 PhotonTransformViewRotationModel	
 PhotonTransformViewScaleControl	
 PhotonTransformViewScaleModel	
 PhotonView	A PhotonView identif network (viewID) and client updates remote
 PunExtensions	Small number of exte easier for PUN to wor
 PunRPC	Replacement for RPC Used to flag methods
 SceneManagerHelper	
 ServerSettings	Collection of connecti internally by PhotonNetwork.Con
▼  Realtime	
 ActorProperties	Class for constants. T known" properties for
 AppSettings	Settings for Photon a connect to.

 AuthenticationValues	Container for user authentication values before you
 ConnectionCallbacksContainer	Container type for call IConnectionCallback LoadBalancingCallba
 ConnectionHandler	
 EncryptionDataParameters	
 EnterRoomParams	
 ErrorCode	ErrorCode defines the Photon client/server
 EventCode	Class for constants. T defined by Photon Lc
 EventExt	
 Extensions	This static class defines methods for several e float and others).
 FriendInfo	Used to store info about which room he/she is
 GamePropertyKey	Class for constants. T known" room/game p Loadbalancing.
 IConnectionCallbacks	Collection of "organiz Realtime Api to cover
 IInRoomCallbacks	Collection of "in room to cover: Players ente updates and Master C
 ILobbyCallbacks	Collection of "organiz Realtime Api to cover
 IMatchmakingCallbacks	Collection of "organiz Realtime Api to cover
 IOnEventCallback	Event callback for the from the server and th OpRaiseEvent.
 IWebRpcCallback	Interface for "WebRpc Api. Currently include

	RPCs.
 LoadBalancingClient	This class implements workflow by using a LoadBalancingClient state and will automatically connect between the Master and the server application when the application is in the Cloud.
 LoadBalancingPeer	A LoadbalancingPeer enum definitions need to be defined in the server application when the application is in the Cloud.
 MatchMakingCallbacksContainer	Container type for callbacks. It implements the IMatchmakingCallback interface. MatchMakingCallbackContainer uses these constants.
 OperationCode	Class for constants. It uses these constants.
 OpJoinRandomRoomParams	
 ParameterCode	Class for constants. It uses these constants.
 PhotonPing	
 PingMono	Uses C# Socket class (Unity usually does).
 Player	Summarizes a "player" that room) by ID (or "name").
 RaiseEventOptions	Aggregates several local operation RaiseEvent usage details.
 Region	
 RegionHandler	Provides methods to connect to the (Photon Cloud) and to get the best ping.
 RegionPinger	
 Room	This class represents a room.
 RoomInfo	A simplified room with properties that are not set (e.g. name and join, used for the room info properties are not set etc).

 RoomOptions	Wraps up common room options you create rooms. Read more details.
 SupportLogger	Helper class to debug Photon client and vitas.
 TypedLobby	Refers to a specific lobby.
 TypedLobbyInfo	
 WebFlags	Optional flags to be used with Op RaiseEvent and C mainly for webhooks forwarded HTTP requests.
 WebRpcResponse	Reads an operation response and provides convenient access.



Photon Unity Networking 2 2.12

[Main Page](#)[Related Pages](#)[Modules](#)[Classes](#)[Package Functions](#)[Namespaces](#)

Photon Namespace Reference

Namespaces

namespace	Chat
-----------	-------------

namespace	Pun
-----------	------------

namespace	Realtime
-----------	-----------------

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

[Main Page](#)[Related Pages](#)[Modules](#)[Classes](#)[Package Functions](#)[Photon](#)[Chat](#)[Classes](#) | [Enumerations](#)

Photon.Chat Namespace Reference

Classes

class **AuthenticationValues**
Container for user authentication in **Photon**. Set AuthValues before you connect - all else is handled. [More...](#)

class **ChannelCreationOptions**

class **ChannelWellKnownProperties**

class **ChatChannel**
A channel of communication in **Photon Chat**, updated by **ChatClient** and provided as READ ONLY. [More...](#)

class **ChatClient**
Central class of the **Photon Chat** API to connect, handle channels and messages. [More...](#)

class **ChatEventCode**
Wraps up internally used constants in **Photon Chat** events. You don't have to use them directly usually. [More...](#)

class **ChatOperationCode**
Wraps up codes for operations used internally in **Photon Chat**. You don't have to use them directly usually. [More...](#)

class **ChatParameterCode**
Wraps up codes for parameters (in operations and events) used internally in **Photon Chat**. You don't have to use them directly usually. [More...](#)

class **ChatPeer**
Provides basic operations of the **Photon Chat** server. This internal class is used by public **ChatClient**. [More...](#)

class **ChatUserStatus**

Contains commonly used status values for SetOnlineStatus. You can define your own. [More...](#)

class **ErrorCode**

ErrorCode defines the default codes associated with **Photon** client/server communication. [More...](#)

interface **IChatClientListener**

Callback interface for **Chat** client side. Contains callback methods to notify your app about updates. Must be provided to new **ChatClient** in constructor [More...](#)

class **ParameterCode**

Class for constants. Codes for parameters of Operations and Events. [More...](#)

Enumerations

enum **ChatDisconnectCause**
Enumeration of causes for Disconnects (used in ChatClient.DisconnectedCause). [More...](#)

enum **CustomAuthenticationType** : byte
Options for optional "Custom Authentication" services used with **Photon**. Used by OpAuthenticate after connecting to **Photon**. [More...](#)

enum **ChatState**
Possible states for a Chat Client. [More...](#)

Enumeration Type Documentation

enum ChatDisconnectCause

strong

Enumeration of causes for Disconnects (used in **ChatClient.DisconnectedCause**).

Read the individual descriptions to find out what to do about this type of disconnect.

Enumerator	
None	No error was tracked.
DisconnectByServerUserLimit	OnStatusChanged: The CCUs count of your Photon Server License is exhausted (temporarily).
ExceptionOnConnect	OnStatusChanged: The server is not available or the address is wrong. Make sure the port is provided and the server is up.
DisconnectByServer	OnStatusChanged: The server disconnected this client. Most likely the server's send buffer is full (receiving too much from other clients).
TimeoutDisconnect	OnStatusChanged: This client detected that the server's responses are not received in due time. Maybe you send / receive too much?
Exception	OnStatusChanged: Some

	internal exception caused the socket code to fail. Contact Exit Games.
InvalidAuthentication	OnOperationResponse: Authenticate in the Photon Cloud with invalid Appld. Update your subscription or contact Exit Games.
MaxCcuReached	OnOperationResponse: Authenticate (temporarily) failed when using a Photon Cloud subscription without CCU Burst. Update your subscription.
InvalidRegion	OnOperationResponse: Authenticate when the app's Photon Cloud subscription is locked to some (other) region(s). Update your subscription or change region.
OperationNotAllowedInCurrentState	OnOperationResponse: Operation that's (currently) not available for this client (not authorized usually). Only tracked for op Authenticate.
CustomAuthenticationFailed	OnOperationResponse: Authenticate in the Photon Cloud with invalid client values or custom authentication setup in Cloud Dashboard.

enum ChatState

strong

Possible states for a **Chat** Client.

Enumerator	
Uninitialized	Peer is created but not used yet.
ConnectingToNameServer	Connecting to name server.
ConnectedToNameServer	Connected to name server.
Authenticating	Authenticating on current server.
Authenticated	Finished authentication on current server.
DisconnectingFromNameServer	Disconnecting from name server. This is usually a transition from name server to frontend server.
ConnectingToFrontEnd	Connecting to frontend server.
ConnectedToFrontEnd	Connected to frontend server.
DisconnectingFromFrontEnd	Disconnecting from frontend server.
QueuedComingFromFrontEnd	Currently not used.
Disconnecting	The client disconnects (from any server).
Disconnected	The client is no longer connected (to any server).

enum CustomAuthenticationType : byte

strong

Options for optional "Custom Authentication" services used with **Photon**. Used by OpAuthenticate after connecting to **Photon**.

Enumerator	
Custom	Use a custom authentication service. Currently the only implemented option.
Steam	Authenticates users by their Steam Account. Set auth values accordingly!

Facebook	Authenticates users by their Facebook Account. Set auth values accordingly!
Oculus	Authenticates users by their Oculus Account and token.
PlayStation	Authenticates users by their PSN Account and token.
Xbox	Authenticates users by their Xbox Account and XSTS token.
Viveport	Authenticates users by their HTC VIVEPORT Account and user token.
None	Disables custom authentication. Same as not providing any AuthenticationValues for connect (more precisely for: OpAuthenticate).



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	AuthenticationValues		

[Public Member Functions](#) | [Properties](#) |

[List of all members](#)

AuthenticationValues Class Reference

Container for user authentication in **Photon**. Set AuthValues before you connect - all else is handled. [More...](#)

Public Member Functions

AuthenticationValues ()

Creates empty auth values without any info. [More...](#)

AuthenticationValues (string userId)

Creates minimal info about the user. If this is authenticated or not, depends on the set AuthType. [More...](#)

virtual void **SetAuthPostData** (string stringData)

Sets the data to be passed-on to the auth service via POST. [More...](#)

virtual void **SetAuthPostData** (byte[] byteData)

Sets the data to be passed-on to the auth service via POST. [More...](#)

virtual void **AddAuthParameter** (string key, string value)

Adds a key-value pair to the get-parameters used for Custom Auth (AuthGetParameters). [More...](#)

override string **ToString** ()

Transform this object into string. [More...](#)

Properties

CustomAuthenticationType **AuthType** [get, set]
The type of custom authentication provider that should be used. Currently only "Custom" or "None" (turns this off). [More...](#)

string **AuthGetParameters** [get, set]
This string must contain any (http get) parameters expected by the used authentication service. By default, username and token. [More...](#)

object **AuthPostData** [get]
Data to be passed-on to the auth service via POST. Default: null (not sent). Either string or byte[] (see setters). [More...](#)

string **Token** [get, set]
After initial authentication, **Photon** provides a token for this client / user, which is subsequently used as (cached) validation. [More...](#)

string **UserId** [get, set]
The UserId should be a unique identifier per user. This is for finding friends, etc.. [More...](#)

Detailed Description

Container for user authentication in **Photon**. Set AuthValues before you connect - all else is handled.

On **Photon**, user authentication is optional but can be useful in many cases. If you want to FindFriends, a unique ID per user is very practical.

There are basically three options for user authentication: None at all, the client sets some UserId or you can use some account web-service to authenticate a user (and set the UserId server-side).

Custom Authentication lets you verify end-users by some kind of login or token. It sends those values to **Photon** which will verify them before granting access or disconnecting the client.

The **Photon** Cloud Dashboard will let you enable this feature and set important server values for it. <https://dashboard.photonengine.com>

Constructor & Destructor Documentation

AuthenticationValues ()

Creates empty auth values without any info.

AuthenticationValues (string **userId**)

Creates minimal info about the user. If this is authenticated or not, depends on the set AuthType.

Parameters

userId Some UserId to set in **Photon**.

Member Function Documentation

virtual void AddAuthParameter (string **key,
string **value**
)**

virtual

Adds a key-value pair to the get-parameters used for Custom Auth (AuthGetParameters).

This method does uri-encoding for you.

Parameters

key Key for the value to set.

value Some value relevant for Custom Authentication.

virtual void SetAuthPostData (string **stringData)**

virtual

Sets the data to be passed-on to the auth service via POST.

Parameters

stringData String data to be used in the body of the POST request. Null or empty string will set AuthPostData to null.

virtual void SetAuthPostData (byte[] **byteData)**

virtual

Sets the data to be passed-on to the auth service via POST.

Parameters

byteData Binary token / auth-data to pass on.

override string ToString ()

Transform this object into string.

Returns

string representation of this object.

Property Documentation

string AuthGetParameters

get set

This string must contain any (http get) parameters expected by the used authentication service. By default, username and token.

Maps to operation parameter 216. Standard http get parameters are used here and passed on to the service that's defined in the server (**Photon** Cloud Dashboard).

object AuthPostData

get

Data to be passed-on to the auth service via POST. Default: null (not sent). Either string or byte[] (see setters).

Maps to operation parameter 214.

CustomAuthenticationType AuthType

get set

The type of custom authentication provider that should be used. Currently only "Custom" or "None" (turns this off).

string Token

get set

After initial authentication, **Photon** provides a token for this client / user, which is subsequently used as (cached) validation.

string UserId

get set

The UserId should be a unique identifier per user. This is for finding friends, etc..



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChannelCreationOptions		

[Static Public Attributes](#) | [Properties](#) |

[List of all members](#)

ChannelCreationOptions Class Reference

Static Public Attributes

static ChannelCreationOptions	Default = new ChannelCreationOptions() Default values of channel creation options. More...
--------------------------------------	---

Properties

bool **PublishSubscribers** [get, set]

Whether or not the channel to be created will allow client to keep a list of users. [More...](#)

int **MaxSubscribers** [get, set]

Limit of the number of users subscribed to the channel to be created. [More...](#)

Member Data Documentation

**ChannelCreationOptions Default = new
ChannelCreationOptions()**

static

Default values of channel creation options.

Property Documentation

int MaxSubscribers

get **set**

Limit of the number of users subscribed to the channel to be created.

bool PublishSubscribers

get **set**

Whether or not the channel to be created will allow client to keep a list of users.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChannelWellKnownProperties		

[Public Attributes](#) | [List of all members](#)

ChannelWellKnownProperties Class Reference

Public Attributes

```
const byte MaxSubscribers = 255
```

```
const byte PublishSubscribers = 254
```

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChatChannel		

ChatChannel Class Reference

[Public Member Functions](#) | [Public Attributes](#) |
[Properties](#) | [List of all members](#)

A channel of communication in **Photon Chat**, updated by **ChatClient** and provided as READ ONLY. [More...](#)

Public Member Functions

ChatChannel (string name)

Used internally to create new channels. This does NOT create a channel on the server! Use **ChatClient.Subscribe**. [More...](#)

void **Add** (string sender, object message, int msgId)

Used internally to add messages to this channel. [More...](#)

void **Add** (string[] senders, object[] messages, int lastMsgId)

Used internally to add messages to this channel. [More...](#)

void **TruncateMessages** ()

Reduces the number of locally cached messages in this channel to the MessageLimit (if set). [More...](#)

void **ClearMessages** ()

Clear the local cache of messages currently stored. This frees memory but doesn't affect the server. [More...](#)

string **ToStringMessages** ()

Provides a string-representation of all messages in this channel. [More...](#)

Public Attributes

readonly string **Name**
Name of the channel (used to subscribe and unsubscribe). [More...](#)

readonly List< string > **Senders** = new List<string>()
Senders of messages in chronological order. Senders and Messages refer to each other by index. Senders[x] is the sender of Messages[x]. [More...](#)

readonly List< object > **Messages** = new List<object>()
Messages in chronological order. Senders and Messages refer to each other by index. Senders[x] is the sender of Messages[x]. [More...](#)

int **MessageLimit**
If greater than 0, this channel will limit the number of messages, that it caches locally. [More...](#)

readonly HashSet< string > **Subscribers** = new HashSet<string>()
Subscribed users. [More...](#)

Properties

bool **IsPrivate** [get, set]
Is this a private 1:1 channel? [More...](#)

int **MessageCount** [get]
Count of messages this client still buffers/knows for this channel. [More...](#)

int **LastMsgId** [get, protected set]
ID of the last message received. [More...](#)

bool **PublishSubscribers** [get, protected set]
Whether or not this channel keeps track of the list of its subscribers. [More...](#)

int **MaxSubscribers** [get, protected set]
Maximum number of channel subscribers. 0 means infinite. [More...](#)

Detailed Description

A channel of communication in **Photon Chat**, updated by **ChatClient** and provided as READ ONLY.

Contains messages and senders to use (read!) and display by your GUI. Access these by: **ChatClient.PublicChannels**
ChatClient.PrivateChannels

Constructor & Destructor Documentation

ChatChannel (string *name*)

Used internally to create new channels. This does NOT create a channel on the server! Use **ChatClient.Subscribe**.

Member Function Documentation

```
void Add ( string sender,  
           object message,  
           int    msgId  
           )
```

Used internally to add messages to this channel.

```
void Add ( string[] senders,  
           object[] messages,  
           int    lastMsgId  
           )
```

Used internally to add messages to this channel.

```
void ClearMessages ( )
```

Clear the local cache of messages currently stored. This frees memory but doesn't affect the server.

```
string ToStringMessages ( )
```

Provides a string-representation of all messages in this channel.

Returns

All known messages in format "Sender: Message", line by line.

```
void TruncateMessages ( )
```

Reduces the number of locally cached messages in this channel to the MessageLimit (if set).

Member Data Documentation

int MessageLimit

If greater than 0, this channel will limit the number of messages, that it caches locally.

readonly List<object> Messages = new List<object>()

Messages in chronological order. Senders and Messages refer to each other by index. Senders[x] is the sender of Messages[x].

readonly string Name

Name of the channel (used to subscribe and unsubscribe).

readonly List<string> Senders = new List<string>()

Senders of messages in chronological order. Senders and Messages refer to each other by index. Senders[x] is the sender of Messages[x].

readonly HashSet<string> Subscribers = new HashSet<string>()

Subscribed users.

Property Documentation

bool IsPrivate

get set

Is this a private 1:1 channel?

int LastMsgId

get protected set

ID of the last message received.

int MaxSubscribers

get protected set

Maximum number of channel subscribers. 0 means infinite.

int MessageCount

get

Count of messages this client still buffers/knows for this channel.

bool PublishSubscribers

get protected set

Whether or not this channel keeps track of the list of its subscribers.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChatClient		

ChatClient Class Reference

[Public Member Functions](#) | [Public Attributes](#) |
[Properties](#) | [List of all members](#)

Central class of the **Photon Chat** API to connect, handle channels and messages. [More...](#)

Inherits IPhotonPeerListener.

Public Member Functions

bool CanChatInChannel (string channelName)
Checks if this client is ready to publish messages inside a public channel. [More...](#)

ChatClient (IChatClientListener listener, ConnectionProtocol protocol=ConnectionProtocol.Udp)
Chat client constructor. [More...](#)

bool Connect (string appld, string appVersion, **AuthenticationValues** authValues)
Connects this client to the **Photon Chat** Cloud service, which will also authenticate the user (and set a UserId). [More...](#)

bool ConnectAndSetStatus (string appld, string appVersion, **AuthenticationValues** authValues, int status=**ChatUserStatus.Online**, object message=null)
Connects this client to the **Photon Chat** Cloud service, which will also authenticate the user (and set a UserId). This also sets an online status once connected. By default it will set user status to **ChatUserStatus.Online**. See **SetOnlineStatus(int,object)** for more information. [More...](#)

void Service ()
Must be called regularly to keep connection between client and server alive and to process incoming messages. [More...](#)

void SendAcksOnly ()
Obsolete: Better use UseBackgroundWorkerForSending and **Service()**. [More...](#)

void Disconnect ()
Disconnects from the **Chat** Server by sending a "disconnect command", which prevents a timeout server-side. [More...](#)

void **StopThread** ()

Locally shuts down the connection to the **Chat** Server. This resets states locally but the server will have to timeout this peer. [More...](#)

bool **Subscribe** (string[] channels)

Sends operation to subscribe to a list of channels by name. [More...](#)

bool **Subscribe** (string[] channels, int[] lastMsgIds)

Sends operation to subscribe to a list of channels by name and possibly retrieve messages we did not receive while unsubscribed. [More...](#)

bool **Subscribe** (string[] channels, int messagesFromHistory)

Sends operation to subscribe client to channels, optionally fetching a number of messages from the cache. [More...](#)

bool **Unsubscribe** (string[] channels)

Unsubscribes from a list of channels, which stops getting messages from those. [More...](#)

bool **PublishMessage** (string channelName, object message, bool forwardAsWebhook=false)

Sends a message to a public channel which this client subscribed to. [More...](#)

bool **SendPrivateMessage** (string target, object message, bool forwardAsWebhook=false)

Sends a private message to a single target user. Calls OnPrivateMessage on the receiving client. [More...](#)

bool **SendPrivateMessage** (string target, object message, bool encrypt, bool forwardAsWebhook)

Sends a private message to a single target user. Calls OnPrivateMessage on the receiving client. [More...](#)

bool **SetOnlineStatus** (int status)
Sets the user's status without changing your status-message.
[More...](#)

bool **SetOnlineStatus** (int status, object message)
Sets the user's status without changing your status-message.
[More...](#)

bool **AddFriends** (string[] friends)
Adds friends to a list on the **Chat** Server which will send you status updates for those. [More...](#)

bool **RemoveFriends** (string[] friends)
Removes the provided entries from the list on the **Chat** Server and stops their status updates. [More...](#)

string **GetPrivateChannelNameByUser** (string userName)
Get you the (locally used) channel name for the chat between this client and another user. [More...](#)

bool **TryGetChannel** (string channelName, bool isPrivate, out **ChatChannel** channel)
Simplified access to either private or public channels by name. [More...](#)

bool **TryGetChannel** (string channelName, out **ChatChannel** channel)
Simplified access to all channels by name. Checks public channels first, then private ones. [More...](#)

bool **Subscribe** (string channel, int lastMsgId=0, int messagesFromHistory=-1, **ChannelCreationOptions** creationOptions=null)
Subscribe to a single channel and optionally sets its well-know channel properties in case the channel is created. [More...](#)

Public Attributes

	const int	DefaultMaxSubscribers Default maximum value p ChatChannel.MaxSubsc when ChatChannel.PublishSu is enabled More...
	int	MessageLimit If greater than 0, new cha limit the number of messa cache locally. More...
readonly Dictionary< string, ChatChannel >		PublicChannels Public channels this client subscribed to. More...
readonly Dictionary< string, ChatChannel >		PrivateChannels Private channels in which has exchanged messages
	ChatPeer	chatPeer = null The Chat Peer used by th More...

Properties

	string	NameServerAddress [get]	The address of last connected Server. More...
	string	FrontendAddress [get]	The address of the actual chat assigned from NameServer. P read only. More...
	string	ChatRegion [get, set]	Settable only before you connect Defaults to "EU". More...
	ChatState	State [get]	Current state of the ChatClient use CanChat. More...
ChatDisconnectCause		DisconnectedCause [get]	Disconnection cause. Check the IChatClientListener.OnDisconnected More...
	bool	CanChat [get]	Checks if this client is ready to send messages. More...
	string	AppVersion [get]	The version of your client. A new version also creates a new "version" to separate players from older versions. More...
	string	AppId [get]	

	The AppID as assigned from the Photon Cloud. More...
AuthenticationValues	AuthValues [get, set] Settable only before you connect. More...
string	UserId [get] The unique ID of a user/person in AuthValues.UserId. Set it before connect. More...
bool	UseBackgroundWorkerForSendOutgoingCommands [get, set] Defines if a background thread is used for SendOutgoingCommands, when code calls Service to dispatch messages. More...
ConnectionProtocol	TransportProtocol [get, set] Exposes the TransportProtocol used PhotonPeer. Settable when connected. More...
Dictionary< ConnectionProtocol, Type >	SocketImplementationConfig Defines which IPhotonSocket implementation to use per ConnectionProtocol. More...
DebugLevel	DebugOut [get, set] Sets the level (and amount) of output provided by the library.

Detailed Description

Central class of the **Photon Chat** API to connect, handle channels and messages.

This class must be instantiated with a **IChatClientListener** instance to get the callbacks. Integrate it into your game loop by calling `Service` regularly. If the target platform supports `Threads/Tasks`, set `UseBackgroundWorkerForSending = true`, to let the **ChatClient** keep the connection by sending from an independent thread.

Call `Connect` with an `AppId` that is setup as **Photon Chat** application. Note: `Connect` covers multiple messages between this client and the servers. A short workflow will connect you to a chat server.

Each **ChatClient** resembles a user in chat (set in `Connect`). Each user automatically subscribes a channel for incoming private messages and can message any other user privately. Before you publish messages in any non-private channel, that channel must be subscribed.

`PublicChannels` is a list of subscribed channels, containing messages and senders. `PrivateChannels` contains all incoming and sent private messages.

Constructor & Destructor Documentation

```
ChatClient ( IChatClientListener listener,  
             ConnectionProtocol protocol = ConnectionProtocol.Udp  
             )
```

Chat client constructor.

Parameters

listener The chat listener implementation.

protocol Connection protocol to be used by this client. Default is ConnectionProtocol.Udp.

Member Function Documentation

bool AddFriends (string[] friends)

Adds friends to a list on the **Chat** Server which will send you status updates for those.

AddFriends and RemoveFriends enable clients to handle their friend list in the **Photon Chat** server. Having users on your friends list gives you access to their current online status (and whatever info your client sets in it).

Each user can set an online status consisting of an integer and an arbitrary (serializable) object. The object can be null, Hashtable, object[] or anything else **Photon** can serialize.

The status is published automatically to friends (anyone who set your user ID with AddFriends).

Photon flushes friends-list when a chat client disconnects, so it has to be set each time. If your community API gives you access to online status already, you could filter and set online friends in AddFriends.

Actual friend relations are not persistent and have to be stored outside of **Photon**.

Parameters

friends Array of friend userIDs.

Returns

If the operation could be sent.

bool CanChatInChannel (string channelName)

Checks if this client is ready to publish messages inside a public channel.

Parameters

channelName The channel to do the check with.

Returns

Whether or not this client is ready to publish messages inside the public channel with the specified channelName.

```
bool Connect ( string          appld,  
               string          appVersion,  
               AuthenticationValues authValues  
             )
```

Connects this client to the **Photon Chat** Cloud service, which will also authenticate the user (and set a UserId).

Parameters

appld Get your **Photon Chat** Appld from the [Dashboard](#).

appVersion Any version string you make up. Used to separate users and variants of your clients, which might be incompatible.

authValues Values for authentication. You can leave this null, if you set a UserId before. If you set authValues, they will override any UserId set before.

Returns

```
bool  
ConnectAndSetStatus ( string          appld,  
                     string          appVersion,  
                     AuthenticationValues authValues,  
                     int             status = ChatUserSta  
                     object          message = null
```

)

Connects this client to the **Photon Chat** Cloud service, which will also authenticate the user (and set a `UserId`). This also sets an online status connected. By default it will set user status to **ChatUserStatus.Online**. **SetOnlineStatus(int,object)** for more information.

Parameters

- appld** Get your **Photon Chat** AppId from the [Dashboard](#).
- appVersion** Any version string you make up. Used to separate user variants of your clients, which might be incompatible.
- authValues** Values for authentication. You can leave this null, if you have a `UserId` before. If you set `authValues`, they will override `UserId` set before.
- status** User status to set when connected. Predefined states are in the class **ChatUserStatus**. Other values can be used at your discretion.
- message** Optional status message. Also sets a status-message which you can get.

Returns

If the connection attempt could be sent at all.

void Disconnect ()

Disconnects from the **Chat** Server by sending a "disconnect command", which prevents a timeout server-side.

string GetPrivateChannelNameByUser (string **userName**)

Get you the (locally used) channel name for the chat between this client and another user.

Parameters

- userName** Remote user's name or `UserId`.

Returns

The (locally used) channel name for a private channel.

```
bool PublishMessage ( string  channelName,  
                      object  message,  
                      bool    forwardAsWebhook = false  
                      )
```

Sends a message to a public channel which this client subscribed to.

Before you publish to a channel, you have to subscribe it. Everyone in that channel will get the message.

Parameters

channelName	Name of the channel to publish to.
message	Your message (string or any serializable data).
forwardAsWebhook	Optionally, public messages can be forwarded as webhooks. Configure webhooks for your Chat app to use this.

Returns

False if the client is not yet ready to send messages.

```
bool RemoveFriends ( string[] friends )
```

Removes the provided entries from the list on the **Chat** Server and stops their status updates.

Photon flushes friends-list when a chat client disconnects. Unless you want to remove individual entries, you don't have to RemoveFriends.

AddFriends and RemoveFriends enable clients to handle their friend list in the **Photon Chat** server. Having users on your friends list gives you access to their current online status (and whatever info your client sets in it).

Each user can set an online status consisting of an integer and an arbitrary (serializable) object. The object can be null, Hashtable, object[] or anything else **Photon** can serialize.

The status is published automatically to friends (anyone who set your user ID with AddFriends).

Photon flushes friends-list when a chat client disconnects, so it has to be set each time. If your community API gives you access to online status already, you could filter and set online friends in AddFriends.

Actual friend relations are not persistent and have to be stored outside of **Photon**.

AddFriends and RemoveFriends enable clients to handle their friend list in the **Photon Chat** server. Having users on your friends list gives you access to their current online status (and whatever info your client sets in it).

Each user can set an online status consisting of an integer and an arbitrary (serializable) object. The object can be null, Hashtable, object[] or anything else **Photon** can serialize.

The status is published automatically to friends (anyone who set your user ID with AddFriends).

Actual friend relations are not persistent and have to be stored outside of **Photon**.

Parameters

friends Array of friend userIDs.

Returns

If the operation could be sent.

void SendAcksOnly ()

Obsolete: Better use UseBackgroundWorkerForSending and

Service().

```
bool SendPrivateMessage ( string target,  
                          object message,  
                          bool forwardAsWebhook = false  
                        )
```

Sends a private message to a single target user. Calls OnPrivateMessage on the receiving client.

Parameters

target	Username to send this message to.
message	The message you want to send. Can be a simple string or anything serializable.
forwardAsWebhook	Optionally, private messages can be forwarded as webhooks. Configure webhooks for your Chat app to use this.

Returns

True if this clients can send the message to the server.

```
bool SendPrivateMessage ( string target,  
                          object message,  
                          bool encrypt,  
                          bool forwardAsWebhook  
                        )
```

Sends a private message to a single target user. Calls OnPrivateMessage on the receiving client.

Parameters

target	Username to send this message to.
message	The message you want to send. Can be a simple string or anything serializable.
encrypt	Optionally, private messages can be

encrypted. Encryption is not end-to-end as the server decrypts the message.

forwardAsWebhook Optionally, private messages can be forwarded as webhooks. Configure webhooks for your **Chat** app to use this.

Returns

True if this clients can send the message to the server.

void Service ()

Must be called regularly to keep connection between client and server alive and to process incoming messages.

This method limits the effort it does automatically using the private variable `msDeltaForServiceCalls`. That value is lower for connect and multiplied by 4 when chat-server connection is ready.

bool SetOnlineStatus (int **status**)

Sets the user's status without changing your status-message.

The predefined status values can be found in class **ChatUserStatus**. State **ChatUserStatus.Invisible** will make you offline for everyone and send no message.

You can set custom values in the status integer. Aside from the pre-configured ones, all states will be considered visible and online. Else, no one would see the custom state.

This overload does not change the set message.

Parameters

status Predefined states are in class **ChatUserStatus**. Other values can be used at will.

Returns

True if the operation gets called on the server.

```
bool SetOnlineStatus ( int      status,  
                      object message  
                      )
```

Sets the user's status without changing your status-message.

The predefined status values can be found in class **ChatUserStatus**. State **ChatUserStatus.Invisible** will make you offline for everyone and send no message.

You can set custom values in the status integer. Aside from the pre-configured ones, all states will be considered visible and online. Else, no one would see the custom state.

The message object can be anything that **Photon** can serialize, including (but not limited to) Hashtable, object[] and string. This value is defined by your own conventions.

Parameters

status Predefined states are in class **ChatUserStatus**. Other values can be used at will.

message Also sets a status-message which your friends can get.

Returns

True if the operation gets called on the server.

```
void StopThread ( )
```

Locally shuts down the connection to the **Chat** Server. This resets states locally but the server will have to timeout this peer.

```
bool Subscribe ( string[] channels )
```

Sends operation to subscribe to a list of channels by name.

Parameters

channels List of channels to subscribe to. Avoid null or empty values.

Returns

If the operation could be sent at all (Example: Fails if not connected to **Chat Server**).

```
bool Subscribe ( string[] channels,  
                int[]   lastMsgIds  
                )
```

Sends operation to subscribe to a list of channels by name and possibly retrieve messages we did not receive while unsubscribed.

Parameters

channels List of channels to subscribe to. Avoid null or empty values.

lastMsgIds ID of last message received per channel. Useful when re subscribing to receive only messages we missed.

Returns

If the operation could be sent at all (Example: Fails if not connected to **Chat Server**).

```
bool Subscribe ( string[] channels,  
                int    messagesFromHistory  
                )
```

Sends operation to subscribe client to channels, optionally fetching a number of messages from the cache.

Subscribes channels will forward new messages to this user. Use PublishMessage to do so. The messages cache is limited but can be useful to get into ongoing conversations, if that's needed.

Parameters

channels	List of channels to subscribe to. Avoid null or empty values.
messagesFromHistory	0: no history. 1 and higher: number of messages in history. -1: all available history.

Returns

If the operation could be sent at all (Example: Fails if not connected to **Chat Server**).

```
bool
Subscribe ( string          channel,
            int             lastMsgId = 0,
            int             messagesFromHistory = -1,
            ChannelCreationOptions creationOptions = null
          )
```

Subscribe to a single channel and optionally sets its well-know channel properties in case the channel is created.

Parameters

channel	name of the channel to subscribe to
lastMsgId	ID of the last received message from this channel when re subscribing to receive only missed messages, default is 0
messagesFromHistory	how many missed messages to receive from history, default is none/-1
creationOptions	options to be used in case the channel to subscribe to will be created.

Returns

```
bool TryGetChannel ( string      channelName,  
                    bool      isPrivate,  
                    out ChatChannel channel  
                    )
```

Simplified access to either private or public channels by name.

Parameters

channelName Name of the channel to get. For private channels, the channel-name is composed of both user's names.

isPrivate Define if you expect a private or public channel.

channel Out parameter gives you the found channel, if any.

Returns

True if the channel was found.

```
bool TryGetChannel ( string      channelName,  
                    out ChatChannel channel  
                    )
```

Simplified access to all channels by name. Checks public channels first, then private ones.

Parameters

channelName Name of the channel to get.

channel Out parameter gives you the found channel, if any.

Returns

True if the channel was found.

```
bool Unsubscribe ( string[] channels )
```

Unsubscribes from a list of channels, which stops getting messages from those.

The client will remove these channels from the PublicChannels dictionary once the server sent a response to this request.

The request will be sent to the server and **IChatClientListener.OnUnsubscribed** gets called when the server actually removed the channel subscriptions.

Unsubscribe will fail if you include null or empty channel names.

Parameters

channels Names of channels to unsubscribe.

Returns

False, if not connected to a chat server.

Member Data Documentation

ChatPeer chatPeer = null

The **Chat** Peer used by this client.

const int DefaultMaxSubscribers = 100

Default maximum value possible for **ChatChannel.MaxSubscribers** when **ChatChannel.PublishSubscribers** is enabled

int MessageLimit

If greater than 0, new channels will limit the number of messages they cache locally.

This can be useful to limit the amount of memory used by chats. You can set a MessageLimit per channel but this value gets applied to new ones.

Note: Changing this value, does not affect ChatChannels that are already in use!

readonly Dictionary<string, ChatChannel> PrivateChannels

Private channels in which this client has exchanged messages.

readonly Dictionary<string, ChatChannel> PublicChannels

Public channels this client is subscribed to.

Property Documentation

string AppId

get

The AppID as assigned from the **Photon** Cloud.

string AppVersion

get

The version of your client. A new version also creates a new "virtual app" to separate players from older client versions.

AuthenticationValues AuthValues

get set

Settable only before you connect!

bool CanChat

get

Checks if this client is ready to send messages.

string ChatRegion

get set

Settable only before you connect! Defaults to "EU".

DebugLevel DebugOut

get set

Sets the level (and amount) of debug output provided by the library.
This affects the callbacks to **IChatClientListener.DebugReturn**.

Default Level: Error.

ChatDisconnectCause DisconnectedCause

get

Disconnection cause. Check this inside **IChatClientListener.OnDisconnected**.

string FrontendAddress

get

The address of the actual chat server assigned from NameServer.
Public for read only.

string NameServerAddress

get

The address of last connected Name Server.

Dictionary<ConnectionProtocol, Type> SocketImplementationConfig

get

Defines which IPhotonSocket class to use per ConnectionProtocol.

Several platforms have special Socket implementations and slightly different APIs. To accomodate this, switching the socket implementation for a network protocol was made available. By default, UDP and TCP have socket implementations assigned.

You only need to set the SocketImplementationConfig once, after creating a PhotonPeer and before connecting. If you switch the TransportProtocol, the correct implementation is being used.

ChatState State

get

Current state of the **ChatClient**. Also use CanChat.

ConnectionProtocol TransportProtocol

get set

Exposes the TransportProtocol of the used PhotonPeer. Settable while not connected.

bool UseBackgroundWorkerForSending

get set

Defines if a background thread will call SendOutgoingCommands, while your code calls Service to dispatch received messages.

The benefit of using a background thread to call SendOutgoingCommands is this:

Even if your game logic is being paused, the background thread will keep the connection to the server up. On a lower level, acknowledgements and pings will prevent a server-side timeout while (e.g.) Unity loads assets.

Your game logic still has to call Service regularly, or else incoming messages are not dispatched. As this typically triggers UI updates, it's easier to call Service from the main/UI thread.

string UserId

get

The unique ID of a user/person, stored in AuthValues.UserId. Set it before you connect.

This value wraps AuthValues.UserId. It's not a nickname and we assume users with the same userID are the same person.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChatEventCode		

[Public Attributes](#) | [List of all members](#)

ChatEventCode Class Reference

Wraps up internally used constants in **Photon Chat** events. You don't have to use them directly usually. [More...](#)

Public Attributes

const byte **ChatMessages** = 0
(0) Event code for messages published in public channels. [More...](#)

const byte **Users** = 1
(1) Not Used. [More...](#)

const byte **PrivateMessage** = 2
(2) Event code for messages published in private channels [More...](#)

const byte **FriendsList** = 3
(3) Not Used. [More...](#)

const byte **StatusUpdate** = 4
(4) Event code for status updates. [More...](#)

const byte **Subscribe** = 5
(5) Event code for subscription acks. [More...](#)

const byte **Unsubscribe** = 6
(6) Event code for unsubscribe acks. [More...](#)

const byte **UserSubscribed** = 8
(7) Event code for new user subscription to a channel where **ChatChannel.PublishSubscribers** is enabled. [More...](#)

const byte **UserUnsubscribed** = 9
(8) Event code for when user unsubscribes from a channel where **ChatChannel.PublishSubscribers** is enabled. [More...](#)

Detailed Description

Wraps up internally used constants in **Photon Chat** events. You don't have to use them directly usually.

Member Data Documentation

const byte ChatMessages = 0

(0) Event code for messages published in public channels.

const byte FriendsList = 3

(3) Not Used.

const byte PrivateMessage = 2

(2) Event code for messages published in private channels

const byte StatusUpdate = 4

(4) Event code for status updates.

const byte Subscribe = 5

(5) Event code for subscription acks.

const byte Unsubscribe = 6

(6) Event code for unsubscribe acks.

const byte Users = 1

(1) Not Used.

const byte UserSubscribed = 8

(7) Event code for new user subscription to a channel where **ChatChannel.PublishSubscribers** is enabled.

const byte UserUnsubscribed = 9

(8) Event code for when user unsubscribes from a channel where **ChatChannel.PublishSubscribers** is enabled.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChatOperationCode		

[Public Attributes](#) | [List of all members](#)

ChatOperationCode Class Reference

Wraps up codes for operations used internally in **Photon Chat**. You don't have to use them directly usually. [More...](#)

Public Attributes

const byte **Authenticate** = 230
(230) Operation Authenticate. [More...](#)

const byte **Subscribe** = 0
(0) Operation to subscribe to chat channels. [More...](#)

const byte **Unsubscribe** = 1
(1) Operation to unsubscribe from chat channels. [More...](#)

const byte **Publish** = 2
(2) Operation to publish a message in a chat channel.
[More...](#)

const byte **SendPrivate** = 3
(3) Operation to send a private message to some other user. [More...](#)

const byte **ChannelHistory** = 4
(4) Not used yet. [More...](#)

const byte **UpdateStatus** = 5
(5) Set your (client's) status. [More...](#)

const byte **AddFriends** = 6
(6) Add friends the list of friends that should update you of their status. [More...](#)

const byte **RemoveFriends** = 7
(7) Remove friends from list of friends that should update you of their status. [More...](#)

Detailed Description

Wraps up codes for operations used internally in **Photon Chat**. You don't have to use them directly usually.

Member Data Documentation

const byte AddFriends = 6

(6) Add friends the list of friends that should update you of their status.

const byte Authenticate = 230

(230) Operation Authenticate.

const byte ChannelHistory = 4

(4) Not used yet.

const byte Publish = 2

(2) Operation to publish a message in a chat channel.

const byte RemoveFriends = 7

(7) Remove friends from list of friends that should update you of their status.

const byte SendPrivate = 3

(3) Operation to send a private message to some other user.

const byte Subscribe = 0

(0) Operation to subscribe to chat channels.

const byte Unsubscribe = 1

(1) Operation to unsubscribe from chat channels.

const byte UpdateStatus = 5

(5) Set your (client's) status.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChatParameterCode		

[Public Attributes](#) | [List of all members](#)

ChatParameterCode Class Reference

Wraps up codes for parameters (in operations and events) used internally in **Photon Chat**. You don't have to use them directly usually.
[More...](#)

Public Attributes

const byte **Channels** = 0
(0) Array of chat channels. [More...](#)

const byte **Channel** = 1
(1) Name of a single chat channel. [More...](#)

const byte **Messages** = 2
(2) Array of chat messages. [More...](#)

const byte **Message** = 3
(3) A single chat message. [More...](#)

const byte **Senders** = 4
(4) Array of names of the users who sent the array of chat messages. [More...](#)

const byte **Sender** = 5
(5) Name of a the user who sent a chat message. [More...](#)

const byte **ChannelUserCount** = 6
(6) Not used. [More...](#)

const byte **UserId** = 225
(225) Name of user to send a (private) message to. [More...](#)

const byte **MsgId** = 8
(8) Id of a message. [More...](#)

const byte **MsgIds** = 9
(9) Not used. [More...](#)

const byte **Secret** = 221

(221) Secret token to identify an authorized user. [More...](#)

const byte **SubscribeResults** = 15
(15) Subscribe operation result parameter. A bool[] with result per channel. [More...](#)

const byte **Status** = 10
(10) Status [More...](#)

const byte **Friends** = 11
(11) Friends [More...](#)

const byte **SkipMessage** = 12
(12) SkipMessage is used in SetOnlineStatus and if true, the message is not being broadcast. [More...](#)

const byte **HistoryLength** = 14
(14) Number of message to fetch from history. 0: no history. 1 and higher: number of messages in history. -1: all history. [More...](#)

const byte **WebFlags** = 21
(21) WebFlags object for changing behaviour of webhooks from client. [More...](#)

const byte **Properties** = 22
(22) Properties of channel or user. [More...](#)

const byte **ChannelSubscribers** = 23
(23) Array of UserIds of users already subscribed to a channel. [More...](#)

Detailed Description

Wraps up codes for parameters (in operations and events) used internally in **Photon Chat**. You don't have to use them directly usually.

Member Data Documentation

const byte Channel = 1

(1) Name of a single chat channel.

const byte Channels = 0

(0) Array of chat channels.

const byte ChannelSubscribers = 23

(23) Array of UserIds of users already subscribed to a channel.

Used in Subscribe event when PublishSubscribers is enabled. Does not include local user who just subscribed. Maximum length is (**ChatChannel.MaxSubscribers** - 1).

const byte ChannelUserCount = 6

(6) Not used.

const byte Friends = 11

(11) Friends

const byte HistoryLength = 14

(14) Number of message to fetch from history. 0: no history. 1 and higher: number of messages in history. -1: all history.

const byte Message = 3

(3) A single chat message.

const byte Messages = 2

(2) Array of chat messages.

const byte MsgId = 8

(8) Id of a message.

const byte MsgIds = 9

(9) Not used.

const byte Properties = 22

(22) Properties of channel or user.

In event **ChatEventCode.Subscribe** it's always channel properties.

const byte Secret = 221

(221) Secret token to identify an authorized user.

The code is used in LoadBalancing and copied over here.

const byte Sender = 5

(5) Name of a the user who sent a chat message.

const byte Senders = 4

(4) Array of names of the users who sent the array of chat messages.

const byte SkipMessage = 12

(12) SkipMessage is used in SetOnlineStatus and if true, the message is not being broadcast.

const byte Status = 10

(10) Status

const byte SubscribeResults = 15

(15) Subscribe operation result parameter. A bool[] with result per channel.

const byte UserId = 225

(225) Name of user to send a (private) message to.

The code is used in LoadBalancing and copied over here.

const byte WebFlags = 21

(21) WebFlags object for changing behaviour of webhooks from client.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChatPeer		

ChatPeer Class Reference

[Public Member Functions](#) | [Public Attributes](#) | [Properties](#) | [List of all members](#)

Provides basic operations of the **Photon Chat** server. This internal class is used by public **ChatClient**. [More...](#)

Inherits PhotonPeer.

Public Member Functions

ChatPeer (IPhotonPeerListener listener, ConnectionProtocol protocol)

Chat Peer constructor. [More...](#)

bool **Connect** ()

Connects to NameServer. [More...](#)

bool **AuthenticateOnNameServer** (string appld, string appVersion, string region, **AuthenticationValues** authValues)

Authenticates on NameServer. [More...](#)

Public Attributes

const string **NameServerHost** = "ns.exitgames.com"
Name Server Host Name for **Photon** Cloud. Without port and without any prefix. [More...](#)

const string **NameServerHttp** =
"http://ns.exitgamescloud.com:80/photon/n"
Name Server for HTTP connections to the **Photon** Cloud. Includes prefix and port. [More...](#)

Properties

string **NameServerAddress** [get]

Name Server Address for **Photon** Cloud (based on current protocol). You can use the default values and usually won't have to set this value. [More...](#)

Detailed Description

Provides basic operations of the **Photon Chat** server. This internal class is used by public **ChatClient**.

Constructor & Destructor Documentation

```
ChatPeer ( IPhotonPeerListener listener,  
           ConnectionProtocol protocol  
           )
```

Chat Peer constructor.

Parameters

listener Chat listener implementation.

protocol Protocol to be used by the peer.

Member Function Documentation

```
bool  
AuthenticateOnNameServer ( string           appId,  
                           string           appVersion,  
                           string           region,  
                           AuthenticationValues authValues  
                           )
```

Authenticates on NameServer.

Returns

If the authentication operation request could be sent.

```
bool Connect ( )
```

Connects to NameServer.

Returns

If the connection attempt could be sent.

Member Data Documentation

```
const string NameServerHost = "ns.exitgames.com"
```

Name Server Host Name for **Photon** Cloud. Without port and without any prefix.

```
const string NameServerHttp =  
"http://ns.exitgamescloud.com:80/photon/n"
```

Name Server for HTTP connections to the **Photon** Cloud. Includes prefix and port.

Property Documentation

string NameServerAddress

get

Name Server Address for **Photon** Cloud (based on current protocol). You can use the default values and usually won't have to set this value.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChatUserStatus		

[Public Attributes](#) | [List of all members](#)

ChatUserStatus Class Reference

Contains commonly used status values for SetOnlineStatus. You can define your own. [More...](#)

Public Attributes

const int **Offline** = 0
(0) Offline. [More...](#)

const int **Invisible** = 1
(1) Be invisible to everyone. Sends no message. [More...](#)

const int **Online** = 2
(2) Online and available. [More...](#)

const int **Away** = 3
(3) Online but not available. [More...](#)

const int **DND** = 4
(4) Do not disturb. [More...](#)

const int **LFG** = 5
(5) Looking For Game/Group. Could be used when you want to be invited or do matchmaking. [More...](#)

const int **Playing** = 6
(6) Could be used when in a room, playing. [More...](#)

Detailed Description

Contains commonly used status values for SetOnlineStatus. You can define your own.

While "online" (value 2 and up), the status message will be sent to anyone who has you on his friend list.

Define custom online status values as you like with these rules: 0: Means "offline". It will be used when you are not connected. In this status, there is no status message. 1: Means "invisible" and is sent to friends as "offline". They see status 0, no message but you can chat. 2: And any higher value will be treated as "online". Status can be set.

Member Data Documentation

const int Away = 3

(3) Online but not available.

const int DND = 4

(4) Do not disturb.

const int Invisible = 1

(1) Be invisible to everyone. Sends no message.

const int LFG = 5

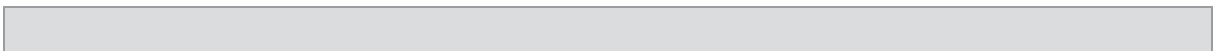
(5) Looking For Game/Group. Could be used when you want to be invited or do matchmaking.

const int Offline = 0

(0) Offline.

const int Online = 2

(2) Online and available.



const int Playing = 6

(6) Could be used when in a room, playing.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ErrorCode		

[Public Attributes](#) | [List of all members](#)

ErrorCode Class Reference

ErrorCode defines the default codes associated with **Photon** client/server communication. [More...](#)

Public Attributes

const int **Ok** = 0

(0) is always "OK", anything else an error or specific situation. [More...](#)

const int **OperationNotAllowedInCurrentState** = -3

(-3) Operation can't be executed yet (e.g. OpJoin can't be called before being authenticated, RaiseEvent can't be used before getting into a room). [More...](#)

const int **InvalidOperationCode** = -2

(-2) The operation you called is not implemented on the server (application) you connect to. Make sure you run the fitting applications. [More...](#)

const int **InternalServerError** = -1

(-1) Something went wrong in the server. Try to reproduce and contact Exit Games. [More...](#)

const int **InvalidAuthentication** = 0x7FFF

(32767) Authentication failed. Possible cause: AppId is unknown to **Photon** (in cloud service). [More...](#)

const int **GameIdAlreadyExists** = 0x7FFF - 1

(32766) GameId (name) already in use (can't create another). Change name. [More...](#)

const int **GameFull** = 0x7FFF - 2

(32765) Game is full. This rarely happens when some player joined the room before your join completed. [More...](#)

const int **GameClosed** = 0x7FFF - 3

(32764) Game is closed and can't be joined. Join another game. [More...](#)

const int **ServerFull** = 0x7FFF - 5
(32762) Not in use currently. [More...](#)

const int **UserBlocked** = 0x7FFF - 6
(32761) Not in use currently. [More...](#)

const int **NoRandomMatchFound** = 0x7FFF - 7
(32760) Random matchmaking only succeeds if a room exists that is neither closed nor full. Repeat in a few seconds or create a new room. [More...](#)

const int **GameDoesNotExist** = 0x7FFF - 9
(32758) Join can fail if the room (name) is not existing (anymore). This can happen when players leave while you join. [More...](#)

const int **MaxCcuReached** = 0x7FFF - 10
(32757) Authorization on the **Photon** Cloud failed because the concurrent users (CCU) limit of the app's subscription is reached. [More...](#)

const int **InvalidRegion** = 0x7FFF - 11
(32756) Authorization on the **Photon** Cloud failed because the app's subscription does not allow to use a particular region's server. [More...](#)

const int **CustomAuthenticationFailed** = 0x7FFF - 12
(32755) Custom Authentication of the user failed due to setup reasons (see Cloud Dashboard) or the provided user data (like username or token). Check error message for details. [More...](#)

Detailed Description

ErrorCode defines the default codes associated with **Photon** client/server communication.

Member Data Documentation

const int CustomAuthenticationFailed = 0x7FFF - 12

(32755) Custom Authentication of the user failed due to setup reasons (see Cloud Dashboard) or the provided user data (like username or token). Check error message for details.

const int GameClosed = 0x7FFF - 3

(32764) Game is closed and can't be joined. Join another game.

const int GameDoesNotExist = 0x7FFF - 9

(32758) Join can fail if the room (name) is not existing (anymore). This can happen when players leave while you join.

const int GameFull = 0x7FFF - 2

(32765) Game is full. This rarely happens when some player joined the room before your join completed.

const int GameldAlreadyExists = 0x7FFF - 1

(32766) Gameld (name) already in use (can't create another). Change name.

const int InternalServerError = -1

(-1) Something went wrong in the server. Try to reproduce and contact Exit Games.

const int InvalidAuthentication = 0x7FFF

(32767) Authentication failed. Possible cause: AppId is unknown to **Photon** (in cloud service).

const int InvalidOperationCode = -2

(-2) The operation you called is not implemented on the server (application) you connect to. Make sure you run the fitting applications.

const int InvalidRegion = 0x7FFF - 11

(32756) Authorization on the **Photon** Cloud failed because the app's subscription does not allow to use a particular region's server.

Some subscription plans for the **Photon** Cloud are region-bound. Servers of other regions can't be used then. Check your master server address and compare it with your **Photon** Cloud Dashboard's info. <https://cloud.photonengine.com/dashboard>

OpAuthorize is part of connection workflow but only on the **Photon** Cloud, this error can happen. Self-hosted **Photon** servers with a CCU limited license won't let a client connect at all.

const int MaxCcuReached = 0x7FFF - 10

(32757) Authorization on the **Photon** Cloud failed because the concurrent users (CCU) limit of the app's subscription is reached.

Unless you have a plan with "CCU Burst", clients might fail the authentication step during connect. Affected client are unable to call operations. Please note that players who end a game and return to the master server will disconnect and re-connect, which means that they just played and are rejected in the next minute / re-connect. This is a temporary measure. Once the CCU is below the limit, players will be able to connect and play again.

OpAuthorize is part of connection workflow but only on the **Photon** Cloud, this error can happen. Self-hosted **Photon** servers with a CCU limited license won't let a client connect at all.

const int NoRandomMatchFound = 0x7FFF - 7

(32760) Random matchmaking only succeeds if a room exists that is neither closed nor full. Repeat in a few seconds or create a new room.

const int Ok = 0

(0) is always "OK", anything else an error or specific situation.

const int OperationNotAllowedInCurrentState = -3

(-3) Operation can't be executed yet (e.g. OpJoin can't be called before being authenticated, RaiseEvent can't be used before getting into a room).

Before you call any operations on the Cloud servers, the automated client workflow must complete its authorization. In PUN, wait until State is: JoinedLobby or ConnectedToMaster

const int ServerFull = 0x7FFF - 5

(32762) Not in use currently.

const int UserBlocked = 0x7FFF - 6

(32761) Not in use currently.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	IChatClientListener		

[Public Member Functions](#) | [List of all members](#)

IChatClientListener Interface Reference

Callback interface for **Chat** client side. Contains callback methods to notify your app about updates. Must be provided to new **ChatClient** in constructor [More...](#)

Public Member Functions

void **DebugReturn** (DebugLevel level, string message)
All debug output of the library will be reported through this method. Print it or put it in a buffer to use it on-screen. [More...](#)

void **OnDisconnected** ()
Disconnection happened. [More...](#)

void **OnConnected** ()
Client is connected now. [More...](#)

void **OnChatStateChange** (ChatState state)
The **ChatClient**'s state changed. Usually, OnConnected and OnDisconnected are the callbacks to react to. [More...](#)

void **OnGetMessages** (string channelName, string[] senders, object[] messages)
Notifies app that client got new messages from server Number of senders is equal to number of messages in 'messages'. Sender with number '0' corresponds to message with number '0', sender with number '1' corresponds to message with number '1' and so on [More...](#)

void **OnPrivateMessage** (string sender, object message, string channelName)
Notifies client about private message [More...](#)

void **OnSubscribed** (string[] channels, bool[] results)
Result of Subscribe operation. Returns subscription result for every requested channel name. [More...](#)

void **OnUnsubscribed** (string[] channels)
Result of Unsubscribe operation. Returns for channel name if the channel is now unsubscribed. [More...](#)

void **OnStatusUpdate** (string user, int status, bool gotMessage, object message)
New status of another user (you get updates for users set in your friends list). [More...](#)

void **OnUserSubscribed** (string channel, string user)
A user has subscribed to a public chat channel [More...](#)

void **OnUserUnsubscribed** (string channel, string user)
A user has unsubscribed from a public chat channel [More...](#)

Detailed Description

Callback interface for **Chat** client side. Contains callback methods to notify your app about updates. Must be provided to new **ChatClient** in constructor

Member Function Documentation

```
void DebugReturn ( DebugLevel level,  
                  string    message  
                  )
```

All debug output of the library will be reported through this method. Print it or put it in a buffer to use it on-screen.

Parameters

level DebugLevel (severity) of the message.
message Debug text. Print to System.Console or screen.

```
void OnChatStateChange ( ChatState state )
```

The **ChatClient**'s state changed. Usually, OnConnected and OnDisconnected are the callbacks to react to.

Parameters

state The new state.

```
void OnConnected ( )
```

Client is connected now.

Clients have to be connected before they can send their state, subscribe to channels and send any messages.

```
void OnDisconnected ( )
```


Disconnection happened.

```
void OnGetMessages ( string  channelName,  
                    string[] senders,  
                    object[] messages  
                    )
```

Notifies app that client got new messages from server Number of senders is equal to number of messages in 'messages'. Sender with number '0' corresponds to message with number '0', sender with number '1' corresponds to message with number '1' and so on

Parameters

channelName channel from where messages came
senders list of users who sent messages
messages list of messages it self

```
void OnPrivateMessage ( string sender,  
                       object message,  
                       string channelName  
                       )
```

Notifies client about private message

Parameters

sender user who sent this message
message message it self
channelName channelName for private messages (messages you sent yourself get added to a channel per target username)

```
void OnStatusUpdate ( string user,  
                     int status,
```

```
bool gotMessage,  
object message  
)
```

New status of another user (you get updates for users set in your friends list).

Parameters

user	Name of the user.
status	New status of that user.
gotMessage	True if the status contains a message you should cache locally. False: This status update does not include a message (keep any you have).
message	Message that user set.

```
void OnSubscribed ( string[] channels,  
bool[] results  
)
```

Result of Subscribe operation. Returns subscription result for every requested channel name.

If multiple channels sent in Subscribe operation, OnSubscribed may be called several times, each call with part of sent array or with single channel in "channels" parameter. Calls order and order of channels in "channels" parameter may differ from order of channels in "channels" parameter of Subscribe operation.

Parameters

channels	Array of channel names.
results	Per channel result if subscribed.

```
void OnUnsubscribed ( string[] channels )
```

Result of Unsubscribe operation. Returns for channel name if the

channel is now unsubscribed.

If multiple channels sent in Unsubscribe operation, OnUnsubscribed may be called several times, each call with part of sent array or with single channel in "channels" parameter. Calls order and order of channels in "channels" parameter may differ from order of channels in "channels" parameter of Unsubscribe operation.

Parameters

channels Array of channel names that are no longer subscribed.

```
void OnUserSubscribed ( string channel,  
                        string user  
                        )
```

A user has subscribed to a public chat channel

Parameters

channel Name of the chat channel
user UserId of the user who subscribed

```
void OnUserUnsubscribed ( string channel,  
                           string user  
                           )
```

A user has unsubscribed from a public chat channel

Parameters

channel Name of the chat channel
user UserId of the user who unsubscribed



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ParameterCode		

[Public Attributes](#) | [List of all members](#)

ParameterCode Class Reference

Class for constants. Codes for parameters of Operations and Events.
[More...](#)

Public Attributes

const byte **ApplicationId** = 224
(224) Your application's ID: a name on your own **Photon** or a GUID on the **Photon** Cloud [More...](#)

const byte **Secret** = 221
(221) Internally used to establish encryption [More...](#)

const byte **AppVersion** = 220
(220) Version of your application [More...](#)

const byte **ClientAuthenticationType** = 217
(217) This key's (byte) value defines the target custom authentication type/service the client connects with. Used in OpAuthenticate [More...](#)

const byte **ClientAuthenticationParams** = 216
(216) This key's (string) value provides parameters sent to the custom authentication type/service the client connects with. Used in OpAuthenticate [More...](#)

const byte **ClientAuthenticationData** = 214
(214) This key's (string or byte[]) value provides parameters sent to the custom authentication service setup in **Photon** Dashboard. Used in OpAuthenticate [More...](#)

const byte **Region** = 210
(210) Used for region values in OpAuth and OpGetRegions. [More...](#)

const byte **Address** = 230
(230) Address of a (game) server to use. [More...](#)

const byte **UserId** = 225

(225) User's ID [More...](#)

Detailed Description

Class for constants. Codes for parameters of Operations and Events.

Member Data Documentation

const byte Address = 230

(230) Address of a (game) server to use.

const byte ApplicationId = 224

(224) Your application's ID: a name on your own **Photon** or a GUID on the **Photon** Cloud

const byte AppVersion = 220

(220) Version of your application

const byte ClientAuthenticationData = 214

(214) This key's (string or byte[]) value provides parameters sent to the custom authentication service setup in **Photon** Dashboard. Used in OpAuthenticate

const byte ClientAuthenticationParams = 216

(216) This key's (string) value provides parameters sent to the custom authentication type/service the client connects with. Used in OpAuthenticate

const byte ClientAuthenticationType = 217

(217) This key's (byte) value defines the target custom authentication type/service the client connects with. Used in OpAuthenticate

const byte Region = 210

(210) Used for region values in OpAuth and OpGetRegions.

const byte Secret = 221

(221) Internally used to establish encryption

const byte UserId = 225

(225) User's ID



Photon Unity Networking 2 2.12

[Main Page](#)[Related Pages](#)[Modules](#)[Classes](#)[Package Functions](#)[Photon](#)[Pun](#)[Namespaces](#) | [Classes](#) | [Typedefs](#) |[Enumerations](#)

Photon.Pun Namespace Reference

Namespaces

namespace	UtilityScripts
-----------	-----------------------

Classes

class **CustomTypes**

Internally used class, containing de/serialization methods for various Unity-specific classes. Adding those to the **Photon** serialization protocol allows you to send them in events, etc.

class **DefaultPool**

The default implementation of a PrefabPool for PUN, which actually Instantiates and Destroys GameObjects but pools a resource. [More...](#)

struct **InstantiateParameters**

interface **IPunInstantiateMagicCallback**

interface **IPunObservable**

Defines the OnPhotonSerializeView method to make it easy to implement correctly for observable scripts. [More...](#)

interface **IPunOwnershipCallbacks**

This interface is used as definition of all callback methods of PUN, except OnPhotonSerializeView. Preferably, implement them individually. [More...](#)

interface **IPunPrefabPool**

Defines an interface for object pooling, used in PhotonNetwork.Instantiate and **PhotonNetwork.Destroy**. [More...](#)

class **MonoBehaviourPun**

This class adds the property photonView, while logging a warning when your game still uses the networkView. [More...](#)

class **MonoBehaviourPunCallbacks**

This class provides a `.photonView` and all callbacks/events that PUN can call. Override the events/methods you want to use. [More...](#)

class **PhotonAnimatorView**

This class helps you to synchronize Mecanim animations. Simply add the component to your `GameObject` and make sure that the **PhotonAnimatorView** is added to the list of observed components. [More...](#)

class **PhotonHandler**

Internal `MonoBehaviour` that allows **Photon** to run an Update loop.

struct **PhotonMessageInfo**

Container class for info about a particular message, RPC or update. [More...](#)

class **PhotonNetwork**

The main class to use the **PhotonNetwork** plugin. This class is static. [More...](#)

class **PhotonRigidbody2DView**

class **PhotonRigidbodyView**

class **PhotonStream**

This container is used in `OnPhotonSerializeView()` to either provide incoming data of a **PhotonView** or for you to provide it. [More...](#)

class **PhotonStreamQueue**

The **PhotonStreamQueue** helps you poll object states at higher frequencies than what **PhotonNetwork.SendRate** dictates and then sends all those states at once when **Serialize()** is called. On the receiving end you can call **Deserialize()** and then the stream will roll out the received

object states in the same order and timeStep they were recorded in. [More...](#)

class **PhotonTransformView**

class **PhotonTransformViewClassic**

This class helps you to synchronize position, rotation and scale of a GameObject. It also gives you many different options to make the synchronized values appear smooth, even when the data is only send a couple of times per second. Simply add the component to your GameObject and make sure that the **PhotonTransformViewClassic** is added to the list of observed components [More...](#)

class **PhotonTransformViewPositionControl**

class **PhotonTransformViewPositionModel**

class **PhotonTransformViewRotationControl**

class **PhotonTransformViewRotationModel**

class **PhotonTransformViewScaleControl**

class **PhotonTransformViewScaleModel**

class **PhotonView**

A **PhotonView** identifies an object across the network (viewID) and configures how the controlling client updates remote instances. [More...](#)

class **PunEvent**

Defines **Photon** event-codes as used by PUN.

class **PunExtensions**

Small number of extension methods that make it easier for PUN to work cross-Unity-versions. [More...](#)

class **PunRPC**

Replacement for RPC attribute with different name. Used to flag methods as remote-callable. [More...](#)

class **SceneManagerHelper**

class **ServerSettings**

Collection of connection-relevant settings, used internally by **PhotonNetwork.ConnectUsingSettings**. [More...](#)

Typedefs

```
using Debug = UnityEngine.Debug
```

```
using Hashtable = ExitGames.Client.Photon.Hashtable
```

```
using SupportClassPun = ExitGames.Client.Photon.SupportClass
```

Enumerations

enum **ConnectMethod**

Which PhotonNetwork method was called to connect (which influences the regions we want pinged). [More...](#)

enum **PunLogLevel**

Used to define the level of logging output created by the PUN classes. Either log errors, info (some more) or full. [More...](#)

enum **RpcTarget**

Enum of "target" options for RPCs. These define which remote clients get your RPC call. [More...](#)

enum **ViewSynchronization**

enum **OwnershipOption**

Options to define how Ownership Transfer is handled per PhotonView. [More...](#)

Enumeration Type Documentation

enum ConnectMethod

strong

Which **PhotonNetwork** method was called to connect (which influences the regions we want pinged).

PhotonNetwork.ConnectUsingSettings will call either **ConnectToMaster**, **ConnectToRegion** or **ConnectToBest**, depending on the settings.

enum OwnershipOption

strong

Options to define how Ownership Transfer is handled per **PhotonView**.

This setting affects how **RequestOwnership** and **TransferOwnership** work at runtime.

Enumerator	
Fixed	Ownership is fixed. Instantiated objects stick with their creator, scene objects always belong to the Master Client.
Takeover	Ownership can be taken away from the current owner who can't object.
Request	Ownership can be requested with PhotonView.RequestOwnership but the current owner has to agree to give up ownership. The current owner has to implement IPunCallbacks.OnOwnershipRequest to react to the ownership request.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

[Main Page](#)[Related Pages](#)[Modules](#)[Classes](#)[Package Functions](#)[Photon](#)[Pun](#)[UtilityScripts](#)[Classes](#)

Photon.Pun.UtilityScripts Namespace Reference

Classes

class **ButtonInsideScrollList**

Button inside scroll list will stop scrolling ability of scrollRect container, so that when pressing down on a button and draggin up and down will not affect scrolling. this doesn't do anything if no scrollRect component found in Parent Hierarchy. [More...](#)

class **CellTree**

Represents the tree accessible from its root node. [More...](#)

class **CellTreeNode**

Represents a single node of the tree. [More...](#)

class **ConnectAndJoinRandom**

Simple component to call ConnectUsingSettings and to get into a PUN room easily. [More...](#)

class **CountdownTimer**

This is a basic **CountdownTimer**. In order to start the timer, the MasterClient can add a certain entry to the Custom Room Properties, which contains the property's name 'StartTime' and the actual start time describing the moment, the timer has been started. To have a synchronized timer, the best practice is to use **PhotonNetwork.Time**. In order to subscribe to the CountdownTimerHasExpired event you can call **CountdownTimer.OnCountdownTimerHasExpired += OnCountdownTimerIsExpired;** from Unity's OnEnable function for example. For unsubscribing simply call **CountdownTimer.OnCountdownTimerHasExpired -= OnCountdownTimerIsExpired;**. You can do this from Unity's OnDisable function for example. [More...](#)

class **CullArea**

Represents the cull area used for network culling. [More...](#)

class **CullingHandler**
Handles the network culling. [More...](#)

class **EventSystemSpawner**
Event system spawner. Will add an EventSystem GameObject with an EventSystem component and a StandaloneInputModule component Use this in additive scene loading context where you would otherwise get a "Multiple eventsystem in scene... this is not supported" error from Unity [More...](#)

class **GraphicToggleIsOnTransition**
Use this on toggles texts to have some color transition on the text depending on the isOn State. [More...](#)

interface **IPunTurnManagerCallbacks**

class **MoveByKeys**
Very basic component to move a GameObject by WASD and Space. [More...](#)

class **OnClickDestroy**
Destroys the networked GameObject either by **PhotonNetwork.Destroy** or by sending an RPC which calls Object.Destroy(). [More...](#)

class **OnClickInstantiate**
Instantiates a networked GameObject on click. [More...](#)

class **OnClickRpc**
This component will instantiate a network GameObject when in a room and the user click on that component's GameObject. Uses PhysicsRaycaster for positioning. [More...](#)

class **OnEscapeQuit**

This component will quit the application when escape key is pressed [More...](#)

class **OnJoinedInstantiate**
This component will instantiate a network GameObject when a room is joined [More...](#)

class **OnPointerOverTooltip**
Set focus to a given photonView when pointed is over [More...](#)

class **OnStartDelete**
This component will destroy the GameObject it is attached to (in Start()). [More...](#)

class **PhotonLagSimulationGui**
This MonoBehaviour is a basic GUI for the **Photon** client's network-simulation feature. It can modify lag (fixed delay), jitter (random lag) and packet loss. [More...](#)

class **PhotonStatsGui**
Basic GUI to show traffic and health statistics of the connection to **Photon**, toggled by shift+tab. [More...](#)

class **PlayerNumbering**
Implements consistent numbering in a room/game with help of room properties. Access them by Player.GetPlayerNumber() extension. [More...](#)

class **PlayerNumberingExtensions**
Extension used for PlayerRoomIndexing and Player class. [More...](#)

class **PointedAtGameObjectInfo**
Display ViewId, OwnerActorNr, IsCeneView and IsMine when clicked. [More...](#)

class **PunPlayerScores**
Scoring system for PhotonPlayer [More...](#)

class **PunTeams**
Implements teams in a room/game with help of player properties. Access them by Player.GetTeam extension. [More...](#)

class **PunTurnManager**
Pun turnBased Game manager. Provides an Interface (**IPunTurnManagerCallbacks**) for the typical turn flow and logic, between players Provides Extensions for Player, Room and RoomInfo to feature dedicated api for TurnBased Needs [More...](#)

class **ScoreExtensions**

class **SmoothSyncMovement**
Smoothed out movement for network gameobjects [More...](#)

class **StatesGui**
Output detailed information about **Pun** Current states, using the old Unity UI framework. [More...](#)

class **TabViewManager**
Tab view manager. Handles **Tab** views activation and deactivation, and provides a Unity Event Callback when a tab was selected. [More...](#)

class **TeamExtensions**
Extension used for **PunTeams** and Player class. Wraps access to the player's custom property. [More...](#)

class **TextButtonTransition**
Use this on Button texts to have some color transition on the text as well without corrupting button's behaviour. [More...](#)

class **TextToggleIsOnTransition**

Use this on toggles texts to have some color transition on the text depending on the isOn State. [More...](#)

class **TurnExtensions**



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	ButtonInsideScrollList	

[List of all members](#)

ButtonInsideScrollList Class Reference

Button inside scroll list will stop scrolling ability of scrollRect container, so that when pressing down on a button and dragging up and down will not affect scrolling. this doesn't do anything if no scrollRect component found in Parent Hierarchy. [More...](#)

Inherits MonoBehaviour, IPointerDownHandler, and IPointerUpHandler.

Detailed Description

Button inside scroll list will stop scrolling ability of scrollRect container, so that when pressing down on a button and draggin up and down will not affect scrolling. this doesn't do anything if no scrollRect component found in Parent Hierarchy.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	CellTree			

CellTree Class Reference

[Public Member Functions](#) | [Properties](#) | [List of all members](#)

Represents the tree accessible from its root node. [More...](#)

Public Member Functions

CellTree ()

Default constructor. [More...](#)

CellTree (CellTreeNode root)

Constructor to define the root node. [More...](#)

Properties

CellTreeNode **RootNode** [get]

Represents the root node of the cell tree. [More...](#)

Detailed Description

Represents the tree accessible from its root node.

Constructor & Destructor Documentation

CellTree ()

Default constructor.

CellTree (CellTreeNode **root**)

Constructor to define the root node.

Parameters

root The root node of the tree.

Property Documentation

CellTreeNode RootNode

get

Represents the root node of the cell tree.



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	CellTreeNode			

CellTreeNode Class Reference

[Public Types](#) | [Public Member Functions](#) |
[Public Attributes](#) | [List of all members](#)

Represents a single node of the tree. [More...](#)

Public Types

enum	ENodeType
------	------------------

Public Member Functions

CellTreeNode ()

Default constructor. [More...](#)

CellTreeNode (byte id, ENodeType nodeType, **CellTreeNode** parent)

Constructor to define the ID and the node type as well as setting a parent node. [More...](#)

void **AddChild** (**CellTreeNode** child)

Adds the given child to the node. [More...](#)

void **Draw** ()

Draws the cell in the editor. [More...](#)

void **GetActiveCells** (List< byte > activeCells, bool yIsUpAxis, Vector3 position)

Gathers all cell IDs the player is currently inside or nearby. [More...](#)

bool **IsPointInsideCell** (bool yIsUpAxis, Vector3 point)

Checks if the given point is inside the cell. [More...](#)

bool **IsPointNearCell** (bool yIsUpAxis, Vector3 point)

Checks if the given point is near the cell. [More...](#)

Public Attributes

byte **Id**

Represents the unique ID of the cell. [More...](#)

Vector3 **Center**

Represents the center, top-left or bottom-right position of the cell or the size of the cell. [More...](#)

ENodeType **NodeType**

Describes the current node type of the cell tree node. [More...](#)

CellTreeNode **Parent**

Reference to the parent node. [More...](#)

List< CellTreeNode > **Childs**

A list containing all child nodes. [More...](#)

Detailed Description

Represents a single node of the tree.

Constructor & Destructor Documentation

CellTreeNode ()

Default constructor.

```
CellTreeNode ( byte      id,  
               ENodeType  nodeType,  
               CellTreeNode parent  
               )
```

Constructor to define the ID and the node type as well as setting a parent node.

Parameters

id The ID of the cell is used as the interest group.
nodeType The node type of the cell tree node.
parent The parent node of the cell tree node.

Member Function Documentation

void AddChild (CellTreeNode **child)**

Adds the given child to the node.

Parameters

child The child which is added to the node.

void Draw ()

Draws the cell in the editor.

void GetActiveCells (List< byte > **activeCells,
 bool **ylsUpAxis**,
 Vector3 **position**
)**

Gathers all cell IDs the player is currently inside or nearby.

Parameters

activeCells The list to add all cell IDs to the player is currently inside or nearby.

ylsUpAxis Describes if the y-axis is used as up-axis.

position The current position of the player.

bool IsPointInsideCell (bool **ylsUpAxis,
 Vector3 **point**
)**

Checks if the given point is inside the cell.

Parameters

yIsUpAxis Describes if the y-axis is used as up-axis.

point The point to check.

Returns

True if the point is inside the cell, false if the point is not inside the cell.

```
bool IsPointNearCell ( bool    yIsUpAxis,  
                      Vector3 point  
                      )
```

Checks if the given point is near the cell.

Parameters

yIsUpAxis Describes if the y-axis is used as up-axis.

point The point to check.

Returns

True if the point is near the cell, false if the point is too far away.

Member Data Documentation

Vector3 Center

Represents the center, top-left or bottom-right position of the cell or the size of the cell.

List<CellTreeNode> Childs

A list containing all child nodes.

byte Id

Represents the unique ID of the cell.

ENodeType NodeType

Describes the current node type of the cell tree node.

CellTreeNode Parent

Reference to the parent node.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	ConnectAndJoinRandom	

[Public Member Functions](#) | [Public Attributes](#) |

[List of all members](#)

ConnectAndJoinRandom Class Reference

Simple component to call `ConnectUsingSettings` and to get into a PUN room easily. [More...](#)

Inherits **MonoBehaviourPunCallbacks**.

Public Member Functions

void **Start** ()

void **ConnectNow** ()

override void **OnConnectedToMaster** ()

Called when the client is connected to the Master Server and ready for matchmaking and other tasks. [More...](#)

override void **OnJoinedLobby** ()

Called on entering a lobby on the Master Server. The actual room-list updates will call OnRoomListUpdate. [More...](#)

override void **OnJoinRandomFailed** (short returnCode, string message)

Called when a previous OpJoinRandom call failed on the server. [More...](#)

override void **OnDisconnected** (**DisconnectCause** cause)

Called after disconnecting from the **Photon** server. It could be a failure or intentional [More...](#)

override void **OnJoinedRoom** ()

Called when the LoadBalancingClient entered a room, no matter if this client created it or simply joined. [More...](#)

‣ Public Member Functions inherited from **MonoBehaviourPunCallbacks**

virtual void **OnEnable** ()

virtual void **OnDisable** ()

virtual void **OnConnected ()**
Called to signal that the raw connection got established but before the client can call operation on the server. [More...](#)

virtual void **OnLeftRoom ()**
Called when the local user/client left a room, so the game's logic can clean up it's internal state. [More...](#)

virtual void **OnMasterClientSwitched (Player newMasterClient)**
Called after switching to a new MasterClient when the current one leaves. [More...](#)

virtual void **OnCreateRoomFailed** (short returnCode, string message)
Called when the server couldn't create a room (OpCreateRoom failed). [More...](#)

virtual void **OnJoinRoomFailed** (short returnCode, string message)
Called when a previous OpJoinRoom call failed on the server. [More...](#)

virtual void **OnCreatedRoom ()**
Called when this client created a room and entered it. **OnJoinedRoom()** will be called as well. [More...](#)

virtual void **OnLeftLobby ()**
Called after leaving a lobby. [More...](#)

virtual void **OnRegionListReceived (RegionHandler regionHandler)**
Called when the Name Server provided a list of regions for your title. [More...](#)

virtual void **OnRoomListUpdate** (List< **RoomInfo** > roomList)
Called for any update of the room-listing while in a

lobby (InLobby) on the Master Server. [More...](#)

virtual void **OnPlayerEnteredRoom** (**Player** newPlayer)
Called when a remote player entered the room. This Player is already added to the playerlist. [More...](#)

virtual void **OnPlayerLeftRoom** (**Player** otherPlayer)
Called when a remote player left the room or became inactive. Check otherPlayer.IsInactive. [More...](#)

virtual void **OnRoomPropertiesUpdate** (Hashtable propertiesThatChanged)
Called when a room's custom properties changed. The propertiesThatChanged contains all that was set via **Room.SetCustomProperties**. [More...](#)

virtual void **OnPlayerPropertiesUpdate** (**Player** target, Hashtable changedProps)
Called when custom player-properties are changed. Player and the changed properties are passed as object[]. [More...](#)

virtual void **OnFriendListUpdate** (List< **FriendInfo** > friendList)
Called when the server sent the response to a FindFriends request. [More...](#)

virtual void **OnCustomAuthenticationResponse** (Dictionary< string, object > data)
Called when your Custom Authentication service responds with additional data. [More...](#)

virtual void **OnCustomAuthenticationFailed** (string debugMessage)
Called when the custom authentication failed. Followed by disconnect! [More...](#)

virtual void **OnWebRpcResponse** (OperationResponse response)

virtual void **OnLobbyStatisticsUpdate** (List< **TypedLobbyInfo** > lobbyStatistics)

Called when the Master Server sent an update for the Lobby Statistics, updating PhotonNetwork.LobbyStatistics. [More...](#)

Public Attributes

bool **AutoConnect** = true

Connect automatically? If false you can set this to true later on or call `ConnectUsingSettings` in your own scripts. [More...](#)

byte **Version** = 1

Used as **PhotonNetwork.GameVersion**. [More...](#)

Additional Inherited Members

‣ Properties inherited from **MonoBehaviourPun**

PhotonView **photonView** [get]

A cached reference to a **PhotonView** on this
GameObject. [More...](#)

Detailed Description

Simple component to call `ConnectUsingSettings` and to get into a PUN room easily.

A custom inspector provides a button to connect in PlayMode, should `AutoConnect` be false.

Member Function Documentation

override void OnConnectedToMaster ()

virtual

Called when the client is connected to the Master Server and ready for matchmaking and other tasks.

The list of available rooms won't become available unless you join a lobby via **LoadBalancingClient.OpJoinLobby**. You can join rooms and create them even without being in a lobby. The default lobby is used in that case.

Reimplemented from **MonoBehaviourPunCallbacks**.

override void OnDisconnected (DisconnectCause **cause)**

virtual

Called after disconnecting from the **Photon** server. It could be a failure or intentional

The reason for this disconnect is provided as DisconnectCause.

Reimplemented from **MonoBehaviourPunCallbacks**.

override void OnJoinedLobby ()

virtual

Called on entering a lobby on the Master Server. The actual room-list updates will call OnRoomListUpdate.

While in the lobby, the roomlist is automatically updated in fixed intervals (which you can't modify in the public cloud). The room list gets available via OnRoomListUpdate.

Reimplemented from **MonoBehaviourPunCallbacks**.

override void OnJoinedRoom ()

virtual

Called when the LoadBalancingClient entered a room, no matter if this client created it or simply joined.

When this is called, you can access the existing players in **Room.Players**, their custom properties and **Room.CustomProperties**.

In this callback, you could create player objects. For example in Unity, instantiate a prefab for the player.

If you want a match to be started "actively", enable the user to signal "ready" (using OpRaiseEvent or a Custom Property).

Reimplemented from **MonoBehaviourPunCallbacks**.

override void OnJoinRandomFailed (short **returnCode**, string **message**)

virtual

Called when a previous OpJoinRandom call failed on the server.

The most common causes are that a room is full or does not exist (due to someone else being faster or closing the room).

When using multiple lobbies (via OpJoinLobby or a TypedLobby parameter), another lobby might have more/fitting rooms.

Parameters

returnCode Operation ReturnCode from the server.

message Debug message for the error.

Reimplemented from **MonoBehaviourPunCallbacks**.

Member Data Documentation

bool AutoConnect = true

Connect automatically? If false you can set this to true later on or call `ConnectUsingSettings` in your own scripts.

byte Version = 1

Used as `PhotonNetwork.GameVersion`.



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List	Class Index	Class Hierarchy		Class Members		
Photon	Pun	UtilityScripts	CountdownTimer			

[Public Member Functions](#) | [Public Attributes](#) |
[Events](#) | [List of all members](#)

CountdownTimer Class Reference

This is a basic **CountdownTimer**. In order to start the timer, the MasterClient can add a certain entry to the Custom Room Properties, which contains the property's name 'StartTime' and the actual start time describing the moment, the timer has been started. To have a synchronized timer, the best practice is to use **PhotonNetwork.Time**. In order to subscribe to the CountdownTimerHasExpired event you can call **CountdownTimer.OnCountdownTimerHasExpired += OnCountdownTimerIsExpired;** from Unity's OnEnable function for example. For unsubscribing simply call **CountdownTimer.OnCountdownTimerHasExpired -= OnCountdownTimerIsExpired;** You can do this from Unity's OnDisable function for example. [More...](#)

Inherits **MonoBehaviourPunCallbacks**.

Public Member Functions

delegate void **CountdownTimerHasExpired** ()
OnCountdownTimerHasExpired delegate. [More...](#)

void **Start** ()

void **Update** ()

override void **OnRoomPropertiesUpdate** (Hashtable propertiesThatChanged)
Called when a room's custom properties changed. The propertiesThatChanged contains all that was set via **Room.SetCustomProperties**. [More...](#)

› Public Member Functions inherited from MonoBehaviourPunCallbacks

virtual void **OnEnable** ()

virtual void **OnDisable** ()

virtual void **OnConnected** ()
Called to signal that the raw connection got established but before the client can call operation on the server. [More...](#)

virtual void **OnLeftRoom** ()
Called when the local user/client left a room, so the game's logic can clean up it's internal state. [More...](#)

virtual void **OnMasterClientSwitched** (**Player** newMasterClient)
Called after switching to a new MasterClient when the current one leaves. [More...](#)

virtual void **OnCreateRoomFailed** (short returnCode, string message)

Called when the server couldn't create a room (OpCreateRoom failed). [More...](#)

virtual void **OnJoinRoomFailed** (short returnCode, string message)
Called when a previous OpJoinRoom call failed on the server. [More...](#)

virtual void **OnCreatedRoom** ()
Called when this client created a room and entered it. **OnJoinedRoom()** will be called as well. [More...](#)

virtual void **OnJoinedLobby** ()
Called on entering a lobby on the Master Server. The actual room-list updates will call OnRoomListUpdate. [More...](#)

virtual void **OnLeftLobby** ()
Called after leaving a lobby. [More...](#)

virtual void **OnDisconnected** (DisconnectCause cause)
Called after disconnecting from the **Photon** server. It could be a failure or intentional [More...](#)

virtual void **OnRegionListReceived** (RegionHandler regionHandler)
Called when the Name Server provided a list of regions for your title. [More...](#)

virtual void **OnRoomListUpdate** (List< **RoomInfo** > roomList)
Called for any update of the room-listing while in a lobby (InLobby) on the Master Server. [More...](#)

virtual void **OnJoinedRoom** ()
Called when the LoadBalancingClient entered a room, no matter if this client created it or simply joined. [More...](#)

virtual void **OnPlayerEnteredRoom** (**Player** newPlayer)
Called when a remote player entered the room. This Player is already added to the playerlist. [More...](#)

virtual void **OnPlayerLeftRoom** (**Player** otherPlayer)
Called when a remote player left the room or became inactive. Check otherPlayer.IsInactive. [More...](#)

virtual void **OnJoinRandomFailed** (short returnCode, string message)
Called when a previous OpJoinRandom call failed on the server. [More...](#)

virtual void **OnConnectedToMaster** ()
Called when the client is connected to the Master Server and ready for matchmaking and other tasks. [More...](#)

virtual void **OnPlayerPropertiesUpdate** (**Player** target, Hashtable changedProps)
Called when custom player-properties are changed. Player and the changed properties are passed as object[]. [More...](#)

virtual void **OnFriendListUpdate** (List< **FriendInfo** > friendList)
Called when the server sent the response to a FindFriends request. [More...](#)

virtual void **OnCustomAuthenticationResponse** (Dictionary< string, object > data)
Called when your Custom Authentication service responds with additional data. [More...](#)

virtual void **OnCustomAuthenticationFailed** (string debugMessage)
Called when the custom authentication failed.

Followed by disconnect! [More...](#)

virtual void **OnWebRpcResponse** (OperationResponse response)

virtual void **OnLobbyStatisticsUpdate** (List< **TypedLobbyInfo** > lobbyStatistics)

Called when the Master Server sent an update for the Lobby Statistics, updating PhotonNetwork.LobbyStatistics. [More...](#)

Public Attributes

```
const string CountdownStartTime = "StartTime"
```

```
Text Text
```

```
float Countdown = 5.0f
```

Events

static CountdownTimerHasExpired	OnCountdownTimerHasExpire Called when the timer has expired. More...
--	--

Additional Inherited Members

‣ Properties inherited from **MonoBehaviourPun**

PhotonView **photonView** [get]

A cached reference to a **PhotonView** on this
GameObject. [More...](#)

Detailed Description

This is a basic **CountdownTimer**. In order to start the timer, the MasterClient can add a certain entry to the Custom Room Properties, which contains the property's name 'StartTime' and the actual start time describing the moment, the timer has been started. To have a synchronized timer, the best practice is to use **PhotonNetwork.Time**. In order to subscribe to the CountdownTimerHasExpired event you can call **CountdownTimer.OnCountdownTimerHasExpired += OnCountdownTimerIsExpired;** from Unity's OnEnable function for example. For unsubscribing simply call **CountdownTimer.OnCountdownTimerHasExpired -= OnCountdownTimerIsExpired;** You can do this from Unity's OnDisable function for example.

Member Function Documentation

delegate void CountdownTimerHasExpired ()

OnCountdownTimerHasExpired delegate.

**override void
OnRoomPropertiesUpdate (Hashtable **propertiesThatChanged**)**

Called when a room's custom properties changed. The **propertiesThatChanged** contains all that was set via **Room.SetCustomProperties**.

Since v1.25 this method has one parameter: Hashtable **propertiesThatChanged**.

Changing properties must be done by **Room.SetCustomProperties**, which causes this callback locally, too.

Parameters

propertiesThatChanged

Reimplemented from **MonoBehaviourPunCallbacks**.

Event Documentation

CountdownTimerHasExpired
OnCountdownTimerHasExpired

static

Called when the timer has expired.



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	CullArea			

CullArea Class Reference

[Public Member Functions](#) | [Public Attributes](#) | [Properties](#) | [List of all members](#)

Represents the cull area used for network culling. [More...](#)

Inherits MonoBehaviour.

Public Member Functions

void **OnDrawGizmos** ()

Creates the cell hierarchy in editor and draws the cell view. [More...](#)

List< byte > **GetActiveCells** (Vector3 position)

Gets a list of all cell IDs the player is currently inside or nearby. [More...](#)

Public Attributes

const int **MAX_NUMBER_OF_SUBDIVISIONS** = 3

readonly byte **FIRST_GROUP_ID** = 1

This represents the first ID which is assigned to the first created cell. If you already have some interest groups blocking this first ID, feel free to change it. However increasing the first group ID decreases the maximum amount of allowed cells. Allowed values are in range from 1 to 250. [More...](#)

readonly int[] **SUBDIVISION_FIRST_LEVEL_ORDER** = new int[4] { 0, 1, 1, 1 }

This represents the order in which updates are sent. The number represents the subdivision of the cell hierarchy: [More...](#)

readonly int[] **SUBDIVISION_SECOND_LEVEL_ORDER** = new int[8] { 0, 2, 1, 2, 0, 2, 1, 2 }

This represents the order in which updates are sent. The number represents the subdivision of the cell hierarchy: [More...](#)

readonly int[] **SUBDIVISION_THIRD_LEVEL_ORDER** = new int[12] { 0, 3, 2, 3, 1, 3, 2, 3, 1, 3, 2, 3 }

This represents the order in which updates are sent. The number represents the subdivision of the cell hierarchy: [More...](#)

Vector2 **Center**

Vector2 **Size** = new Vector2(25.0f, 25.0f)

Vector2[] **Subdivisions** = new Vector2[MAX_NUMBER_OF_SUBDIVISIONS]

int **NumberOfSubdivisions**

bool **YIsUpAxis** = false

bool **RecreateCellHierarchy** = false

Properties

int **CellCount** [get]

CellTree **CellTree** [get]

Dictionary< int, GameObject > **Map** [get]

Detailed Description

Represents the cull area used for network culling.

Member Function Documentation

List<byte> GetActiveCells (Vector3 **position)**

Gets a list of all cell IDs the player is currently inside or nearby.

Parameters

position The current position of the player.

Returns

A list containing all cell IDs the player is currently inside or nearby.

void OnDrawGizmos ()

Creates the cell hierarchy in editor and draws the cell view.

Member Data Documentation

readonly byte FIRST_GROUP_ID = 1

This represents the first ID which is assigned to the first created cell. If you already have some interest groups blocking this first ID, feel free to change it. However increasing the first group ID decreases the maximum amount of allowed cells. Allowed values are in range from 1 to 250.

**readonly int [] SUBDIVISION_FIRST_LEVEL_ORDER = new int[4]
{ 0, 1, 1, 1 }**

This represents the order in which updates are sent. The number represents the subdivision of the cell hierarchy:

- 0: message is sent to all players
- 1: message is sent to players who are interested in the matching cell of the first subdivision If there is only one subdivision we are sending one update to all players before sending three consequent updates only to players who are in the same cell or interested in updates of the current cell.

**readonly int [] SUBDIVISION_SECOND_LEVEL_ORDER = new
int[8] { 0, 2, 1, 2, 0, 2, 1, 2 }**

This represents the order in which updates are sent. The number represents the subdivision of the cell hierarchy:

- 0: message is sent to all players
- 1: message is sent to players who are interested in the matching cell of the first subdivision
- 2: message is sent to players who are interested in the

matching cell of the second subdivision If there are two subdivisions we are sending every second update only to players who are in the same cell or interested in updates of the current cell.

```
readonly int [] SUBDIVISION_THIRD_LEVEL_ORDER = new  
int[12] { 0, 3, 2, 3, 1, 3, 2, 3, 1, 3, 2, 3 }
```

This represents the order in which updates are sent. The number represents the subdivision of the cell hierarchy:

- 0: message is sent to all players
- 1: message is sent to players who are interested in the matching cell of the first subdivision
- 2: message is sent to players who are interested in the matching cell of the second subdivision
- 3: message is sent to players who are interested in the matching cell of the third subdivision If there are two subdivisions we are sending every second update only to players who are in the same cell or interested in updates of the current cell.



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List	Class Index	Class Hierarchy		Class Members		
Photon	Pun	UtilityScripts	CullingHandler			

[Public Member Functions](#) | [List of all members](#)

CullingHandler Class Reference

Handles the network culling. [More...](#)

Inherits MonoBehaviour, and **IPunObservable**.

Public Member Functions

void **OnPhotonSerializeView** (**PhotonStream** stream,
PhotonMessageInfo info)

This time **OnPhotonSerializeView** is not used to send or receive any kind of data. It is used to change the currently active group of the **PhotonView** component, making it work together with PUN more directly. Keep in mind that this function is only executed, when there is at least one more player in the room.
[More...](#)

Detailed Description

Handles the network culling.

Member Function Documentation

```
void OnPhotonSerializeView ( PhotonStream      stream,  
                             PhotonMessageInfo info  
                             )
```

This time OnPhotonSerializeView is not used to send or receive any kind of data. It is used to change the currently active group of the **PhotonView** component, making it work together with PUN more directly. Keep in mind that this function is only executed, when there is at least one more player in the room.

Implements **IPunObservable**.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	EventSystemSpawner	

[List of all members](#)

EventSystemSpawner Class Reference

Event system spawner. Will add an EventSystem GameObject with an EventSystem component and a StandaloneInputModule component. Use this in additive scene loading context where you would otherwise get a "Multiple eventsystem in scene... this is not supported" error from Unity [More...](#)

Inherits MonoBehaviour.

Detailed Description

Event system spawner. Will add an EventSystem GameObject with an EventSystem component and a StandaloneInputModule component
Use this in additive scene loading context where you would otherwise get a "Multiple eventsystem in scene... this is not supported" error from Unity

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	GraphicToggleIsOnTransition	

[Public Member Functions](#) | [Public Attributes](#) |

[List of all members](#)

GraphicToggleIsOnTransition Class Reference

Use this on toggles texts to have some color transition on the text depending on the isOn State. [More...](#)

Inherits MonoBehaviour, IPointerEnterHandler, and IPointerExitHandler.

Public Member Functions

void **OnPointerEnter** (PointerEventData eventData)

void **OnPointerExit** (PointerEventData eventData)

void **OnEnable** ()

void **OnDisable** ()

void **OnValueChanged** (bool isOn)

Public Attributes

Toggle **toggle**

Color **NormalOnColor** = Color.white

Color **NormalOffColor** = Color.black

Color **HoverOnColor** = Color.black

Color **HoverOffColor** = Color.black

Detailed Description

Use this on toggles texts to have some color transition on the text depending on the isOn State.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	IPunTurnManagerCallbacks	

[Public Member Functions](#) | [List of all members](#)

IPunTurnManagerCallbacks Interface Reference

Public Member Functions

void **OnTurnBegins** (int turn)
Called the turn begins event. [More...](#)

void **OnTurnCompleted** (int turn)
Called when a turn is completed (finished by all players) [More...](#)

void **OnPlayerMove** (**Player** player, int turn, object move)
Called when a player moved (but did not finish the turn) [More...](#)

void **OnPlayerFinished** (**Player** player, int turn, object move)
When a player finishes a turn (includes the action/move of that player) [More...](#)

void **OnTurnTimeEnds** (int turn)
Called when a turn completes due to a time constraint (timeout for a turn) [More...](#)

Member Function Documentation

```
void OnPlayerFinished ( Player player,  
                        int    turn,  
                        object move  
                        )
```

When a player finishes a turn (includes the action/move of that player)

Parameters

player Player reference
turn Turn index
move Move Object data

```
void OnPlayerMove ( Player player,  
                   int    turn,  
                   object move  
                   )
```

Called when a player moved (but did not finish the turn)

Parameters

player Player reference
turn Turn Index
move Move Object data

```
void OnTurnBegins ( int turn )
```

Called the turn begins event.

Parameters

turn Turn Index

void OnTurnCompleted (int **turn**)

Called when a turn is completed (finished by all players)

Parameters

turn Turn Index

void OnTurnTimeEnds (int **turn**)

Called when a turn completes due to a time constraint (timeout for a turn)

Parameters

turn Turn index



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	MoveByKeys	

[Public Member Functions](#) | [Public Attributes](#) |

[List of all members](#)

MoveByKeys Class Reference

Very basic component to move a GameObject by WASD and Space.
[More...](#)

Inherits **MonoBehaviourPun**.

Public Member Functions

void **Start** ()

void **FixedUpdate** ()

Public Attributes

float **Speed** = 10f

float **JumpForce** = 200f

float **JumpTimeout** = 0.5f

Additional Inherited Members

‣ Properties inherited from **MonoBehaviourPun**

PhotonView **photonView** [get]

A cached reference to a **PhotonView** on this
GameObject. [More...](#)

Detailed Description

Very basic component to move a GameObject by WASD and Space.

Requires a **PhotonView**. Disables itself on GameObjects that are not owned on Start.

Speed affects movement-speed. JumpForce defines how high the object "jumps". JumpTimeout defines after how many seconds you can jump again.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	OnClickDestroy			

[Public Member Functions](#) | [Public Attributes](#) |

[List of all members](#)

OnClickDestroy Class Reference

Destroys the networked GameObject either by **PhotonNetwork.Destroy** or by sending an RPC which calls `Object.Destroy()`. [More...](#)

Inherits **MonoBehaviourPun**, and **IPointerClickHandler**.

Public Member Functions

IEnumerator **DestroyRpc** ()

Public Attributes

PointerEventData.InputButton	Button
KeyCode	ModifierKey
bool	DestroyByRpc

Additional Inherited Members

‣ Properties inherited from **MonoBehaviourPun**

PhotonView **photonView** [get]

A cached reference to a **PhotonView** on this
GameObject. [More...](#)

Detailed Description

Destroys the networked GameObject either by **PhotonNetwork.Destroy** or by sending an RPC which calls `Object.Destroy()`.

Using an RPC to Destroy a GameObject is typically a bad idea. It allows any player to Destroy a GameObject and may cause errors.

A client has to clean up the server's event-cache, which contains events for Instantiate and buffered RPCs related to the GO.

A buffered RPC gets cleaned up when the sending player leaves the room, so players joining later won't get those buffered RPCs. This in turn, may mean they don't destroy the GO due to coming later.

Vice versa, a GameObject Instantiate might get cleaned up when the creating player leaves a room. This way, the GameObject that a RPC targets might become lost.

It makes sense to test those cases. Many are not breaking errors and you just have to be aware of them.

Gets `OnClick()` calls by Unity's `IPointerClickHandler`. Needs a `PhysicsRaycaster` on the camera. See:
<https://docs.unity3d.com/ScriptReference/EventSystems.IPointerClickHa>



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy		Class Members
Photon	Pun	UtilityScripts	OnClickInstantiate			

OnClickInstantiate Class Reference

[Public Types](#) | [Public Attributes](#) |
[List of all members](#)

Instantiates a networked GameObject on click. [More...](#)

Inherits MonoBehaviour, and IPointerClickHandler.

Public Types

enum **InstantiateOption**

Public Attributes

PointerEventData.InputButton	Button
KeyCode	ModifierKey
GameObject	Prefab

Detailed Description

Instantiates a networked GameObject on click.

Gets OnClick() calls by Unity's IPointerClickHandler. Needs a PhysicsRaycaster on the camera. See:

<https://docs.unity3d.com/ScriptReference/EventSystems.IPointerClickHa>

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	OnClickRpc	

[Public Member Functions](#) | [Public Attributes](#) |

[List of all members](#)

OnClickRpc Class Reference

This component will instantiate a network GameObject when in a room and the user click on that component's GameObject. Uses PhysicsRaycaster for positioning. [More...](#)

Inherits **MonoBehaviourPun**, and **IPointerClickHandler**.

Public Member Functions

void **ClickRpc** ()

IEnumerator **ClickFlash** ()

Public Attributes

PointerEventData.InputButton	Button
KeyCode	ModifierKey
RpcTarget	Target

Additional Inherited Members

‣ Properties inherited from **MonoBehaviourPun**

PhotonView **photonView** [get]

A cached reference to a **PhotonView** on this
GameObject. [More...](#)

Detailed Description

This component will instantiate a network GameObject when in a room and the user click on that component's GameObject. Uses PhysicsRaycaster for positioning.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	OnEscapeQuit	

[Public Member Functions](#) | [List of all members](#)

OnEscapeQuit Class Reference

This component will quit the application when escape key is pressed
[More...](#)

Inherits MonoBehaviour.

Public Member Functions

`void Update ()`

Detailed Description

This component will quit the application when escape key is pressed

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	OnJoinedInstantiate	

[Public Member Functions](#) | [Public Attributes](#) |

[List of all members](#)

OnJoinedInstantiate Class Reference

This component will instantiate a network GameObject when a room is joined [More...](#)

Inherits MonoBehaviour, **ICollectionCallbacks**,
IMatchmakingCallbacks, and **ILobbyCallbacks**.

Public Member Functions

virtual void **OnEnable** ()

virtual void **OnDisable** ()

void **OnJoinedRoom** ()

Called when the LoadBalancingClient entered a room, no matter if this client created it or simply joined. [More...](#)

void **OnConnected** ()

Called to signal that the "low level connection" got established but before the client can call operation on the server. [More...](#)

void **OnCustomAuthenticationResponse** (Dictionary< string, object > data)

Called when your Custom Authentication service responds with additional data. [More...](#)

void **OnCustomAuthenticationFailed** (string debugMessage)

Called when the custom authentication failed. Followed by disconnect! [More...](#)

void **OnConnectedToMaster** ()

Called when the client is connected to the Master Server and ready for matchmaking and other tasks. [More...](#)

void **OnDisconnected** (**DisconnectCause** cause)

Called after disconnecting from the **Photon** server. It could be a failure or an explicit disconnect call [More...](#)

void **OnRegionListReceived** (**RegionHandler** regionHandler)

Called when the Name Server provided a list of regions for your title. [More...](#)

void **OnRoomListUpdate** (List< **RoomInfo** > roomList)
Called for any update of the room-listing while in a lobby (InLobby) on the Master Server. [More...](#)

void **OnFriendListUpdate** (List< **FriendInfo** > friendList)
Called when the server sent the response to a FindFriends request. [More...](#)

void **OnJoinedLobby** ()
Called on entering a lobby on the Master Server. The actual room-list updates will call OnRoomListUpdate. [More...](#)

void **OnLeftLobby** ()
Called after leaving a lobby. [More...](#)

void **OnLobbyStatisticsUpdate** (List< **TypedLobbyInfo** > lobbyStatistics)
Called when the Master Server sent an update for the Lobby Statistics, updating PhotonNetwork.LobbyStatistics. [More...](#)

void **OnCreatedRoom** ()
Called when this client created a room and entered it. **OnJoinedRoom()** will be called as well. [More...](#)

void **OnCreateRoomFailed** (short returnCode, string message)
Called when the server couldn't create a room (OpCreateRoom failed). [More...](#)

void **OnJoinRoomFailed** (short returnCode, string message)
Called when a previous OpJoinRoom call failed on the server. [More...](#)

void **OnJoinRandomFailed** (short returnCode, string

message)

Called when a previous OpJoinRandom call failed on the server. [More...](#)

void **OnLeftRoom ()**

Called when the local user/client left a room, so the game's logic can clean up it's internal state. [More...](#)

Public Attributes

Transform **SpawnPosition**

float **PositionOffset** = 2.0f

GameObject[] **PrefabsToInstantiate**

Detailed Description

This component will instantiate a network GameObject when a room is joined

Member Function Documentation

void OnConnected ()

Called to signal that the "low level connection" got established but before the client can call operation on the server.

After the (low level transport) connection is established, the client will automatically send the Authentication operation, which needs to get a response before the client can call other operations.

Your logic should wait for either: `OnRegionListReceived` or `OnConnectedToMaster`.

This callback is useful to detect if the server can be reached at all (technically). Most often, it's enough to implement **`OnDisconnected(DisconnectCause cause)`** and check for the cause.

This is not called for transitions from the masterserver to game servers.

Implements **`IConnectionCallbacks`**.

void OnConnectedToMaster ()

Called when the client is connected to the Master Server and ready for matchmaking and other tasks.

The list of available rooms won't become available unless you join a lobby via **`LoadBalancingClient.OpJoinLobby`**. You can join rooms and create them even without being in a lobby. The default lobby is used in that case.

Implements **`IConnectionCallbacks`**.

void OnCreatedRoom ()

Called when this client created a room and entered it. **OnJoinedRoom()** will be called as well.

This callback is only called on the client which created a room (see `OpCreateRoom`).

As any client might close (or drop connection) anytime, there is a chance that the creator of a room does not execute `OnCreatedRoom`.

If you need specific room properties or a "start signal", implement `OnMasterClientSwitched()` and make each new `MasterClient` check the room's state.

Implements **`IMatchmakingCallbacks`**.

void OnCreateRoomFailed (short `returnCode`, string `message`)

Called when the server couldn't create a room (`OpCreateRoom` failed).

Creating a room may fail for various reasons. Most often, the room already exists (roomname in use) or the `RoomOptions` clash and it's impossible to create the room.

When creating a room fails on a Game Server: The client will cache the failure internally and returns to the Master Server before it calls the fail-callback. This way, the client is ready to find/create a room at the moment of the callback. In this case, the client skips calling `OnConnectedToMaster` but returning to the Master Server will still call `OnConnected`. Treat callbacks of `OnConnected` as pure information that the client could connect.

Parameters

returnCode Operation ReturnCode from the server.

message Debug message for the error.

Implements **IMatchmakingCallbacks**.

void OnCustomAuthenticationFailed (string debugMessage)

Called when the custom authentication failed. Followed by disconnect!

Custom Authentication can fail due to user-input, bad tokens/secrets. If authentication is successful, this method is not called. Implement **OnJoinedLobby()** or **OnConnectedToMaster()** (as usual).

During development of a game, it might also fail due to wrong configuration on the server side. In those cases, logging the debugMessage is very important.

Unless you setup a custom authentication service for your app (in the [Dashboard](#)), this won't be called!

Parameters

debugMessage Contains a debug message why authentication failed. This has to be fixed during development.

Implements **IConnectionCallbacks**.

void OnCustomAuthenticationResponse (Dictionary< string, object > c

Called when your Custom Authentication service responds with additional data.

Custom Authentication services can include some custom data in their

response. When present, that data is made available in this callback as Dictionary. While the keys of your data have to be strings, the values can be either string or a number (in Json). You need to make extra sure, that the value type is the one you expect. Numbers become (currently) int64

Example: void OnCustomAuthenticationResponse(Dictionary<string, object> data) { ... }

<https://doc.photonengine.com/en-us/realtime/current/reference/custom-authentication>

Implements **ICollectionCallbacks**.

void OnDisconnected (DisconnectCause cause)

Called after disconnecting from the **Photon** server. It could be a failure or an explicit disconnect call

The reason for this disconnect is provided as DisconnectCause.

Implements **ICollectionCallbacks**.

void OnFriendListUpdate (List< FriendInfo > friendList)

Called when the server sent the response to a FindFriends request.

After calling OpFindFriends, the Master Server will cache the friend list and send updates to the friend list. The friends includes the name, userId, online state and the room (if any) for each requested user/friend.

Use the friendList to update your UI and store it, if the UI should highlight changes.

Implements **IMatchmakingCallbacks**.

void OnJoinedLobby ()

Called on entering a lobby on the Master Server. The actual room-list updates will call OnRoomListUpdate.

While in the lobby, the roomlist is automatically updated in fixed intervals (which you can't modify in the public cloud). The room list gets available via OnRoomListUpdate.

Implements **ILobbyCallbacks**.

void OnJoinedRoom ()

Called when the LoadBalancingClient entered a room, no matter if this client created it or simply joined.

When this is called, you can access the existing players in **Room.Players**, their custom properties and **Room.CustomProperties**.

In this callback, you could create player objects. For example in Unity, instantiate a prefab for the player.

If you want a match to be started "actively", enable the user to signal "ready" (using OpRaiseEvent or a Custom Property).

Implements **IMatchmakingCallbacks**.

void OnJoinRandomFailed (short **returnCode, string **message**)**

Called when a previous OpJoinRandom call failed on the server.

The most common causes are that a room is full or does not exist (due to someone else being faster or closing the room).

This operation is only ever sent to the Master Server. Once a room is found by the Master Server, the client will head off to the

designated Game Server and use the operation Join on the Game Server.

When using multiple lobbies (via OpJoinLobby or a TypedLobby parameter), another lobby might have more/fitting rooms.

Parameters

returnCode Operation ReturnCode from the server.

message Debug message for the error.

Implements **IMatchmakingCallbacks**.

```
void OnJoinRoomFailed ( short returnCode,  
                        string message  
                        )
```

Called when a previous OpJoinRoom call failed on the server.

Joining a room may fail for various reasons. Most often, the room is full or does not exist anymore (due to someone else being faster or closing the room).

When joining a room fails on a Game Server: The client will cache the failure internally and returns to the Master Server before it calls the fail-callback. This way, the client is ready to find/create a room at the moment of the callback. In this case, the client skips calling OnConnectedToMaster but returning to the Master Server will still call OnConnected. Treat callbacks of OnConnected as pure information that the client could connect.

Parameters

returnCode Operation ReturnCode from the server.

message Debug message for the error.

Implements **IMatchmakingCallbacks**.

```
void OnLeftLobby ( )
```

Called after leaving a lobby.

When you leave a lobby, `OpCreateRoom` and `OpJoinRandomRoom` automatically refer to the default lobby.

Implements **`ILobbyCallbacks`**.

`void OnLeftRoom ()`

Called when the local user/client left a room, so the game's logic can clean up it's internal state.

When leaving a room, the `LoadBalancingClient` will disconnect the Game Server and connect to the Master Server. This wraps up multiple internal actions.

Wait for the callback `OnConnectedToMaster`, before you use lobbies and join or create rooms.

Implements **`IMatchmakingCallbacks`**.

`void OnLobbyStatisticsUpdate (List< TypedLobbyInfo > lobbyStatistics)`

Called when the Master Server sent an update for the Lobby Statistics, updating `PhotonNetwork.LobbyStatistics`.

This callback has two preconditions: `EnableLobbyStatistics` must be set to true, before this client connects. And the client has to be connected to the Master Server, which is providing the info about lobbies.

Implements **`ILobbyCallbacks`**.

`void OnRegionListReceived (RegionHandler regionHandler)`

Called when the Name Server provided a list of regions for your title.

Check the RegionHandler class description, to make use of the provided values.

Parameters

regionHandler The currently used RegionHandler.

Implements **ICollectionCallbacks**.

void OnRoomListUpdate (List< RoomInfo > roomList)

Called for any update of the room-listing while in a lobby (InLobby) on the Master Server.

Each item is a RoomInfo which might include custom properties (provided you defined those as lobby-listed when creating a room). Not all types of lobbies provide a listing of rooms to the client. Some are silent and specialized for server-side matchmaking.

Implements **ILobbyCallbacks**.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	OnPointerOverTooltip	

[List of all members](#)

OnPointerOverTooltip Class Reference

Set focus to a given photonView when pointed is over [More...](#)

Inherits MonoBehaviour, IPointerEnterHandler, and IPointerExitHandler.

Detailed Description

Set focus to a given photonView when pointed is over

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	OnStartDelete	

[List of all members](#)

OnStartDelete Class Reference

This component will destroy the GameObject it is attached to (in Start()). [More...](#)

Inherits MonoBehaviour.

Detailed Description

This component will destroy the GameObject it is attached to (in Start()).

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy		Class Members
Photon	Pun	UtilityScripts	PlayerNumbering			

PlayerNumbering Class Reference

[Public Member Functions](#) | [Public Attributes](#) |
[Static Public Attributes](#) | [Events](#) |
[List of all members](#)

Implements consistent numbering in a room/game with help of room properties. Access them by `Player.GetPlayerNumber()` extension.
[More...](#)

Inherits **MonoBehaviourPunCallbacks**.

Public Member Functions

delegate void **PlayerNumberingChanged ()**
OnPlayerNumberingChanged delegate. Use [More...](#)

void **Awake ()**

override void **OnJoinedRoom ()**
Called when the LoadBalancingClient entered a room, no matter if this client created it or simply joined. [More...](#)

override void **OnLeftRoom ()**
Called when the local user/client left a room, so the game's logic can clean up it's internal state. [More...](#)

override void **OnPlayerEnteredRoom (Player newPlayer)**
Called when a remote player entered the room. This Player is already added to the playerlist. [More...](#)

override void **OnPlayerLeftRoom (Player otherPlayer)**
Called when a remote player left the room or became inactive. Check otherPlayer.IsInactive. [More...](#)

override void **OnPlayerPropertiesUpdate (Player targetPlayer, Hashtable changedProps)**
Called when custom player-properties are changed. Player and the changed properties are passed as object[]. [More...](#)

void **RefreshData ()**
Internal call Refresh the cached data and call the OnPlayerNumberingChanged delegate. [More...](#)

► **Public Member Functions inherited from MonoBehaviourPunCallbacks**

virtual void **OnEnable** ()

virtual void **OnDisable** ()

virtual void **OnConnected** ()

Called to signal that the raw connection got established but before the client can call operation on the server. [More...](#)

virtual void **OnMasterClientSwitched** (**Player** newMasterClient)

Called after switching to a new MasterClient when the current one leaves. [More...](#)

virtual void **OnCreateRoomFailed** (short returnCode, string message)

Called when the server couldn't create a room (OpCreateRoom failed). [More...](#)

virtual void **OnJoinRoomFailed** (short returnCode, string message)

Called when a previous OpJoinRoom call failed on the server. [More...](#)

virtual void **OnCreatedRoom** ()

Called when this client created a room and entered it. **OnJoinedRoom()** will be called as well. [More...](#)

virtual void **OnJoinedLobby** ()

Called on entering a lobby on the Master Server. The actual room-list updates will call OnRoomListUpdate. [More...](#)

virtual void **OnLeftLobby** ()

Called after leaving a lobby. [More...](#)

virtual void **OnDisconnected** (**DisconnectCause** cause)

Called after disconnecting from the **Photon** server. It

could be a failure or intentional [More...](#)

virtual void **OnRegionListReceived (RegionHandler regionHandler)**
Called when the Name Server provided a list of regions for your title. [More...](#)

virtual void **OnRoomListUpdate (List< RoomInfo > roomList)**
Called for any update of the room-listing while in a lobby (InLobby) on the Master Server. [More...](#)

virtual void **OnJoinRandomFailed** (short returnCode, string message)
Called when a previous OpJoinRandom call failed on the server. [More...](#)

virtual void **OnConnectedToMaster ()**
Called when the client is connected to the Master Server and ready for matchmaking and other tasks. [More...](#)

virtual void **OnRoomPropertiesUpdate** (Hashtable propertiesThatChanged)
Called when a room's custom properties changed. The propertiesThatChanged contains all that was set via **Room.SetCustomProperties**. [More...](#)

virtual void **OnFriendListUpdate** (List< FriendInfo > friendList)
Called when the server sent the response to a FindFriends request. [More...](#)

virtual void **OnCustomAuthenticationResponse** (Dictionary< string, object > data)
Called when your Custom Authentication service responds with additional data. [More...](#)

virtual void **OnCustomAuthenticationFailed** (string

debugMessage)

Called when the custom authentication failed.

Followed by disconnect! [More...](#)

virtual void **OnWebRpcResponse** (OperationResponse response)

virtual void **OnLobbyStatisticsUpdate** (List< **TypedLobbyInfo** > lobbyStatistics)

Called when the Master Server sent an update for the Lobby Statistics, updating PhotonNetwork.LobbyStatistics. [More...](#)

Public Attributes

const string **RoomPlayerIndexedProp** = "pNr"
Defines the room custom property name to use for room player indexing tracking. [More...](#)

bool **dontDestroyOnLoad** = false
dont destroy on load flag for this Component's GameObject to survive Level Loading. [More...](#)

Static Public Attributes

static PlayerNumbering **instance**

The instance. EntryPoint to query about Room Indexing. [More...](#)

static Player[] **SortedPlayers**

Events

static PlayerNumberingChanged	OnPlayerNumberingChanged Called everytime the room Indexing was updated. Use this for discrete updates. Always better than brute force calls every frame. More...
--------------------------------------	---

Additional Inherited Members

‣ Properties inherited from **MonoBehaviourPun**

PhotonView **photonView** [get]

A cached reference to a **PhotonView** on this
GameObject. [More...](#)

Detailed Description

Implements consistent numbering in a room/game with help of room properties. Access them by `Player.GetPlayerNumber()` extension.

indexing ranges from 0 to the maximum number of Players. indexing remains for the player while in room. If a Player is numbered 2 and player numbered 1 leaves, numbered 1 become vacant and will assigned to the future player joining (the first available vacant number is assigned when joining)

Member Function Documentation

override void OnJoinedRoom ()

virtual

Called when the LoadBalancingClient entered a room, no matter if this client created it or simply joined.

When this is called, you can access the existing players in **Room.Players**, their custom properties and **Room.CustomProperties**.

In this callback, you could create player objects. For example in Unity, instantiate a prefab for the player.

If you want a match to be started "actively", enable the user to signal "ready" (using OpRaiseEvent or a Custom Property).

Reimplemented from **MonoBehaviourPunCallbacks**.

override void OnLeftRoom ()

virtual

Called when the local user/client left a room, so the game's logic can clean up it's internal state.

When leaving a room, the LoadBalancingClient will disconnect the Game Server and connect to the Master Server. This wraps up multiple internal actions.

Wait for the callback OnConnectedToMaster, before you use lobbies and join or create rooms.

Reimplemented from **MonoBehaviourPunCallbacks**.

(Player **newPlayer**)

override void OnPlayerEnteredRoom

virtual

Called when a remote player entered the room. This Player is already added to the playerlist.

If your game starts with a certain number of players, this callback can be useful to check the Room.playerCount and find out if you can start.

Reimplemented from **MonoBehaviourPunCallbacks**.

override void OnPlayerLeftRoom (Player otherPlayer)

virtual

Called when a remote player left the room or became inactive. Check otherPlayer.IsInactive.

If another player leaves the room or if the server detects a lost connection, this callback will be used to notify your game logic.

Depending on the room's setup, players may become inactive, which means they may return and retake their spot in the room. In such cases, the Player stays in the **Room.Players** dictionary.

If the player is not just inactive, it gets removed from the **Room.Players** dictionary, before the callback is called.

Reimplemented from **MonoBehaviourPunCallbacks**.

override void OnPlayerPropertiesUpdate (Player target, Hashtable changedProps)

virtual

Called when custom player-properties are changed. Player and the changed properties are passed as object[].

Changing properties must be done by

Player.SetCustomProperties, which causes this callback locally, too.

Parameters

targetPlayer Contains Player that changed.

changedProps Contains the properties that changed.

Reimplemented from **MonoBehaviourPunCallbacks**.

delegate void PlayerNumberingChanged ()

OnPlayerNumberingChanged delegate. Use

void RefreshData ()

Internal call Refresh the cached data and call the OnPlayerNumberingChanged delegate.

Member Data Documentation

bool dontDestroyOnLoad = false

dont destroy on load flag for this Component's GameObject to survive Level Loading.

PlayerNumbering instance

static

The instance. EntryPoint to query about Room Indexing.

const string RoomPlayerIndexedProp = "pNr"

Defines the room custom property name to use for room player indexing tracking.

Event Documentation

PlayerNumberingChanged OnPlayerNumberingChanged

static

Called everytime the room Indexing was updated. Use this for discrete updates. Always better than brute force calls every frame.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	PlayerNumberingExtensions	

[Static Public Member Functions](#) |

[List of all members](#)

PlayerNumberingExtensions Class Reference

Extension used for PlayerRoomIndexing and Player class. [More...](#)

Static Public Member Functions

static int **GetPlayerNumber** (this **Player** player)

Extension for Player class to wrap up access to the player's custom property. Make sure you use the delegate 'OnPlayerNumberingChanged' to know when you can query the PlayerNumber. Numbering can change over time or not be yet assigned during the initial phase (when player creates a room for example) [More...](#)

static void **SetPlayerNumber** (this **Player** player, int playerNumber)

Sets the player number. It's not recommended to manually interfere with the playerNumbering, but possible. [More...](#)

Detailed Description

Extension used for PlayerRoomIndexing and Player class.

Member Function Documentation

static int GetPlayerNumber (this Player **player)**

static

Extension for Player class to wrap up access to the player's custom property. Make sure you use the delegate 'OnPlayerNumberingChanged' to know when you can query the PlayerNumber. Numbering can change over time or not be yet assigned during the initial phase (when player creates a room for example)

Returns

persistent index in room. -1 for no indexing

static void SetPlayerNumber (this Player **player,
int **playerNumber**
)**

static

Sets the player number. It's not recommended to manually interfere with the playerNumbering, but possible.

Parameters

player Player.
playerNumber Player number.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	PointedAtGameObjectInfo	

[Public Member Functions](#) | [Public Attributes](#) |

[Static Public Attributes](#) | [List of all members](#)

PointedAtGameObjectInfo Class Reference

Display ViewId, OwnerActorNr, IsCeneView and IsMine when clicked.
[More...](#)

Inherits MonoBehaviour.

Public Member Functions

void **SetFocus** (PhotonView pv)

void **RemoveFocus** (PhotonView pv)

Public Attributes

Text text

Static Public Attributes

static PointedAtGameObjectInfo	Instance
---------------------------------------	-----------------

Detailed Description

Display ViewId, OwnerActorNr, IsCeneView and IsMine when clicked.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy		Class Members
Photon	Pun	UtilityScripts	PunPlayerScores			

[Public Attributes](#) | [List of all members](#)

PunPlayerScores Class Reference

Scoring system for PhotonPlayer [More...](#)

Inherits MonoBehaviour.

Public Attributes

```
const string PlayerScoreProp = "score"
```

Detailed Description

Scoring system for PhotonPlayer

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	PunTeams	

PunTeams Class Reference

[Public Types](#) | [Public Member Functions](#) |
[Public Attributes](#) | [Static Public Attributes](#) |
[List of all members](#)

Implements teams in a room/game with help of player properties.
Access them by `Player.GetTeam` extension. [More...](#)

Inherits **MonoBehaviourPunCallbacks**.

Public Types

enum **Team** : byte

Enum defining the teams available. First team should be neutral (it's the default value any field of this enum gets).

[More...](#)

Public Member Functions

void **Start** ()

override void **OnDisable** ()

override void **OnJoinedRoom** ()
Needed to update the team lists when joining a room.
[More...](#)

override void **OnLeftRoom** ()
Called when the local user/client left a room, so the game's logic can clean up it's internal state. [More...](#)

override void **OnPlayerPropertiesUpdate** (**Player** targetPlayer, Hashtable changedProps)
Refreshes the team lists. It could be a non-team related property change, too. [More...](#)

override void **OnPlayerLeftRoom** (**Player** otherPlayer)
Called when a remote player left the room or became inactive. Check otherPlayer.IsInactive. [More...](#)

override void **OnPlayerEnteredRoom** (**Player** newPlayer)
Called when a remote player entered the room. This Player is already added to the playerlist. [More...](#)

void **UpdateTeams** ()

‣ Public Member Functions inherited from **MonoBehaviourPunCallbacks**

virtual void **OnEnable** ()

virtual void **OnConnected** ()
Called to signal that the raw connection got established but before the client can call operation on

the server. [More...](#)

virtual void **OnMasterClientSwitched** (**Player** newMasterClient)
Called after switching to a new MasterClient when the current one leaves. [More...](#)

virtual void **OnCreateRoomFailed** (short returnCode, string message)
Called when the server couldn't create a room (OpCreateRoom failed). [More...](#)

virtual void **OnJoinRoomFailed** (short returnCode, string message)
Called when a previous OpJoinRoom call failed on the server. [More...](#)

virtual void **OnCreatedRoom** ()
Called when this client created a room and entered it. **OnJoinedRoom()** will be called as well. [More...](#)

virtual void **OnJoinedLobby** ()
Called on entering a lobby on the Master Server. The actual room-list updates will call OnRoomListUpdate. [More...](#)

virtual void **OnLeftLobby** ()
Called after leaving a lobby. [More...](#)

virtual void **OnDisconnected** (**DisconnectCause** cause)
Called after disconnecting from the **Photon** server. It could be a failure or intentional [More...](#)

virtual void **OnRegionListReceived** (**RegionHandler** regionHandler)
Called when the Name Server provided a list of regions for your title. [More...](#)

virtual void **OnRoomListUpdate** (List< **RoomInfo** > roomList)
Called for any update of the room-listing while in a lobby (InLobby) on the Master Server. [More...](#)

virtual void **OnJoinRandomFailed** (short returnCode, string message)
Called when a previous OpJoinRandom call failed on the server. [More...](#)

virtual void **OnConnectedToMaster** ()
Called when the client is connected to the Master Server and ready for matchmaking and other tasks. [More...](#)

virtual void **OnRoomPropertiesUpdate** (Hashtable propertiesThatChanged)
Called when a room's custom properties changed. The propertiesThatChanged contains all that was set via **Room.SetCustomProperties**. [More...](#)

virtual void **OnFriendListUpdate** (List< **FriendInfo** > friendList)
Called when the server sent the response to a FindFriends request. [More...](#)

virtual void **OnCustomAuthenticationResponse** (Dictionary< string, object > data)
Called when your Custom Authentication service responds with additional data. [More...](#)

virtual void **OnCustomAuthenticationFailed** (string debugMessage)
Called when the custom authentication failed. Followed by disconnect! [More...](#)

virtual void **OnWebRpcResponse** (OperationResponse response)

virtual void **OnLobbyStatisticsUpdate** (List< **TypedLobbyInfo** >

lobbyStatistics)

Called when the Master Server sent an update for the
Lobby Statistics, updating
PhotonNetwork.LobbyStatistics. [More...](#)

Public Attributes

const string **TeamPlayerProp** = "team"

Defines the player custom property name to use for team affinity of "this" player. [More...](#)

Static Public Attributes

static Dictionary< Team , List< Player > >	PlayersPerTeam The main list of teams with their player-lists. Automatically kept up to date. More...
--	---

Additional Inherited Members

‣ Properties inherited from **MonoBehaviourPun**

PhotonView **photonView** [get]

A cached reference to a **PhotonView** on this
GameObject. [More...](#)

Detailed Description

Implements teams in a room/game with help of player properties.
Access them by `Player.GetTeam` extension.

Teams are defined by enum `Team`. Change this to get more / different teams. There are no rules when / if you can join a team. You could add this in `JoinTeam` or something.

Member Enumeration Documentation

enum Team : byte

strong

Enum defining the teams available. First team should be neutral (it's the default value any field of this enum gets).

Member Function Documentation

override void OnJoinedRoom ()

virtual

Needed to update the team lists when joining a room.

Called by PUN. See enum **MonoBehaviourPunCallbacks** for an explanation.

Reimplemented from **MonoBehaviourPunCallbacks**.

override void OnLeftRoom ()

virtual

Called when the local user/client left a room, so the game's logic can clean up it's internal state.

When leaving a room, the LoadBalancingClient will disconnect the Game Server and connect to the Master Server. This wraps up multiple internal actions.

Wait for the callback OnConnectedToMaster, before you use lobbies and join or create rooms.

Reimplemented from **MonoBehaviourPunCallbacks**.

override void OnPlayerEnteredRoom (Player **newPlayer)**

virtual

Called when a remote player entered the room. This Player is already added to the playerlist.

If your game starts with a certain number of players, this callback can be useful to check the Room.playerCount and find out if you can start.

Reimplemented from **MonoBehaviourPunCallbacks**.

override void OnPlayerLeftRoom (Player **otherPlayer)**

virtual

Called when a remote player left the room or became inactive. Check `otherPlayer.IsInactive`.

If another player leaves the room or if the server detects a lost connection, this callback will be used to notify your game logic.

Depending on the room's setup, players may become inactive, which means they may return and retake their spot in the room. In such cases, the `Player` stays in the **Room.Players** dictionary.

If the player is not just inactive, it gets removed from the **Room.Players** dictionary, before the callback is called.

Reimplemented from **MonoBehaviourPunCallbacks**.

**override void
OnPlayerPropertiesUpdate (Player **targetPlayer**,
Hashtable **changedProps**
)**

virtual

Refreshes the team lists. It could be a non-team related property change, too.

Called by PUN. See enum **MonoBehaviourPunCallbacks** for an explanation.

Reimplemented from **MonoBehaviourPunCallbacks**.

Member Data Documentation

Dictionary<Team, List<Player> > PlayersPerTeam

static

The main list of teams with their player-lists. Automatically kept up to date.

Note that this is static. Can be accessed by PunTeam.PlayersPerTeam. You should not modify this.

const string TeamPlayerProp = "team"

Defines the player custom property name to use for team affinity of "this" player.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	PunTurnManager	

[Public Member Functions](#) | [Public Attributes](#) |

[Properties](#) | [List of all members](#)

PunTurnManager Class Reference

Pun turnBased Game manager. Provides an Interface (**IPunTurnManagerCallbacks**) for the typical turn flow and logic, between players Provides Extensions for Player, Room and RoomInfo to feature dedicated api for TurnBased Needs [More...](#)

Inherits **MonoBehaviourPunCallbacks**, and **IOnEventCallback**.

Public Member Functions

void **BeginTurn** ()
Tells the TurnManager to begins a new turn. [More...](#)

void **SendMove** (object move, bool finished)
Call to send an action. Optionally finish the turn, too. The move object can be anything. Try to optimize though and only send the strict minimum set of information to define the turn move. [More...](#)

bool **GetPlayerFinishedTurn** (Player player)
Gets if the player finished the current turn. [More...](#)

void **OnEvent** (EventData photonEvent)
Called by PhotonNetwork.OnEventCall registration
[More...](#)

override void **OnRoomPropertiesUpdate** (Hashtable propertiesThatChanged)
Called by **PhotonNetwork** [More...](#)

► Public Member Functions inherited from MonoBehaviourPunCallbacks

virtual void **OnEnable** ()

virtual void **OnDisable** ()

virtual void **OnConnected** ()
Called to signal that the raw connection got established but before the client can call operation on the server. [More...](#)

virtual void **OnLeftRoom** ()
Called when the local user/client left a room, so the game's logic can clean up it's internal state. [More...](#)

virtual void **OnMasterClientSwitched** (**Player** newMasterClient)
Called after switching to a new MasterClient when the current one leaves. [More...](#)

virtual void **OnCreateRoomFailed** (short returnCode, string message)
Called when the server couldn't create a room (OpCreateRoom failed). [More...](#)

virtual void **OnJoinRoomFailed** (short returnCode, string message)
Called when a previous OpJoinRoom call failed on the server. [More...](#)

virtual void **OnCreatedRoom** ()
Called when this client created a room and entered it. **OnJoinedRoom()** will be called as well. [More...](#)

virtual void **OnJoinedLobby** ()
Called on entering a lobby on the Master Server. The actual room-list updates will call OnRoomListUpdate. [More...](#)

virtual void **OnLeftLobby** ()
Called after leaving a lobby. [More...](#)

virtual void **OnDisconnected** (**DisconnectCause** cause)
Called after disconnecting from the **Photon** server. It could be a failure or intentional [More...](#)

virtual void **OnRegionListReceived** (**RegionHandler** regionHandler)
Called when the Name Server provided a list of regions for your title. [More...](#)

virtual void **OnRoomListUpdate** (List< **RoomInfo** > roomList)

Called for any update of the room-listing while in a lobby (InLobby) on the Master Server. [More...](#)

virtual void **OnJoinedRoom ()**
Called when the LoadBalancingClient entered a room, no matter if this client created it or simply joined. [More...](#)

virtual void **OnPlayerEnteredRoom (Player newPlayer)**
Called when a remote player entered the room. This Player is already added to the playerlist. [More...](#)

virtual void **OnPlayerLeftRoom (Player otherPlayer)**
Called when a remote player left the room or became inactive. Check otherPlayer.IsInactive. [More...](#)

virtual void **OnJoinRandomFailed** (short returnCode, string message)
Called when a previous OpJoinRandom call failed on the server. [More...](#)

virtual void **OnConnectedToMaster ()**
Called when the client is connected to the Master Server and ready for matchmaking and other tasks. [More...](#)

virtual void **OnPlayerPropertiesUpdate (Player target, Hashtable changedProps)**
Called when custom player-properties are changed. Player and the changed properties are passed as object[]. [More...](#)

virtual void **OnFriendListUpdate** (List< **FriendInfo** > friendList)
Called when the server sent the response to a FindFriends request. [More...](#)

virtual void **OnCustomAuthenticationResponse** (Dictionary<

string, object > data)
Called when your Custom Authentication service responds with additional data. [More...](#)

virtual void **OnCustomAuthenticationFailed** (string debugMessage)
Called when the custom authentication failed. Followed by disconnect! [More...](#)

virtual void **OnWebRpcResponse** (OperationResponse response)

virtual void **OnLobbyStatisticsUpdate** (List< TypedLobbyInfo > lobbyStatistics)
Called when the Master Server sent an update for the Lobby Statistics, updating PhotonNetwork.LobbyStatistics. [More...](#)

Public Attributes

float **TurnDuration** = 20f
The duration of the turn in seconds.
[More...](#)

IPunTurnManagerCallbacks **TurnManagerListener**
The turn manager listener. Set this to your own script instance to catch Callbacks [More...](#)

const byte **TurnManagerEventOffset** = 0
The turn manager event offset event message byte. Used internally for defining data in Room Custom Properties [More...](#)

const byte **EvMove** = 1 + **TurnManagerEventOffset**
The Move event message byte. Used internally for saving data in Room Custom Properties [More...](#)

const byte **EvFinalMove** = 2 + **TurnManagerEventOffset**
The Final Move event message byte. Used internally for saving data in Room Custom Properties [More...](#)

Properties

int **Turn** [get]

Wraps accessing the "turn" custom properties of a room. [More...](#)

float **ElapsedTimeInTurn** [get]

Gets the elapsed time in the current turn in seconds [More...](#)

float **RemainingSecondsInTurn** [get]

Gets the remaining seconds for the current turn. Ranges from 0 to TurnDuration [More...](#)

bool **IsCompletedByAll** [get]

Gets a value indicating whether the turn is completed by all. [More...](#)

bool **IsFinishedByMe** [get]

Gets a value indicating whether the current turn is finished by me. [More...](#)

bool **IsOver** [get]

Gets a value indicating whether the current turn is over. That is the ElapsedTimeInTurn is greater or equal to the TurnDuration [More...](#)

► Properties inherited from **MonoBehaviourPun**

PhotonView **photonView** [get]

A cached reference to a **PhotonView** on this GameObject. [More...](#)

Detailed Description

Pun turnBased Game manager. Provides an Interface (**IPunTurnManagerCallbacks**) for the typical turn flow and logic, between players Provides Extensions for Player, Room and RoomInfo to feature dedicated api for TurnBased Needs

Member Function Documentation

void BeginTurn ()

Tells the TurnManager to begins a new turn.

bool GetPlayerFinishedTurn (Player **player)**

Gets if the player finished the current turn.

Returns

true, if player finished the current turn, false otherwise.

Parameters

player The Player to check for

void OnEvent (EventArgs **photonEvent)**

Called by PhotonNetwork.OnEventCall registration

Parameters

photonEvent Photon event.

Implements **IOneventCallback**.

override void OnRoomPropertiesUpdate (Hashtable **propertiesThatChanged)**

Called by PhotonNetwork

Parameters

propertiesThatChanged Properties that changed.

Reimplemented from **MonoBehaviourPunCallbacks**.

```
void SendMove ( object move,  
                bool  finished  
                )
```

Call to send an action. Optionally finish the turn, too. The move object can be anything. Try to optimize though and only send the strict minimum set of information to define the turn move.

Parameters

move

finished

Member Data Documentation

const byte EvFinalMove = 2 + TurnManagerEventOffset

The Final Move event message byte. Used internally for saving data in Room Custom Properties

const byte EvMove = 1 + TurnManagerEventOffset

The Move event message byte. Used internally for saving data in Room Custom Properties

float TurnDuration = 20f

The duration of the turn in seconds.

const byte TurnManagerEventOffset = 0

The turn manager event offset event message byte. Used internally for defining data in Room Custom Properties

IPunTurnManagerCallbacks TurnManagerListener

The turn manager listener. Set this to your own script instance to catch Callbacks

Property Documentation

float ElapsedTimeInTurn

get

Gets the elapsed time in the current turn in seconds

The elapsed time in the turn.

bool IsCompletedByAll

get

Gets a value indicating whether the turn is completed by all.

true if this turn is completed by all; otherwise, false.

bool IsFinishedByMe

get

Gets a value indicating whether the current turn is finished by me.

true if the current turn is finished by me; otherwise, false.

bool IsOver

get

Gets a value indicating whether the current turn is over. That is the ElapsedTimeInTurn is greater or equal to the TurnDuration

true if the current turn is over; otherwise, false.

float RemainingSecondsInTurn

get

Gets the remaining seconds for the current turn. Ranges from 0 to

TurnDuration

The remaining seconds fo the current turn

int Turn

get

Wraps accessing the "turn" custom properties of a room.

The turn index



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List	Class Index	Class Hierarchy		Class Members		
Photon	Pun	UtilityScripts	ScoreExtensions			

ScoreExtensions Class Reference

[Static Public Member Functions](#) |
[List of all members](#)

Static Public Member Functions

static void **SetScore** (this **Player** player, int newScore)

static void **AddScore** (this **Player** player, int scoreToAddToCurrent)

static int **GetScore** (this **Player** player)



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	SmoothSyncMovement			

[Public Member Functions](#) | [Public Attributes](#) |

[List of all members](#)

SmoothSyncMovement Class Reference

Smoothed out movement for network gameobjects [More...](#)

Inherits **MonoBehaviourPun**, and **IPunObservable**.

Public Member Functions

void **Awake** ()

void **OnPhotonSerializeView** (PhotonStream stream, PhotonMessageInfo info)

Called by PUN several times per second, so that your script can write and read synchronization data for the **PhotonView**.
[More...](#)

void **Update** ()

Public Attributes

float **SmoothingDelay** = 5

Additional Inherited Members

‣ Properties inherited from **MonoBehaviourPun**

PhotonView **photonView** [get]

A cached reference to a **PhotonView** on this
GameObject. [More...](#)

Detailed Description

Smoothed out movement for network gameobjects

Member Function Documentation

```
void OnPhotonSerializeView ( PhotonStream      stream,  
                             PhotonMessageInfo info  
                             )
```

Called by PUN several times per second, so that your script can write and read synchronization data for the **PhotonView**.

This method will be called in scripts that are assigned as Observed component of a **PhotonView**.

PhotonNetwork.SerializationRate affects how often this method is called.

PhotonNetwork.SendRate affects how often packages are sent by this client.

Implementing this method, you can customize which data a **PhotonView** regularly synchronizes. Your code defines what is being sent (content) and how your data is used by receiving clients.

Unlike other callbacks, *OnPhotonSerializeView* only gets called when it is assigned to a **PhotonView** as PhotonView.observed script.

To make use of this method, the **PhotonStream** is essential. It will be in "writing" mode" on the client that controls a PhotonView (PhotonStream.IsWriting == true) and in "reading mode" on the remote clients that just receive that the controlling client sends.

If you skip writing any value into the stream, PUN will skip the update. Used carefully, this can conserve bandwidth and messages (which have a limit per room/second).

Note that OnPhotonSerializeView is not called on remote clients when the sender does not send any update. This can't be used as "x-times per second Update()".

Implements **IPunObservable**.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	StatesGui	

[Public Attributes](#) | [List of all members](#)

StatesGui Class Reference

Output detailed information about **Pun** Current states, using the old Unity UI framework. [More...](#)

Inherits MonoBehaviour.

Public Attributes

Rect **GuiOffset** = new Rect(250, 0, 300, 300)

bool **DontDestroy** = true

bool **ServerTimestamp**

bool **DetailedConnection**

bool **Server**

bool **AppVersion**

bool **UserId**

bool **Room**

bool **RoomProps**

bool **LocalPlayer**

bool **PlayerProps**

bool **Others**

bool **Buttons**

bool **ExpectedUsers**

Detailed Description

Output detailed information about **Pun** Current states, using the old Unity UI framework.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	TabViewManager			

TabViewManager Class Reference

[Classes](#) | [Public Member Functions](#) |
[Public Attributes](#) | [Protected Attributes](#) |
[List of all members](#)

Tab view manager. Handles **Tab** views activation and deactivation, and provides a Unity Event Callback when a tab was selected. [More...](#)

Inherits MonoBehaviour.

Classes

class **Tab**

class **TabChangeEvent**
Tab change event. [More...](#)

Public Member Functions

void **SelectTab** (string id)
Selects a given tab. [More...](#)

Public Attributes

ToggleGroup	ToggleGroup The toggle group component target. More...
-------------	--

Tab[]	Tabs all the tabs for this group More...
-------	--

TabChangeEvent	OnTabChanged The on tab changed Event. More...
----------------	--

Protected Attributes

Tab	CurrentTab
-----	------------

Detailed Description

Tab view manager. Handles **Tab** views activation and deactivation, and provides a Unity Event Callback when a tab was selected.

Member Function Documentation

void SelectTab (string **id)**

Selects a given tab.

Parameters

id Tab Id

Member Data Documentation

TabChangeEvent OnTabChanged

The on tab changed Event.

Tab [] Tabs

all the tabs for this group

ToggleGroup ToggleGroup

The toggle group component target.



Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	TabViewManager	Tab		

[Public Attributes](#) | [List of all members](#)

TabViewManager.Tab Class Reference

Public Attributes

string **ID** = ""

Toggle **Toggle**

RectTransform **View**



Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	TabViewManager	TabChangeEvent		

TabViewManager.TabChangeEvent Class Reference

Tab change event. [More...](#)

Inherits `UnityEvent< string >`.

Detailed Description

Tab change event.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	TeamExtensions	

TeamExtensions Class Reference

[Static Public Member Functions](#) |

[List of all members](#)

Extension used for **PunTeams** and Player class. Wraps access to the player's custom property. [More...](#)

Static Public Member Functions

static **PunTeams.Team** **GetTeam** (this **Player** player)
Extension for Player class to wrap up access to the player's custom property. [More...](#)

static void **SetTeam** (this **Player** player, **PunTeams.Team** team)
Switch that player's team to the one you assign. [More...](#)

Detailed Description

Extension used for **PunTeams** and Player class. Wraps access to the player's custom property.

Member Function Documentation

static PunTeams.Team GetTeam (this Player **player)**

static

Extension for Player class to wrap up access to the player's custom property.

Returns

PunTeam.Team.none if no team was found (yet).

static void SetTeam (this Player **player,
PunTeams.Team **team**
)**

static

Switch that player's team to the one you assign.

Internally checks if this player is in that team already or not. Only team switches are actually sent.

Parameters

player
team



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	TextButtonTransition			

[Public Member Functions](#) | [Public Attributes](#) |

[List of all members](#)

TextButtonTransition Class Reference

Use this on Button texts to have some color transition on the text as well without corrupting button's behaviour. [More...](#)

Inherits MonoBehaviour, IPointerEnterHandler, and IPointerExitHandler.

Public Member Functions

void **Awake** ()

void **OnEnable** ()

void **OnDisable** ()

void **OnPointerEnter** (PointerEventData eventData)

void **OnPointerExit** (PointerEventData eventData)

Public Attributes

Selectable **Selectable**

The selectable Component. [More...](#)

Color **NormalColor** = Color.white

The color of the normal of the transition state. [More...](#)

Color **HoverColor** = Color.black

The color of the hover of the transition state. [More...](#)

Detailed Description

Use this on Button texts to have some color transition on the text as well without corrupting button's behaviour.

Member Data Documentation

Color HoverColor = Color.black

The color of the hover of the transition state.

Color NormalColor = Color.white

The color of the normal of the transition state.

Selectable Selectable

The selectable Component.



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	TextToggleIsOnTransition			

[Public Member Functions](#) | [Public Attributes](#) |

[List of all members](#)

TextToggleIsOnTransition Class Reference

Use this on toggles texts to have some color transition on the text depending on the isOn State. [More...](#)

Inherits MonoBehaviour, IPointerEnterHandler, and IPointerExitHandler.

Public Member Functions

void **OnEnable** ()

void **OnDisable** ()

void **OnValueChanged** (bool isOn)

void **OnPointerEnter** (PointerEventData eventData)

void **OnPointerExit** (PointerEventData eventData)

Public Attributes

Toggle **toggle**
The toggle Component. [More...](#)

Color **NormalOnColor** = Color.white
The color of the normal on transition state. [More...](#)

Color **NormalOffColor** = Color.black
The color of the normal off transition state. [More...](#)

Color **HoverOnColor** = Color.black
The color of the hover on transition state. [More...](#)

Color **HoverOffColor** = Color.black
The color of the hover off transition state. [More...](#)

Detailed Description

Use this on toggles texts to have some color transition on the text depending on the isOn State.

Member Data Documentation

Color HoverOffColor = Color.black

The color of the hover off transition state.

Color HoverOnColor = Color.black

The color of the hover on transition state.

Color NormalOffColor = Color.black

The color of the normal off transition state.

Color NormalOnColor = Color.white

The color of the normal on transition state.

Toggle toggle

The toggle Component.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	TurnExtensions	

TurnExtensions Class Reference

[Static Public Member Functions](#) |
[Static Public Attributes](#) | [List of all members](#)

Static Public Member Functions

static void **SetTurn** (this **Room** room, int turn, bool
setStartTime=false)
Sets the turn. [More...](#)

static int **GetTurn** (this **RoomInfo** room)
Gets the current turn from a RoomInfo [More...](#)

static int **GetTurnStart** (this **RoomInfo** room)
Returns the start time when the turn began. This can be
used to calculate how long it's going on. [More...](#)

static int **GetFinishedTurn** (this **Player** player)
gets the player's finished turn (from the ROOM
properties) [More...](#)

static void **SetFinishedTurn** (this **Player** player, int turn)
Sets the player's finished turn (in the ROOM properties)
[More...](#)

Static Public Attributes

static readonly string **TurnPropKey** = "Turn"
currently ongoing turn number [More...](#)

static readonly string **TurnStartPropKey** = "TStart"
start (server) time for currently ongoing turn
(used to calculate end) [More...](#)

static readonly string **FinishedTurnPropKey** = "FToA"
Finished Turn of Actor (followed by number)
[More...](#)

Member Function Documentation

static int GetFinishedTurn (this Player **player)**

static

gets the player's finished turn (from the ROOM properties)

Returns

The finished turn index

Parameters

player Player reference

static int GetTurn (this RoomInfo **room)**

static

Gets the current turn from a RoomInfo

Returns

The turn index

Parameters

room RoomInfo reference

static int GetTurnStart (this RoomInfo **room)**

static

Returns the start time when the turn began. This can be used to calculate how long it's going on.

Returns

The turn start.

Parameters

room Room.

```
static void SetFinishedTurn ( this Player player,  
                             int      turn  
                             )
```

static

Sets the player's finished turn (in the ROOM properties)

Parameters

player Player Reference

turn Turn Index

```
static void SetTurn ( this Room room,  
                     int      turn,  
                     bool     setStartTime = false  
                     )
```

static

Sets the turn.

Parameters

room Room reference

turn Turn index

setStartTime If set to true set start time.

Member Data Documentation

readonly string FinishedTurnPropKey = "FToA"

static

Finished Turn of Actor (followed by number)

readonly string TurnPropKey = "Turn"

static

currently ongoing turn number

readonly string TurnStartPropKey = "TStart"

static

start (server) time for currently ongoing turn (used to calculate end)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	DefaultPool		

[Public Member Functions](#) | [Public Attributes](#) | [List of all members](#)

DefaultPool Class Reference

The default implementation of a PrefabPool for PUN, which actually Instantiates and Destroys GameObjects but pools a resource. [More...](#)

Inherits **IPunPrefabPool**.

Public Member Functions

GameObject **Instantiate** (string prefabId, Vector3 position, Quaternion rotation)
Returns an inactive instance of a networked GameObject, to be used by PUN. [More...](#)

void **Destroy** (GameObject gameObject)
Simply destroys a GameObject. [More...](#)

Public Attributes

readonly Dictionary< string, GameObject >	ResourceCache = new Dictionary<string, GameObject>() Contains a GameObject per prefabId, to speed up instantiation. More...
---	---

Detailed Description

The default implementation of a PrefabPool for PUN, which actually Instantiates and Destroys GameObjects but pools a resource.

This pool is not actually storing GameObjects for later reuse. Instead, it's destroying used GameObjects. However, prefabs will be loaded from a Resources folder and cached, which speeds up Instantiation a bit.

The ResourceCache is public, so it can be filled without relying on the Resources folders.

Member Function Documentation

void Destroy (GameObject **gameObject)**

Simply destroys a GameObject.

Parameters

gameObject The GameObject to get rid of.

Implements **IPunPrefabPool**.

GameObject Instantiate (string **prefabId,
Vector3 **position**,
Quaternion **rotation**
)**

Returns an inactive instance of a networked GameObject, to be used by PUN.

Parameters

prefabId String identifier for the networked object.

position Location of the new object.

rotation Rotation of the new object.

Returns

Implements **IPunPrefabPool**.

Member Data Documentation

readonly Dictionary<string, GameObject> ResourceCache = new Dictionary<string, GameObject>()

Contains a GameObject per prefabId, to speed up instantiation.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	InstantiateParameters		

[Public Member Functions](#) | [Public Attributes](#) | [List of all members](#)

InstantiateParameters Struct Reference

Public Member Functions

InstantiateParameters (string prefabName, Vector3 position, Quaternion rotation, byte @group, object[] data, byte objLevelPrefix, int[] viewIDs, **Player** creator, int timestamp)

Public Attributes

int[]	viewIDs
-------	----------------

byte	objLevelPrefix
------	-----------------------

object[]	data
----------	-------------

byte	group
------	--------------

Quaternion	rotation
------------	-----------------

Vector3	position
---------	-----------------

string	prefabName
--------	-------------------

Player	creator
---------------	----------------

int	timestamp
-----	------------------



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	IPunPrefabPool		

[Public Member Functions](#) | [List of all members](#)

IPunPrefabPool Interface Reference

Defines an interface for object pooling, used in
PhotonNetwork.Instantiate and **PhotonNetwork.Destroy**. [More...](#)

Inherited by **DefaultPool**.

Public Member Functions

GameObject **Instantiate** (string prefabId, Vector3 position, Quaternion rotation)
Called to get an instance of a prefab. Must return valid, disabled GameObject with **PhotonView**. [More...](#)

void **Destroy** (GameObject gameObject)
Called to destroy (or just return) the instance of a prefab. It's disabled and the pool may reset and cache it for later use in Instantiate. [More...](#)

Detailed Description

Defines an interface for object pooling, used in `PhotonNetwork.Instantiate` and **`PhotonNetwork.Destroy`**.

To apply your custom **`IPunPrefabPool`**, set **`PhotonNetwork.PrefabPool`**.

The pool has to return a valid, disabled `GameObject` when PUN calls `Instantiate`. Also, the position and rotation must be applied.

Note that `Awake` and `Start` are only called once by Unity, so scripts on re-used `GameObjects` should make use of `OnEnable` and or `OnDisable`. When `OnEnable` gets called, the **`PhotonView`** is already updated to the new values.

To be able to enable a `GameObject`, `Instantiate` must return an inactive object.

Before PUN "destroys" `GameObjects`, it will disable them.

If a component implements **`IPunInstantiateMagicCallback`**, PUN will call `OnPhotonInstantiate` when the networked object gets instantiated. If no components implement this on a prefab, PUN will optimize the instantiation and no longer looks up **`IPunInstantiateMagicCallback`** via `GetComponent`s.

Member Function Documentation

void Destroy (GameObject **gameObject)**

Called to destroy (or just return) the instance of a prefab. It's disabled and the pool may reset and cache it for later use in Instantiate.

A pool needs some way to find out which type of GameObject got returned via **Destroy()**. It could be a tag, name, a component or anything similar.

Parameters

gameObject The instance to destroy.

Implemented in **DefaultPool**.

GameObject Instantiate (string **prefabId, Vector3 **position**, Quaternion **rotation**)**

Called to get an instance of a prefab. Must return valid, disabled GameObject with **PhotonView**.

Parameters

prefabId The id of this prefab.

position The position for the instance.

rotation The rotation for the instance.

Returns

A disabled instance to use by PUN or null if the prefabId is unknown.

Implemented in **DefaultPool**.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	MonoBehaviourPun		

[Properties](#) | [List of all members](#)

MonoBehaviourPun Class Reference

This class adds the property `photonView`, while logging a warning when your game still uses the `networkView`. [More...](#)

Inherits `MonoBehaviour`.

Inherited by **`MonoBehaviourPunCallbacks`**, **`MoveByKeys`**, **`OnClickDestroy`**, **`OnClickRpc`**, and **`SmoothSyncMovement`**.

Properties

PhotonView **photonView** [get]

A cached reference to a **PhotonView** on this
GameObject. [More...](#)

Detailed Description

This class adds the property `photonView`, while logging a warning when your game still uses the `networkView`.

Property Documentation

PhotonView photonView

get

A cached reference to a **PhotonView** on this GameObject.

If you intend to work with a **PhotonView** in a script, it's usually easier to write `this.photonView`.

If you intend to remove the **PhotonView** component from the GameObject but keep this Photon.MonoBehaviour, avoid this reference or modify this code to use `PhotonView.Get(obj)` instead.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonAnimatorView		

[Classes](#) | [Public Types](#) |

PhotonAnimatorView Class Reference

[Public Member Functions](#) | [List of all members](#)

This class helps you to synchronize Mecanim animations Simply add the component to your GameObject and make sure that the **PhotonAnimatorView** is added to the list of observed components [More...](#)

Inherits MonoBehaviour, and **IPunObservable**.

Classes

class	SynchronizedLayer
-------	--------------------------

class	SynchronizedParameter
-------	------------------------------

Public Types

enum	ParameterType
------	----------------------

enum	SynchronizeType
------	------------------------

Public Member Functions

void **CacheDiscreteTriggers ()**
Caches the discrete triggers values f
keeping track of raised triggers, and
be reseted after the sync routine got
performed [More...](#)

bool **DoesLayerSynchronizeTypeExist (**
layerIndex)
Check if a specific layer is configured
be synchronize [More...](#)

bool **DoesParameterSynchronizeTypeE**
(string name)
Check if the specified parameter is
configured to be synchronized [More.](#)

List< **SynchronizedLayer** > **GetSynchronizedLayers ()**
Get a list of all synchronized layers
[More...](#)

List< **SynchronizedParameter** > **GetSynchronizedParameters ()**
Get a list of all synchronized
parameters [More...](#)

SynchronizeType **GetLayerSynchronizeType** (int
layerIndex)
Gets the type how the layer is
synchronized [More...](#)

SynchronizeType **GetParameterSynchronizeType** (st
name)
Gets the type how the parameter is
synchronized [More...](#)

void **SetLayerSynchronized** (int layerInc
SynchronizeType synchronizeType)
Sets the how a layer should be
synchronized [More...](#)

void **SetParameterSynchronized** (string
name, ParameterType type,
SynchronizeType synchronizeType)
Sets the how a parameter should be
synchronized [More...](#)

void **OnPhotonSerializeView**
(**PhotonStream** stream,
PhotonMessageInfo info)
Called by PUN several times per
second, so that your script can write
and read synchronization data for the
PhotonView. [More...](#)

Detailed Description

This class helps you to synchronize Mecanim animations. Simply add the component to your GameObject and make sure that the **PhotonAnimatorView** is added to the list of observed components.

When Using Trigger Parameters, make sure the component that sets the trigger is higher in the stack of Components on the GameObject than '**PhotonAnimatorView**'. Triggers are raised true during one frame only.

Member Function Documentation

void CacheDiscreteTriggers ()

Caches the discrete triggers values for keeping track of raised triggers, and will be reseted after the sync routine got performed

bool DoesLayerSynchronizeTypeExist (int **layerIndex)**

Check if a specific layer is configured to be synchronize

Parameters

layerIndex Index of the layer.

Returns

True if the layer is synchronized

bool DoesParameterSynchronizeTypeExist (string **name)**

Check if the specified parameter is configured to be synchronized

Parameters

name The name of the parameter.

Returns

True if the parameter is synchronized

SynchronizeType GetLayerSynchronizeType (int **layerIndex)**

Gets the type how the layer is synchronized

Parameters

layerIndex Index of the layer.

Returns

Disabled/Discrete/Continuous

SynchronizeType GetParameterSynchronizeType (string **name)**

Gets the type how the parameter is synchronized

Parameters

name The name of the parameter.

Returns

Disabled/Discrete/Continuous

List<SynchronizedLayer> GetSynchronizedLayers ()

Get a list of all synchronized layers

Returns

List of **SynchronizedLayer** objects

List<SynchronizedParameter> GetSynchronizedParameters ()

Get a list of all synchronized parameters

Returns

List of **SynchronizedParameter** objects

```
void OnPhotonSerializeView ( PhotonStream stream,  
                             PhotonMessageInfo info  
                             )
```

Called by PUN several times per second, so that your script can write and read synchronization data for the **PhotonView**.

This method will be called in scripts that are assigned as Observed component of a **PhotonView**.

PhotonNetwork.SerializationRate affects how often this method is called.

PhotonNetwork.SendRate affects how often packages are sent by this client.

Implementing this method, you can customize which data a **PhotonView** regularly synchronizes. Your code defines what is being sent (content) and how your data is used by receiving clients.

Unlike other callbacks, *OnPhotonSerializeView* only gets called when it is assigned to a **PhotonView** as PhotonView.observed script.

To make use of this method, the **PhotonStream** is essential. It will be in "writing" mode" on the client that controls a PhotonView (PhotonStream.IsWriting == true) and in "reading mode" on the remote clients that just receive that the controlling client sends.

If you skip writing any value into the stream, PUN will skip the update. Used carefully, this can conserve bandwidth and messages (which have a limit per room/second).

Note that OnPhotonSerializeView is not called on remote clients when the sender does not send any update. This can't be used as "x-times per second Update()".

Implements **IPunObservable**.

```
void  
SetLayerSynchronized ( int layerIndex,  
                      SynchronizeType synchronizeType  
                      )
```

Sets the how a layer should be synchronized

Parameters

layerIndex Index of the layer.
synchronizeType Disabled/Discrete/Continuous

```
void  
SetParameterSynchronized ( string      name,  
                           ParameterType type,  
                           SynchronizeType synchronizeType  
                           )
```

Sets the how a parameter should be synchronized

Parameters

name The name of the parameter.
type The type of the parameter.
synchronizeType Disabled/Discrete/Continuous



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonAnimatorView	SynchronizedLayer	

[Public Attributes](#) | [List of all members](#)

PhotonAnimatorView.SynchronizedLayer Class Reference

Public Attributes

SynchronizeType	SynchronizeType
-----------------	------------------------

int	LayerIndex
-----	-------------------

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonAnimatorView	SynchronizedParameter	

[Public Attributes](#) | [List of all members](#)

PhotonAnimatorView.SynchronizedParameter Class Reference

Public Attributes

ParameterType	Type
---------------	-------------

SynchronizeType	SynchronizeType
-----------------	------------------------

string	Name
--------	-------------

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonRigidbody2DView		

[Public Member Functions](#) | [Public Attributes](#) |

[List of all members](#)

PhotonRigidbody2DView Class Reference

Inherits MonoBehaviour, and **IPunObservable**.

Public Member Functions

void **Awake** ()

void **FixedUpdate** ()

void **OnPhotonSerializeView** (**PhotonStream** stream,
PhotonMessageInfo info)

Called by PUN several times per second, so that your script can write and read synchronization data for the **PhotonView**.
[More...](#)

Public Attributes

bool **m_SynchronizeVelocity** = true

bool **m_SynchronizeAngularVelocity** = false

bool **m_TeleportEnabled** = false

float **m_TeleportIfDistanceGreaterThan** = 3.0f

Member Function Documentation

```
void OnPhotonSerializeView ( PhotonStream      stream,  
                             PhotonMessageInfo info  
                             )
```

Called by PUN several times per second, so that your script can write and read synchronization data for the **PhotonView**.

This method will be called in scripts that are assigned as Observed component of a **PhotonView**.

PhotonNetwork.SerializationRate affects how often this method is called.

PhotonNetwork.SendRate affects how often packages are sent by this client.

Implementing this method, you can customize which data a **PhotonView** regularly synchronizes. Your code defines what is being sent (content) and how your data is used by receiving clients.

Unlike other callbacks, *OnPhotonSerializeView only gets called when it is assigned to a **PhotonView** as PhotonView.observed script.*

To make use of this method, the **PhotonStream** is essential. It will be in "writing" mode" on the client that controls a PhotonView (PhotonStream.IsWriting == true) and in "reading mode" on the remote clients that just receive that the controlling client sends.

If you skip writing any value into the stream, PUN will skip the update. Used carefully, this can conserve bandwidth and messages (which have a limit per room/second).

Note that OnPhotonSerializeView is not called on remote clients when the sender does not send any update. This can't be used as "x-times per second Update()".

Implements **IPunObservable**.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonRigidbodyView		

[Public Member Functions](#) | [Public Attributes](#) | [List of all members](#)

PhotonRigidbodyView Class Reference

Inherits MonoBehaviour, and **IPunObservable**.

Public Member Functions

void **Awake** ()

void **FixedUpdate** ()

void **OnPhotonSerializeView** (PhotonStream stream,
PhotonMessageInfo info)

Called by PUN several times per second, so that your script can write and read synchronization data for the **PhotonView**.
[More...](#)

Public Attributes

bool **m_SynchronizeVelocity** = true

bool **m_SynchronizeAngularVelocity** = false

bool **m_TeleportEnabled** = false

float **m_TeleportIfDistanceGreaterThan** = 3.0f

Member Function Documentation

```
void OnPhotonSerializeView ( PhotonStream      stream,  
                             PhotonMessageInfo info  
                             )
```

Called by PUN several times per second, so that your script can write and read synchronization data for the **PhotonView**.

This method will be called in scripts that are assigned as Observed component of a **PhotonView**.

PhotonNetwork.SerializationRate affects how often this method is called.

PhotonNetwork.SendRate affects how often packages are sent by this client.

Implementing this method, you can customize which data a **PhotonView** regularly synchronizes. Your code defines what is being sent (content) and how your data is used by receiving clients.

Unlike other callbacks, *OnPhotonSerializeView only gets called when it is assigned to a **PhotonView** as PhotonView.observed script.*

To make use of this method, the **PhotonStream** is essential. It will be in "writing" mode" on the client that controls a PhotonView (PhotonStream.IsWriting == true) and in "reading mode" on the remote clients that just receive that the controlling client sends.

If you skip writing any value into the stream, PUN will skip the update. Used carefully, this can conserve bandwidth and messages (which have a limit per room/second).

Note that OnPhotonSerializeView is not called on remote clients when the sender does not send any update. This can't be used as "x-times per second Update()".

Implements **IPunObservable**.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonStreamQueue		

[Public Member Functions](#) | [List of all members](#)

PhotonStreamQueue Class Reference

The **PhotonStreamQueue** helps you poll object states at higher frequencies than what **PhotonNetwork.SendRate** dictates and then sends all those states at once when **Serialize()** is called. On the receiving end you can call **Deserialize()** and then the stream will roll out the received object states in the same order and timeStep they were recorded in. [More...](#)

Public Member Functions

PhotonStreamQueue (int sampleRate)

Initializes a new instance of the **PhotonStreamQueue** class.
[More...](#)

void **Reset** ()

Resets the **PhotonStreamQueue**. You need to do this whenever the amount of objects you are observing changes
[More...](#)

void **SendNext** (object obj)

Adds the next object to the queue. This works just like **PhotonStream.SendNext** [More...](#)

bool **HasQueuedObjects** ()

Determines whether the queue has stored any objects [More...](#)

object **ReceiveNext** ()

Receives the next object from the queue. This works just like **PhotonStream.ReceiveNext** [More...](#)

void **Serialize** (PhotonStream stream)

Serializes the specified stream. Call this in your OnPhotonSerializeView method to send the whole recorded stream. [More...](#)

void **Deserialize** (PhotonStream stream)

Deserializes the specified stream. Call this in your OnPhotonSerializeView method to receive the whole recorded stream. [More...](#)

Detailed Description

The **PhotonStreamQueue** helps you poll object states at higher frequencies than what **PhotonNetwork.SendRate** dictates and then sends all those states at once when **Serialize()** is called. On the receiving end you can call **Deserialize()** and then the stream will roll out the received object states in the same order and timeStep they were recorded in.

Constructor & Destructor Documentation

PhotonStreamQueue (int *sampleRate*)

Initializes a new instance of the **PhotonStreamQueue** class.

Parameters

sampleRate How many times per second should the object states be sampled

Member Function Documentation

void Deserialize (PhotonStream **stream)**

Deserializes the specified stream. Call this in your OnPhotonSerializeView method to receive the whole recorded stream.

Parameters

stream The **PhotonStream** you receive as a parameter in OnPhotonSerializeView

bool HasQueuedObjects ()

Determines whether the queue has stored any objects

object ReceiveNext ()

Receives the next object from the queue. This works just like **PhotonStream.ReceiveNext**

Returns

void Reset ()

Resets the **PhotonStreamQueue**. You need to do this whenever the amount of objects you are observing changes

void SendNext (object **obj)**

Adds the next object to the queue. This works just like **PhotonStream.SendNext**

Parameters

obj The object you want to add to the queue

void Serialize (PhotonStream **stream)**

Serializes the specified stream. Call this in your OnPhotonSerializeView method to send the whole recorded stream.

Parameters

stream The **PhotonStream** you receive as a parameter in OnPhotonSerializeView



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonTransformView		

[Public Member Functions](#) | [Public Attributes](#) | [List of all members](#)

PhotonTransformView Class Reference

Inherits MonoBehaviour, and **IPunObservable**.

Public Member Functions

void **Awake** ()

void **Update** ()

void **OnPhotonSerializeView** (PhotonStream stream,
PhotonMessageInfo info)

Called by PUN several times per second, so that your script can write and read synchronization data for the **PhotonView**.
[More...](#)

Public Attributes

bool **m_SynchronizePosition** = true

bool **m_SynchronizeRotation** = true

bool **m_SynchronizeScale** = false

Member Function Documentation

```
void OnPhotonSerializeView ( PhotonStream      stream,  
                             PhotonMessageInfo info  
                             )
```

Called by PUN several times per second, so that your script can write and read synchronization data for the **PhotonView**.

This method will be called in scripts that are assigned as Observed component of a **PhotonView**.

PhotonNetwork.SerializationRate affects how often this method is called.

PhotonNetwork.SendRate affects how often packages are sent by this client.

Implementing this method, you can customize which data a **PhotonView** regularly synchronizes. Your code defines what is being sent (content) and how your data is used by receiving clients.

Unlike other callbacks, *OnPhotonSerializeView only gets called when it is assigned to a **PhotonView** as PhotonView.observed script.*

To make use of this method, the **PhotonStream** is essential. It will be in "writing" mode" on the client that controls a PhotonView (PhotonStream.IsWriting == true) and in "reading mode" on the remote clients that just receive that the controlling client sends.

If you skip writing any value into the stream, PUN will skip the update. Used carefully, this can conserve bandwidth and messages (which have a limit per room/second).

Note that OnPhotonSerializeView is not called on remote clients when the sender does not send any update. This can't be used as "x-times per second Update()".

Implements **IPunObservable**.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonTransformViewClassic		

[Public Member Functions](#) | [Public Attributes](#) |

[List of all members](#)

PhotonTransformViewClassic Class Reference

This class helps you to synchronize position, rotation and scale of a `GameObject`. It also gives you many different options to make the synchronized values appear smooth, even when the data is only send a couple of times per second. Simply add the component to your `GameObject` and make sure that the **PhotonTransformViewClassic** is added to the list of observed components [More...](#)

Inherits `MonoBehaviour`, and **`IPunObservable`**.

Public Member Functions

void SetSynchronizedValues (Vector3 speed, float turnSpeed)
These values are synchronized to the remote objects if the interpolation mode or the extrapolation mode SynchronizeValues is used. Your movement script should pass on the current speed (in units/second) and turning speed (in angles/second) so the remote object can use them to predict the objects movement. [More...](#)

void OnPhotonSerializeView (PhotonStream stream, PhotonMessageInfo info)
Called by PUN several times per second, so that your script can write and read synchronization data for the **PhotonView**. [More...](#)

Public Attributes

PhotonTransformViewPositionModel	m_PositionModel = new PhotonTransformViewPositi
---	---

PhotonTransformViewRotationModel	m_RotationModel = new PhotonTransformViewRotati
---	---

PhotonTransformViewScaleModel	m_ScaleModel = new PhotonTransformViewScale
--------------------------------------	---

Detailed Description

This class helps you to synchronize position, rotation and scale of a `GameObject`. It also gives you many different options to make the synchronized values appear smooth, even when the data is only send a couple of times per second. Simply add the component to your `GameObject` and make sure that the **PhotonTransformViewClassic** is added to the list of observed components

Member Function Documentation

```
void OnPhotonSerializeView ( PhotonStream      stream,  
                             PhotonMessageInfo info  
                             )
```

Called by PUN several times per second, so that your script can write and read synchronization data for the **PhotonView**.

This method will be called in scripts that are assigned as Observed component of a **PhotonView**.

PhotonNetwork.SerializationRate affects how often this method is called.

PhotonNetwork.SendRate affects how often packages are sent by this client.

Implementing this method, you can customize which data a **PhotonView** regularly synchronizes. Your code defines what is being sent (content) and how your data is used by receiving clients.

Unlike other callbacks, *OnPhotonSerializeView only gets called when it is assigned to a **PhotonView** as PhotonView.observed script.*

To make use of this method, the **PhotonStream** is essential. It will be in "writing" mode" on the client that controls a PhotonView (PhotonStream.IsWriting == true) and in "reading mode" on the remote clients that just receive that the controlling client sends.

If you skip writing any value into the stream, PUN will skip the update. Used carefully, this can conserve bandwidth and messages (which have a limit per room/second).

Note that OnPhotonSerializeView is not called on remote clients when the sender does not send any update. This can't be used as "x-times per second Update()".

Implements **IPunObservable**.

```
void SetSynchronizedValues ( Vector3 speed,  
                             float   turnSpeed  
                             )
```

These values are synchronized to the remote objects if the interpolation mode or the extrapolation mode `SynchronizeValues` is used. Your movement script should pass on the current speed (in units/second) and turning speed (in angles/second) so the remote object can use them to predict the objects movement.

Parameters

speed The current movement vector of the object in units/second.

turnSpeed The current turn speed of the object in angles/second.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonTransformViewPositionControl		

[Public Member Functions](#) | [List of all members](#)

PhotonTransformViewPositionControl Class Reference

Public Member Functions

PhotonTransformViewPositionControl (**PhotonTransformViewPositionModel** model)

void **SetSynchronizedValues** (Vector3 speed, float turnSpeed)
These values are synchronized to the remote objects if the interpolation mode or the extrapolation mode SynchronizeValues is used. Your movement script should pass on the current speed (in units/second) and turning speed (in angles/second) so the remote object can use them to predict the objects movement. [More...](#)

Vector3 **UpdatePosition** (Vector3 currentPosition)
Calculates the new position based on the values setup in the inspector [More...](#)

Vector3 **GetNetworkPosition** ()
Gets the last position that was received through the network [More...](#)

Vector3 **GetExtrapolatedPositionOffset** ()
Calculates an estimated position based on the last synchronized position, the time when the last position was received and the movement speed of the object [More...](#)

void **OnPhotonSerializeView** (Vector3 currentPosition, **PhotonStream** stream, **PhotonMessageInfo** info)

Member Function Documentation

Vector3 GetExtrapolatedPositionOffset ()

Calculates an estimated position based on the last synchronized position, the time when the last position was received and the movement speed of the object

Returns

Estimated position of the remote object

Vector3 GetNetworkPosition ()

Gets the last position that was received through the network

Returns

void SetSynchronizedValues (Vector3 **speed, float **turnSpeed**)**

These values are synchronized to the remote objects if the interpolation mode or the extrapolation mode SynchronizeValues is used. Your movement script should pass on the current speed (in units/second) and turning speed (in angles/second) so the remote object can use them to predict the objects movement.

Parameters

speed The current movement vector of the object in units/second.

turnSpeed The current turn speed of the object in angles/second.

Vector3 UpdatePosition (Vector3 **currentPosition**)

Calculates the new position based on the values setup in the inspector

Parameters

currentPosition The current position.

Returns

The new position.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonTransformViewPositionModel		

[Public Types](#) | [Public Attributes](#) |
[List of all members](#)

PhotonTransformViewPositionModel Class Reference

Public Types

enum **InterpolateOptions**

enum **ExtrapolateOptions**

Public Attributes

	bool	SynchronizeEnabled
	bool	TeleportEnabled = true
	float	TeleportIfDistanceGreaterThan = 3f
InterpolateOptions		InterpolateOption = InterpolateOptions.EstimatedSpeed
	float	InterpolateMoveTowardsSpeed = 1f
	float	InterpolateLerpSpeed = 1f
ExtrapolateOptions		ExtrapolateOption = ExtrapolateOptions.Disabled
	float	ExtrapolateSpeed = 1f
	bool	ExtrapolateIncludingRoundTripTime = true
	int	ExtrapolateNumberOfStoredPositions = 1



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonTransformViewRotationControl		

[Public Member Functions](#) | [List of all members](#)

PhotonTransformViewRotationControl Class Reference

Public Member Functions

PhotonTransformViewRotationControl
(**PhotonTransformViewRotationModel** model)

Quaternion **GetNetworkRotation** ()
Gets the last rotation that was received through the network [More...](#)

Quaternion **GetRotation** (Quaternion currentRotation)

void **OnPhotonSerializeView** (Quaternion currentRotation, **PhotonStream** stream, **PhotonMessageInfo** info)

Member Function Documentation

Quaternion GetNetworkRotation ()

Gets the last rotation that was received through the network

Returns



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonTransformViewRotationModel		

[Public Types](#) | [Public Attributes](#) |
[List of all members](#)

PhotonTransformViewRotationModel Class Reference

Public Types

enum **InterpolateOptions**

Public Attributes

bool **SynchronizeEnabled**

InterpolateOptions **InterpolateOption** =
InterpolateOptions.RotateTowards

float **InterpolateRotateTowardsSpeed** = 180

float **InterpolateLerpSpeed** = 5



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonTransformViewScaleControl		

[Public Member Functions](#) | [List of all members](#)

PhotonTransformViewScaleControl Class Reference

Public Member Functions

PhotonTransformViewScaleControl
(**PhotonTransformViewScaleModel** model)

Vector3 **GetNetworkScale** ()
Gets the last scale that was received through the network
[More...](#)

Vector3 **GetScale** (Vector3 currentScale)

void **OnPhotonSerializeView** (Vector3 currentScale,
PhotonStream stream, **PhotonMessageInfo** info)

Member Function Documentation

Vector3 GetNetworkScale ()

Gets the last scale that was received through the network

Returns



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonTransformViewScaleModel		

[Public Types](#) | [Public Attributes](#) |

[List of all members](#)

PhotonTransformViewScaleModel Class Reference

Public Types

enum **InterpolateOptions**

Public Attributes

bool **SynchronizeEnabled**

InterpolateOptions **InterpolateOption** = InterpolateOptions.Disabled

float **InterpolateMoveTowardsSpeed** = 1f

float **InterpolateLerpSpeed**



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PunExtensions		

PunExtensions Class Reference

[Static Public Member Functions](#) |
[Static Public Attributes](#) | [List of all members](#)

Small number of extension methods that make it easier for PUN to work cross-Unity-versions. [More...](#)

Static Public Member Functions

static ParameterInfo[] **GetCachedParemeters** (this MethodInfo mo)

static PhotonView[] **GetPhotonViewsInChildren** (this
UnityEngine.GameObject go)

static PhotonView **GetPhotonView** (this
UnityEngine.GameObject go)

static bool **AlmostEquals** (this Vector3 target, Vector3
second, float sqrMagnitudePrecision)
compares the squared magnitude of target -
second to given float value [More...](#)

static bool **AlmostEquals** (this Vector2 target, Vector2
second, float sqrMagnitudePrecision)
compares the squared magnitude of target -
second to given float value [More...](#)

static bool **AlmostEquals** (this Quaternion target,
Quaternion second, float maxAngle)
compares the angle between target and
second to given float value [More...](#)

static bool **AlmostEquals** (this float target, float second,
float floatDiff)
compares two floats and returns true of their
difference is less than floatDiff [More...](#)

Static Public Attributes

```
static Dictionary< MethodInfo, ParameterInfo[]> ParametersOfMethod  
= new  
Dictionary<MethodInfo  
ParameterInfo[]>()
```

Detailed Description

Small number of extension methods that make it easier for PUN to work cross-Unity-versions.

Member Function Documentation

```
static bool  
AlmostEquals      ( this Vector3 target,  
                    Vector3      second,  
                    float        sqrMagnitudePrecision  
                    )
```

static

compares the squared magnitude of target - second to given float value

```
static bool  
AlmostEquals      ( this Vector2 target,  
                    Vector2      second,  
                    float        sqrMagnitudePrecision  
                    )
```

static

compares the squared magnitude of target - second to given float value

```
static bool AlmostEquals ( this Quaternion target,  
                           Quaternion      second,  
                           float          maxAngle  
                           )
```

static

compares the angle between target and second to given float value

```
static bool AlmostEquals ( this float target,  
                           float      second,  
                           float      floatDiff
```

)

static

compares two floats and returns true if their difference is less than floatDiff



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PunRPC		

PunRPC Class Reference

Replacement for RPC attribute with different name. Used to flag methods as remote-callable. [More...](#)

Inherits Attribute.

Detailed Description

Replacement for RPC attribute with different name. Used to flag methods as remote-callable.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	SceneManagerHelper		

[Properties](#) | [List of all members](#)

SceneManagerHelper Class Reference

Properties

static string **ActiveSceneName** [get]

static int **ActiveSceneBuildIndex** [get]

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	ServerSettings		

ServerSettings Class Reference

[Public Member Functions](#) |
[Static Public Member Functions](#) |
[Public Attributes](#) | [Properties](#) |
[List of all members](#)

Collection of connection-relevant settings, used internally by **PhotonNetwork.ConnectUsingSettings**. [More...](#)

Inherits ScriptableObject.

Public Member Functions

void **UseCloud** (string cloudAppid, string code="")
Sets appid and region code in the AppSettings. Used in Editor. [More...](#)

override string **ToString** ()
String summary of the AppSettings. [More...](#)

Static Public Member Functions

static bool **IsAppId** (string val)

Checks if a string is a Guid by attempting to create one.

[More...](#)

static void **ResetBestRegionCodeInPreferences** ()

Sets the "best region summary" in the preferences to null.

On next start, the client will ping all available. [More...](#)

Public Attributes

AppSettings **AppSettings**

bool **StartInOfflineMode**

PunLogLevel **PunLogging** = **PunLogLevel.ErrorsOnly**

bool **EnableSupportLogger**

bool **RunInBackground** = true

List< string > **RpcList** = new List<string>()

Properties

static string **BestRegionSummaryInPreferences** [get]
Gets the "best region summary" from the preferences.
[More...](#)

Detailed Description

Collection of connection-relevant settings, used internally by **PhotonNetwork.ConnectUsingSettings**.

Includes the AppSettings class from the **Realtime** APIs plus some other, PUN-relevant, settings.

Member Function Documentation

static bool IsAppId (string **val)**

static

Checks if a string is a Guid by attempting to create one.

Parameters

val The potential guid to check.

Returns

True if new Guid(val) did not fail.

static void ResetBestRegionCodeInPreferences ()

static

Sets the "best region summary" in the preferences to null. On next start, the client will ping all available.

override string ToString ()

String summary of the AppSettings.

void UseCloud (string **cloudAppid,
 string **code** = ""
)**

Sets appid and region code in the AppSettings. Used in Editor.

Property Documentation

string BestRegionSummaryInPreferences

static get

Gets the "best region summary" from the preferences.

The best region code in preferences.



Photon Unity Networking 2 2.12

[Main Page](#)[Related Pages](#)[Modules](#)[Classes](#)[Package Functions](#)[Photon](#)[Realtime](#)[Classes](#) | [Typedefs](#) | [Enumerations](#)

Photon.Realtime Namespace Reference

Classes

class **ActorProperties**

Class for constants. These (byte) values define "well known" properties for an Actor / **Player**. [More...](#)

class **AppSettings**

Settings for **Photon** application(s) and the server to connect to. [More...](#)

class **AuthenticationValues**

Container for user authentication in **Photon**. Set AuthValues before you connect - all else is handled. [More...](#)

class **ConnectionCallbacksContainer**

Container type for callbacks defined by **IConnectionCallbacks**. See LoadBalancingCallbackTargets. [More...](#)

class **ConnectionHandler**

class **EncryptionDataParameters**

class **EnterRoomParams**

class **ErrorCode**

ErrorCode defines the default codes associated with **Photon** client/server communication. [More...](#)

class **EventCode**

Class for constants. These values are for events defined by **Photon** Loadbalancing. [More...](#)

class **EventExt**

class **Extensions**

This static class defines some useful extension methods for several existing classes (e.g. Vector3, float and others). [More...](#)

class **FriendInfo**

Used to store info about a friend's online state and in which room he/she is. [More...](#)

class **GamePropertyKey**

Class for constants. These (byte) values are for "well known" room/game properties used in **Photon** Loadbalancing. [More...](#)

interface **ICollectionCallbacks**

Collection of "organizational" callbacks for the **Realtime** Api to cover: Connection and Regions. [More...](#)

interface **InRoomCallbacks**

Collection of "in room" callbacks for the **Realtime** Api to cover: Players entering or leaving, property updates and Master Client switching. [More...](#)

interface **ILobbyCallbacks**

Collection of "organizational" callbacks for the **Realtime** Api to cover the Lobby. [More...](#)

interface **IMatchmakingCallbacks**

Collection of "organizational" callbacks for the **Realtime** Api to cover Matchmaking. [More...](#)

class **InRoomCallbacksContainer**

Container type for callbacks defined by **InRoomCallbacks**. See InRoomCallbackTargets.

interface **IONEventCallback**

Event callback for the **Realtime** Api. Covers events from

the server and those sent by clients via `OpRaiseEvent`.
[More...](#)

interface **IWebRpcCallback**
Interface for "WebRpc" callbacks for the **Realtime** Api.
Currently includes only responses for Web RPCs. [More...](#)

class **LoadBalancingClient**
This class implements the **Photon** LoadBalancing workflow by using a **LoadBalancingPeer**. It keeps a state and will automatically execute transitions between the Master and Game Servers. [More...](#)

class **LoadBalancingPeer**
A LoadbalancingPeer provides the operations and enum definitions needed to use the loadbalancing server application which is also used in **Photon** Cloud. [More...](#)

class **LobbyCallbacksContainer**
Container type for callbacks defined by **ILobbyCallbacks**.
See LobbyCallbackTargets.

class **MatchMakingCallbacksContainer**
Container type for callbacks defined by **IMatchmakingCallbacks**. See MatchMakingCallbackTargets. [More...](#)

class **OperationCode**
Class for constants. Contains operation codes. **Pun** uses these constants internally. [More...](#)

class **OpJoinRandomRoomParams**

class **ParameterCode**
Class for constants. Codes for parameters of Operations and Events. [More...](#)

class **PhotonPing**

class **PingMono**

Uses C# Socket class from System.Net.Sockets (as Unity usually does). [More...](#)

class **Player**

Summarizes a "player" within a room, identified (in that room) by ID (or "actorNumber"). [More...](#)

class **RaiseEventOptions**

Aggregates several less-often used options for operation RaiseEvent. See field descriptions for usage details. [More...](#)

class **Region**

class **RegionHandler**

Provides methods to work with **Photon**'s regions (**Photon** Cloud) and can be use to find the one with best ping. [More...](#)

class **RegionPinger**

class **Room**

This class represents a room a client joins/joined. [More...](#)

class **RoomInfo**

A simplified room with just the info required to list and join, used for the room listing in the lobby. The properties are not settable (IsOpen, MaxPlayers, etc). [More...](#)

class **RoomOptions**

Wraps up common room properties needed when you create rooms. Read the individual entries for more details. [More...](#)

class **SupportLogger**
Helper class to debug log basic information about **Photon** client and vital traffic statistics. [More...](#)

class **TypedLobby**
Refers to a specific lobby (and type) on the server. [More...](#)

class **TypedLobbyInfo**

class **WebFlags**
Optional flags to be used in **Photon** client SDKs with Op RaiseEvent and Op SetProperties. Introduced mainly for webhooks 1.2 to control behavior of forwarded HTTP requests. [More...](#)

class **WebRpcCallbacksContainer**
Container type for callbacks defined by **IWebRpcCallback**. See WebRpcCallbackTargets.

class **WebRpcResponse**
Reads an operation response of a WebRpc and provides convenient access to most common values. [More...](#)

Typedefs

```
using SupportClass = ExitGames.Client.Photon.SupportClass
```

```
using Stopwatch = System.Diagnostics.Stopwatch
```

Enumerations

enum **ClientState**

State values for a client, which handles switching **Photon** server types, some operations, etc. [More...](#)

enum **DisconnectCause**

Enumeration of causes for Disconnects (used in LoadBalancingClient.DisconnectedCause). [More...](#)

enum **ServerConnection**

Available server (types) for internally used field: server. [More...](#)

enum **EncryptionMode**

Defines how the communication gets encrypted. [More...](#)

enum **JoinMode** : byte

Defines possible values for OpJoinRoom and OpJoinOrCreate. It tells the server if the room can be only be joined normally, created implicitly or found on a web-service for Turnbased games. [More...](#)

enum **MatchmakingMode** : byte

Options for matchmaking rules for OpJoinRandom. [More...](#)

enum **ReceiverGroup** : byte

Lite - OpRaiseEvent lets you chose which actors in the room should receive events. By default, events are sent to "Others" but you can overrule this. [More...](#)

enum **EventCaching** : byte

Lite - OpRaiseEvent allows you to cache events and automatically send them to joining players in a room. Events are cached per event code and player: Event 100 (example!) can be stored once per player. Cached events can be

modified, replaced and removed. [More...](#)

enum **PropertyTypeFlag** : byte
Flags for "types of properties", being used as filter in
OpGetProperties. [More...](#)

enum **LobbyType** : byte
Options of lobby types available. Lobby types might be
implemented in certain **Photon** versions and won't be
available on older servers. [More...](#)

enum **AuthModeOption**
Options for authentication modes. From "classic" auth on
each server to AuthOnce (on NameServer). [More...](#)

enum **CustomAuthenticationType** : byte
Options for optional "Custom Authentication" services used
with **Photon**. Used by OpAuthenticate after connecting to
Photon. [More...](#)

Enumeration Type Documentation

enum AuthModeOption

strong

Options for authentication modes. From "classic" auth on each server to AuthOnce (on NameServer).

enum CustomAuthenticationType : byte

strong

Options for optional "Custom Authentication" services used with **Photon**. Used by OpAuthenticate after connecting to **Photon**.

Enumerator	
Custom	Use a custom authentication service. Currently the only implemented option.
Steam	Authenticates users by their Steam Account. Set auth values accordingly!
Facebook	Authenticates users by their Facebook Account. Set auth values accordingly!
Oculus	Authenticates users by their Oculus Account and token.
PlayStation	Authenticates users by their PSN Account and token.
Xbox	Authenticates users by their Xbox Account and XSTS token.
Viveport	Authenticates users by their HTC Viveport Account and user token. Set AuthGetParameters to "userToken=[userToken]"
NintendoSwitch	Authenticates users by their NSA ID.
None	Disables custom authentication. Same as not providing any AuthenticationValues for connect

(more precisely for: OpAuthenticate).

enum DisconnectCause

strong

Enumeration of causes for Disconnects (used in **LoadBalancingClient.DisconnectedCause**).

Read the individual descriptions to find out what to do about this type of disconnect.

Enumerator	
None	No error was tracked.
ExceptionOnConnect	OnStatusChanged: The server is not available or the address is wrong. Make sure the port is provided and the server is up.
Exception	OnStatusChanged: Some internal exception caused the socket code to fail. This may happen if you attempt to connect locally but the server is not available. In doubt: Contact Exit Games.
ServerTimeout	OnStatusChanged: The server disconnected this client due to timing out (missing acknowledgement from the client).
DisconnectByServer	OnStatusChanged: The server disconnected this client. Most likely the server's send buffer is full (receiving too much from

	other clients).
ClientTimeout	OnStatusChanged: This client detected that the server's responses are not received in due time.
TimeoutDisconnect	OnStatusChanged: This client detected that the server's responses are not received in due time.
DisconnectByServerLogic	OnStatusChanged: The server disconnected this client from within the room's logic (the C# code).
DisconnectByServerReasonUnknown	OnStatusChanged: The server disconnected this client for unknown reasons.
InvalidAuthentication	OnOperationResponse: Authenticate in the Photon Cloud with invalid AppId. Update your subscription or contact Exit Games.
CustomAuthenticationFailed	OnOperationResponse: Authenticate in the Photon Cloud with invalid client values or custom authentication setup in Cloud Dashboard.
AuthenticationTicketExpired	The authentication ticket should provide access to any Photon Cloud server without doing another authentication-service call. However, the ticket expired.

MaxCcuReached	OnOperationResponse: Authenticate (temporarily) failed when using a Photon Cloud subscription without CCU Burst. Update your subscription.
DisconnectByServerUserLimit	OnStatusChanged: The current CCUs exceed the CCUs of your subscription (or license). Get a suitable subscription (some include overage).
InvalidRegion	OnOperationResponse: Authenticate when the app's Photon Cloud subscription is locked to some (other) region(s). Update your subscription or master server address.
OperationNotAllowedInCurrentState	OnOperationResponse: Operation that's (currently) not available for this client (not authorized usually). Only tracked for op Authenticate.
DisconnectByClientLogic	OnStatusChanged: The client disconnected from within the logic (the C# code).

enum EncryptionMode

strong

Defines how the communication gets encrypted.

Enumerator

PayloadEncryption

This is the default

	encryption mode: Messages get encrypted only on demand (when you send operations with the "encrypt" parameter set to true).
DatagramEncryption	With this encryption mode for UDP, the connection gets setup and all further datagrams get encrypted almost entirely. On-demand message encryption (like in PayloadEncryption) is unavailable.
DatagramEncryptionRandomSequence	With this encryption mode for UDP, the connection gets setup with random sequence numbers and all further datagrams get encrypted almost entirely. On-demand message encryption (like in PayloadEncryption) is unavailable.

enum EventCaching : byte

Lite - OpRaiseEvent allows you to cache events and automatically send them to joining players in a room. Events are cached per event code and player 100 (example!) can be stored once per player. Cached events can be replaced and removed.

Caching works only in combination with ReceiverGroup options. Others are not supported.

Enumerator

DoNotCache

Default value (not sent).

MergeCache	Will merge this event's keys those already cached.
ReplaceCache	Replaces the event cache for eventCode with this event's
RemoveCache	Removes this event (by eventCode) from the cache.
AddToRoomCache	Adds an event to the room's
AddToRoomCacheGlobal	Adds this event to the cache as actor 0 (becoming a "global owned" event in the cache).
RemoveFromRoomCache	Remove fitting event from the cache.
RemoveFromRoomCacheForActorsLeft	Removes events of players already left the room (cleaning)
SliceIncreaseIndex	Increase the index of the slice in the cache.
SliceSetIndex	Set the index of the sliced cache. You must set RaiseEventOptions.CacheSliceIndex for this.
SlicePurgeIndex	Purge cache slice with index. Exactly one slice is removed from the cache. You must set RaiseEventOptions.CacheSliceIndex for this.
SlicePurgeUpToIndex	Purge cache slices with index and anything lower than index. You must set RaiseEventOptions.CacheSliceIndex for this.

enum JoinMode : byte

strong

Defines possible values for OpJoinRoom and OpJoinOrCreate. It tells the server if the room can be only be joined normally, created

implicitly or found on a web-service for Turnbased games.

These values are not directly used by a game but implicitly set.

Enumerator	
Default	Regular join. The room must exist.
CreatelfNotExists	Join or create the room if it's not existing. Used for OpJoinOrCreate for example.
JoinOrRejoin	The room might be out of memory and should be loaded (if possible) from a Turnbased web-service.
RejoinOnly	Only re-join will be allowed. If the user is not yet in the room, this will fail.

enum LobbyType : byte

strong

Options of lobby types available. Lobby types might be implemented in certain **Photon** versions and won't be available on older servers.

Enumerator	
Default	This lobby is used unless another is defined by game or JoinRandom. Room-lists will be sent and JoinRandomRoom can filter by matching properties.
SqlLobby	This lobby type lists rooms like Default but JoinRandom has a parameter for SQL-like "where" clauses for filtering. This allows bigger, less, or and and combinations.
AsyncRandomLobby	This lobby does not send lists of games. It is only used for OpJoinRandomRoom. It keeps rooms available for a while when there are only inactive users left.

enum MatchmakingMode : byte

strong

Options for matchmaking rules for OpJoinRandom.

Enumerator	
FillRoom	Fills up rooms (oldest first) to get players together as fast as possible. Default. Makes most sense with MaxPlayers > 0 and games that can only start with more players.
SerialMatching	Distributes players across available rooms sequentially but takes filter into account. Without filter, rooms get players evenly distributed.
RandomMatching	Joins a (fully) random room. Expected properties must match but aside from this, any available room might be selected.

enum PropertyTypeFlag : byte

strong

Flags for "types of properties", being used as filter in OpGetProperties.

Enumerator	
None	(0x00) Flag type for no property type.
Game	(0x01) Flag type for game-attached properties.
Actor	(0x02) Flag type for actor related propeties.
GameAndActor	(0x01) Flag type for game AND actor properties. Equal to 'Game'

enum ReceiverGroup : byte

strong

Lite - OpRaiseEvent lets you chose which actors in the room should receive events. By default, events are sent to "Others" but you can overrule this.

Enumerator	
Others	Default value (not sent). Anyone else gets my event.
All	Everyone in the current room (including this peer) will get this event.
MasterClient	<p>The server sends this event only to the actor with the lowest actorNumber.</p> <p>The "master client" does not have special rights but is the one who is in this room the longest time.</p>

enum ServerConnection

strong

Available server (types) for internally used field: server.

Photon uses 3 different roles of servers: Name Server, Master Server and Game Server.

Enumerator	
MasterServer	This server is where matchmaking gets done and where clients can get lists of rooms in lobbies.
GameServer	This server handles a number of rooms to execute and relay the messages between players (in a room).
NameServer	This server is used initially to get the address (IP) of a Master Server for a specific region. Not used for Photon OnPremise (self hosted).



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	ActorProperties		

[Public Attributes](#) | [List of all members](#)

ActorProperties Class Reference

Class for constants. These (byte) values define "well known" properties for an Actor / **Player**. [More...](#)

Public Attributes

const byte **PlayerName** = 255
(255) Name of a player/actor. [More...](#)

const byte **IsInactive** = 254
(254) Tells you if the player is currently in this game (getting events live). [More...](#)

const byte **UserId** = 253
(253) UserId of the player. Sent when room gets created with **RoomOptions.PublishUserId** = true. [More...](#)

Detailed Description

Class for constants. These (byte) values define "well known" properties for an Actor / **Player**.

Pun uses these constants internally. "Custom properties" have to use a string-type as key. They can be assigned at will.

Member Data Documentation

const byte IsInactive = 254

(254) Tells you if the player is currently in this game (getting events live).

A server-set value for async games, where players can leave the game and return later.

const byte PlayerName = 255

(255) Name of a player/actor.

const byte UserId = 253

(253) UserId of the player. Sent when room gets created with **RoomOptions.PublishUserId = true**.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	AppSettings		

AppSettings Class Reference

[Public Member Functions](#) | [Public Attributes](#) |
[Properties](#) | [List of all members](#)

Settings for **Photon** application(s) and the server to connect to. [More...](#)

Public Member Functions

string **ToStringFull** ()

ToString but with more details. [More...](#)

Public Attributes

string **AppIdRealtime**
AppId for **Realtime** or PUN. [More...](#)

string **AppIdChat**
AppId for the **Chat** Api. [More...](#)

string **AppIdVoice**
AppId for use in the Voice Api. [More...](#)

string **AppVersion**
The AppVersion can be used to identify builds and will split the AppId distinct "Virtual AppIds" (important for matchmaking). [More...](#)

bool **UseNameServer** = true
If false, the app will attempt to connect to a Master Server (which is obsolete but sometimes still necessary). [More...](#)

string **FixedRegion**
Can be set to any of the **Photon** Cloud's region names to directly connect to that region. [More...](#)

string **Server**
The address (hostname or IP) of the server to connect to. [More...](#)

int **Port**
If not null, this sets the port of the first **Photon** server to connect to (that will "forward" the client as needed). [More...](#)

ConnectionProtocol **Protocol** = ConnectionProtocol.Udp
The network level protocol to use. [More...](#)

bool **EnableLobbyStatistics**

If true, the client will request the list of currently available lobbies. [More...](#)

DebugLevel **NetworkLogging** = DebugLevel.ERROR

Log level for the network lib. [More...](#)

Properties

bool **IsMasterServerAddress** [get]

If true, the Server field contains a Master Server address (if any address at all). [More...](#)

bool **IsBestRegion** [get]

If true, the client should fetch the region list from the Name Server and find the one with best ping. [More...](#)

bool **IsDefaultNameServer** [get]

If true, the default nameserver address for the **Photon** Cloud should be used. [More...](#)

bool **IsDefaultPort** [get]

If true, the default ports for a protocol will be used. [More...](#)

Detailed Description

Settings for **Photon** application(s) and the server to connect to.

This is Serializable for Unity, so it can be included in ScriptableObject instances.

Member Function Documentation

string ToStringFull ()

ToString but with more details.

Member Data Documentation

string AppIdChat

AppId for the **Chat** Api.

string AppIdRealtime

AppId for **Realtime** or PUN.

string AppIdVoice

AppId for use in the Voice Api.

string AppVersion

The AppVersion can be used to identify builds and will split the AppId distinct "Virtual AppIds" (important for matchmaking).

bool EnableLobbyStatistics

If true, the client will request the list of currently available lobbies.

string FixedRegion

Can be set to any of the **Photon** Cloud's region names to directly connect to that region.

if this IsNullOrEmpty() AND UseNameServer == true, use BestRegion. else, use a server

DebugLevel NetworkLogging = DebugLevel.ERROR

Log level for the network lib.

int Port

If not null, this sets the port of the first **Photon** server to connect to (that will "forward" the client as needed).

ConnectionProtocol Protocol = ConnectionProtocol.Udp

The network level protocol to use.

string Server

The address (hostname or IP) of the server to connect to.

bool UseNameServer = true

If false, the app will attempt to connect to a Master Server (which is obsolete but sometimes still necessary).

if true, Server points to a NameServer (or is null, using the default), else it points to a MasterServer.

Property Documentation

bool IsBestRegion

get

If true, the client should fetch the region list from the Name Server and find the one with best ping.

See "Best Region" in the online docs.

bool IsDefaultNameServer

get

If true, the default nameserver address for the **Photon** Cloud should be used.

bool IsDefaultPort

get

If true, the default ports for a protocol will be used.

bool IsMasterServerAddress

get

If true, the Server field contains a Master Server address (if any address at all).



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	AuthenticationValues		

[Public Member Functions](#) | [Properties](#) |

[List of all members](#)

AuthenticationValues Class Reference

Container for user authentication in **Photon**. Set AuthValues before you connect - all else is handled. [More...](#)

Public Member Functions

AuthenticationValues ()

Creates empty auth values without any info. [More...](#)

AuthenticationValues (string userId)

Creates minimal info about the user. If this is authenticated or not, depends on the set AuthType. [More...](#)

virtual void **SetAuthPostData (string stringData)**

Sets the data to be passed-on to the auth service via POST. [More...](#)

virtual void **SetAuthPostData (byte[] byteData)**

Sets the data to be passed-on to the auth service via POST. [More...](#)

virtual void **SetAuthPostData (Dictionary< string, object > dictData)**

Sets data to be passed-on to the auth service as Json (Content-Type: "application/json") via Post. [More...](#)

virtual void **AddAuthParameter (string key, string value)**

Adds a key-value pair to the get-parameters used for Custom Auth (AuthGetParameters). [More...](#)

override string **ToString ()**

Properties

CustomAuthenticationType **AuthType** [get, set]
The type of custom authentication provider that should be used. Currently only "Custom" or "None" (turns this off). [More...](#)

string **AuthGetParameters** [get, set]
This string must contain any (http get) parameters expected by the used authentication service. By default, username and token. [More...](#)

object **AuthPostData** [get]
Data to be passed-on to the auth service via POST. Default: null (not sent). Either string or byte[] (see setters). [More...](#)

string **Token** [get, set]
After initial authentication, **Photon** provides a token for this client / user, which is subsequently used as (cached) validation. [More...](#)

string **UserId** [get, set]
The UserId should be a unique identifier per user. This is for finding friends, etc.. [More...](#)

Detailed Description

Container for user authentication in **Photon**. Set AuthValues before you connect - all else is handled.

On **Photon**, user authentication is optional but can be useful in many cases. If you want to FindFriends, a unique ID per user is very practical.

There are basically three options for user authentication: None at all, the client sets some UserId or you can use some account web-service to authenticate a user (and set the UserId server-side).

Custom Authentication lets you verify end-users by some kind of login or token. It sends those values to **Photon** which will verify them before granting access or disconnecting the client.

The AuthValues are sent in OpAuthenticate when you connect, so they must be set before you connect. Should you not set any AuthValues, PUN will create them and set the playerName as userId in them. If the AuthValues.userId is null or empty when it's sent to the server, then the **Photon** Server assigns a userId!

The **Photon** Cloud Dashboard will let you enable this feature and set important server values for it. <https://dashboard.photonengine.com>

Constructor & Destructor Documentation

AuthenticationValues ()

Creates empty auth values without any info.

AuthenticationValues (string **userId**)

Creates minimal info about the user. If this is authenticated or not, depends on the set AuthType.

Parameters

userId Some UserId to set in **Photon**.

Member Function Documentation

virtual void AddAuthParameter (string **key,
string **value**
)**

virtual

Adds a key-value pair to the get-parameters used for Custom Auth (AuthGetParameters).

This method does uri-encoding for you.

Parameters

key Key for the value to set.

value Some value relevant for Custom Authentication.

virtual void SetAuthPostData (string **stringData)**

virtual

Sets the data to be passed-on to the auth service via POST.

AuthPostData is just one value. Each SetAuthPostData replaces any previous value. It can be either a string, a byte[] or a dictionary. Each SetAuthPostData replaces any previous value.

Parameters

stringData String data to be used in the body of the POST request. Null or empty string will set AuthPostData to null.

virtual void SetAuthPostData (byte[] **byteData)**

virtual

Sets the data to be passed-on to the auth service via POST.

AuthPostData is just one value. Each SetAuthPostData replaces any previous value. It can be either a string, a byte[] or a dictionary. Each SetAuthPostData replaces any previous value.

Parameters

byteData Binary token / auth-data to pass on.

virtual void

SetAuthPostData (Dictionary< string, object > **dictData**) virtual

Sets data to be passed-on to the auth service as Json (Content-Type: "application/json") via Post.

AuthPostData is just one value. Each SetAuthPostData replaces any previous value. It can be either a string, a byte[] or a dictionary. Each SetAuthPostData replaces any previous value.

Parameters

dictData A authentication-data dictionary will be converted to Json and passed to the Auth webservice via HTTP Post.

Property Documentation

string AuthGetParameters

get set

This string must contain any (http get) parameters expected by the used authentication service. By default, username and token.

Maps to operation parameter 216. Standard http get parameters are used here and passed on to the service that's defined in the server (**Photon** Cloud Dashboard).

object AuthPostData

get

Data to be passed-on to the auth service via POST. Default: null (not sent). Either string or byte[] (see setters).

Maps to operation parameter 214.

CustomAuthenticationType AuthType

get set

The type of custom authentication provider that should be used. Currently only "Custom" or "None" (turns this off).

string Token

get set

After initial authentication, **Photon** provides a token for this client / user, which is subsequently used as (cached) validation.

string UserId

get set

The UserId should be a unique identifier per user. This is for finding friends, etc..

See remarks of AuthValues for info about how this is set and used.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	ConnectionCallbacksContainer		

[Public Member Functions](#) | [List of all members](#)

ConnectionCallbacksContainer Class Reference

Container type for callbacks defined by **IConnectionCallbacks**. See [LoadBalancingCallbackTargets](#). [More...](#)

Inherits `List< IConnectionCallbacks >`, and **IConnectionCallbacks**.

Public Member Functions

void **AddCallbackTarget** (IConnectionCallbacks target)

void **RemoveCallbackTarget** (IConnectionCallbacks target)

void **OnConnected** ()
Called to signal that the "low level connection" got established but before the client can call operation on the server. [More...](#)

void **OnConnectedToMaster** ()
Called when the client is connected to the Master Server and ready for matchmaking and other tasks. [More...](#)

void **OnRegionListReceived** (RegionHandler regionHandler)
Called when the Name Server provided a list of regions for your title. [More...](#)

void **OnDisconnected** (DisconnectCause cause)
Called after disconnecting from the **Photon** server. It could be a failure or an explicit disconnect call [More...](#)

void **OnCustomAuthenticationResponse** (Dictionary< string, object > data)
Called when your Custom Authentication service responds with additional data. [More...](#)

void **OnCustomAuthenticationFailed** (string debugMessage)
Called when the custom authentication failed. Followed by disconnect! [More...](#)

Detailed Description

Container type for callbacks defined by **ICallbacks**. See **LoadBalancingCallbackTargets**.

While the interfaces of callbacks wrap up the methods that will be called, the container classes implement a simple way to call a method on all registered objects.

Member Function Documentation

void OnConnected ()

Called to signal that the "low level connection" got established but before the client can call operation on the server.

After the (low level transport) connection is established, the client will automatically send the Authentication operation, which needs to get a response before the client can call other operations.

Your logic should wait for either: `OnRegionListReceived` or `OnConnectedToMaster`.

This callback is useful to detect if the server can be reached at all (technically). Most often, it's enough to implement **`OnDisconnected(DisconnectCause cause)`** and check for the cause.

This is not called for transitions from the masterserver to game servers.

Implements **`IConnectionCallbacks`**.

void OnConnectedToMaster ()

Called when the client is connected to the Master Server and ready for matchmaking and other tasks.

The list of available rooms won't become available unless you join a lobby via **`LoadBalancingClient.OpJoinLobby`**. You can join rooms and create them even without being in a lobby. The default lobby is used in that case.

Implements **`IConnectionCallbacks`**.

void OnCustomAuthenticationFailed (string debugMessage)

Called when the custom authentication failed. Followed by disconnect!

Custom Authentication can fail due to user-input, bad tokens/secrets. If authentication is successful, this method is not called. Implement OnJoinedLobby() or **OnConnectedToMaster()** (as usual).

During development of a game, it might also fail due to wrong configuration on the server side. In those cases, logging the debugMessage is very important.

Unless you setup a custom authentication service for your app (in the [Dashboard](#)), this won't be called!

Parameters

debugMessage Contains a debug message why authentication failed. This has to be fixed during development.

Implements **ICollectionCallbacks**.

void OnCustomAuthenticationResponse (Dictionary< string, object > c

Called when your Custom Authentication service responds with additional data.

Custom Authentication services can include some custom data in their response. When present, that data is made available in this callback as Dictionary. While the keys of your data have to be strings, the values can be either string or a number (in Json). You need to make extra sure, that the value type is the one you expect. Numbers become (currently) int64

Example: void OnCustomAuthenticationResponse(Dictionary<string,

```
object> data) { ... }
```

<https://doc.photonengine.com/en-us/realtime/current/reference/custom-authentication>

Implements **ICollectionCallbacks**.

void OnDisconnected (DisconnectCause **cause)**

Called after disconnecting from the **Photon** server. It could be a failure or an explicit disconnect call

The reason for this disconnect is provided as DisconnectCause.

Implements **ICollectionCallbacks**.

void OnRegionListReceived (RegionHandler **regionHandler)**

Called when the Name Server provided a list of regions for your title.

Check the **RegionHandler** class description, to make use of the provided values.

Parameters

regionHandler The currently used **RegionHandler**.

Implements **ICollectionCallbacks**.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	ConnectionHandler		

ConnectionHandler Class Reference

[Public Member Functions](#) | [Public Attributes](#) |
[Properties](#) | [List of all members](#)

Inherited by PhotonHandler.

Public Member Functions

void **StartFallbackSendAckThread** ()

void **StopFallbackSendAckThread** ()

bool **RealtimeFallbackThread** ()

A thread which runs independent from the Update() calls.
Keeps connections online while loading or in background. See
PhotonNetwork.BackgroundTimeout. [More...](#)

Public Attributes

int **KeepAliveInBackground** = 60000

Defines for how long the Fallback Thread should keep the connection, before it may time out as usual. [More...](#)

Properties

LoadBalancingClient Client [get, set]
Photon client to log information and statistics from. [More...](#)

int **CountSendAcksOnly** [get]
Counts how often the Fallback Thread called SendAcksOnly, which is purely of interest to monitor if the game logic called SendOutgoingCommands as intended.
[More...](#)

bool **FallbackThreadRunning** [get]

Member Function Documentation

bool RealtimeFallbackThread ()

A thread which runs independent from the Update() calls. Keeps connections online while loading or in background. See PhotonNetwork.BackgroundTimeout.

Member Data Documentation

int KeepAliveInBackground = 60000

Defines for how long the Fallback Thread should keep the connection, before it may time out as usual.

We want to the Client to keep it's connection when an app is in the background (and doesn't call Update / Service Clients should not keep their connection indefinitely in the background, so after some milliseconds, the Fallback Thread should stop keeping it up.

Property Documentation

LoadBalancingClient Client

get set

Photon client to log information and statistics from.

int CountSendAcksOnly

get

Counts how often the Fallback Thread called SendAcksOnly, which is purely of interest to monitor if the game logic called SendOutgoingCommands as intended.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	EncryptionDataParameters		

[Public Attributes](#) | [List of all members](#)

EncryptionDataParameters Class Reference

Public Attributes

const byte **Mode** = 0
Key for encryption mode [More...](#)

const byte **Secret1** = 1
Key for first secret [More...](#)

const byte **Secret2** = 2
Key for second secret [More...](#)

Member Data Documentation

const byte Mode = 0

Key for encryption mode

const byte Secret1 = 1

Key for first secret

const byte Secret2 = 2

Key for second secret



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	EnterRoomParams		

[Public Attributes](#) | [List of all members](#)

EnterRoomParams Class Reference

Public Attributes

string	RoomName
--------	-----------------

RoomOptions	RoomOptions
--------------------	--------------------

TypedLobby	Lobby
-------------------	--------------

Hashtable	PlayerProperties
-----------	-------------------------

bool	CreatelfNotExists
------	--------------------------

bool	RejoinOnly
------	-------------------

string[]	ExpectedUsers
----------	----------------------



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	ErrorCode		

[Public Attributes](#) | [List of all members](#)

ErrorCode Class Reference

ErrorCode defines the default codes associated with **Photon** client/server communication. [More...](#)

Public Attributes

const int **Ok** = 0

(0) is always "OK", anything else an error or specific situation. [More...](#)

const int **OperationNotAllowedInCurrentState** = -3

(-3) Operation can't be executed yet (e.g. OpJoin can't be called before being authenticated, RaiseEvent can't be used before getting into a room). [More...](#)

const int **InvalidOperationCode** = -2

(-2) The operation you called is not implemented on the server (application) you connect to. Make sure you run the fitting applications. [More...](#)

const int **InvalidOperation** = -2

(-2) The operation you called could not be executed on the server. [More...](#)

const int **InternalServerError** = -1

(-1) Something went wrong in the server. Try to reproduce and contact Exit Games. [More...](#)

const int **InvalidAuthentication** = 0x7FFF

(32767) Authentication failed. Possible cause: AppId is unknown to **Photon** (in cloud service). [More...](#)

const int **GameldAlreadyExists** = 0x7FFF - 1

(32766) Gameld (name) already in use (can't create another). Change name. [More...](#)

const int **GameFull** = 0x7FFF - 2

(32765) Game is full. This rarely happens when some player joined the room before your join completed. [More...](#)

const int **GameClosed** = 0x7FFF - 3
(32764) Game is closed and can't be joined. Join another game. [More...](#)

const int **AlreadyMatched** = 0x7FFF - 4

const int **ServerFull** = 0x7FFF - 5
(32762) Not in use currently. [More...](#)

const int **UserBlocked** = 0x7FFF - 6
(32761) Not in use currently. [More...](#)

const int **NoRandomMatchFound** = 0x7FFF - 7
(32760) Random matchmaking only succeeds if a room exists that's neither closed nor full. Repeat in a few seconds or create a new room. [More...](#)

const int **GameDoesNotExist** = 0x7FFF - 9
(32758) Join can fail if the room (name) is not existing (anymore). This can happen when players leave while you join. [More...](#)

const int **MaxCcuReached** = 0x7FFF - 10
(32757) Authorization on the **Photon** Cloud failed because the concurrent users (CCU) limit of the app's subscription is reached. [More...](#)

const int **InvalidRegion** = 0x7FFF - 11
(32756) Authorization on the **Photon** Cloud failed because the app's subscription does not allow to use a particular region's server. [More...](#)

const int **CustomAuthenticationFailed** = 0x7FFF - 12
(32755) Custom Authentication of the user failed due to setup reasons (see Cloud Dashboard) or the provided user data (like username or token). Check error message for details. [More...](#)

const int **AuthenticationTicketExpired** = 0x7FF1

(32753) The Authentication ticket expired. Usually, this is refreshed behind the scenes. Connect (and authorize) again. [More...](#)

const int **PluginReportedError** = 0x7FFF - 15

(32752) A server-side plugin (or webhook) failed to execute and reported an error. Check the `OperationResponse.DebugMessage`. [More...](#)

const int **PluginMismatch** = 0x7FFF - 16

(32751) CreateGame/JoinGame/Join operation fails if expected plugin does not correspond to loaded one. [More...](#)

const int **JoinFailedPeerAlreadyJoined** = 32750

(32750) for join requests. Indicates the current peer already called join and is joined to the room. [More...](#)

const int **JoinFailedFoundInactiveJoiner** = 32749

(32749) for join requests. Indicates the list of InactiveActors already contains an actor with the requested ActorNr or UserId. [More...](#)

const int **JoinFailedWithRejoinerNotFound** = 32748

(32748) for join requests. Indicates the list of Actors (active and inactive) did not contain an actor with the requested ActorNr or UserId. [More...](#)

const int **JoinFailedFoundExcludedUserId** = 32747

(32747) for join requests. Note: for future use - Indicates the requested UserId was found in the ExcludedList. [More...](#)

const int **JoinFailedFoundActiveJoiner** = 32746

(32746) for join requests. Indicates the list of ActiveActors

already contains an actor with the requested ActorNr or UserId. [More...](#)

const int **HttpLimitReached** = 32745
(32745) for SetPropertyes and Raiseevent (if flag HttpForward is true) requests. Indicates the maximum allowed http requests per minute was reached. [More...](#)

const int **ExternalHttpCallFailed** = 32744
(32744) for WebRpc requests. Indicates the the call to the external service failed. [More...](#)

const int **SlotError** = 32742
(32742) Server error during matchmaking with slot reservation. E.g. the reserved slots can not exceed MaxPlayers. [More...](#)

const int **InvalidEncryptionParameters** = 32741
(32741) Server will react with this error if invalid encryption parameters provided by token [More...](#)

Detailed Description

ErrorCode defines the default codes associated with **Photon** client/server communication.

Member Data Documentation

const int AuthenticationTicketExpired = 0x7FF1

(32753) The Authentication ticket expired. Usually, this is refreshed behind the scenes. Connect (and authorize) again.

const int CustomAuthenticationFailed = 0x7FFF - 12

(32755) Custom Authentication of the user failed due to setup reasons (see Cloud Dashboard) or the provided user data (like username or token). Check error message for details.

const int ExternalHttpCallFailed = 32744

(32744) for WebRpc requests. Indicates the the call to the external service failed.

const int GameClosed = 0x7FFF - 3

(32764) Game is closed and can't be joined. Join another game.

const int GameDoesNotExist = 0x7FFF - 9

(32758) Join can fail if the room (name) is not existing (anymore). This can happen when players leave while you join.

const int GameFull = 0x7FFF - 2

(32765) Game is full. This rarely happens when some player joined the room before your join completed.

const int GameldAlreadyExists = 0x7FFF - 1

(32766) Gameld (name) already in use (can't create another).
Change name.

const int HttpLimitReached = 32745

(32745) for SetPropertyes and Raisevent (if flag HttpForward is true) requests. Indicates the maximum allowed http requests per minute was reached.

const int InternalServerError = -1

(-1) Something went wrong in the server. Try to reproduce and contact Exit Games.

const int InvalidAuthentication = 0x7FFF

(32767) Authentication failed. Possible cause: AppId is unknown to **Photon** (in cloud service).

const int InvalidEncryptionParameters = 32741

(32741) Server will react with this error if invalid encryption parameters provided by token

const int InvalidOperation = -2

(-2) The operation you called could not be executed on the server.

Make sure you are connected to the server you expect.

This code is used in several cases: The arguments/parameters of the operation might be out of range, missing entirely or conflicting. The operation you called is not implemented on the server (application). Server-side plugins affect the available operations.

const int InvalidOperationCode = -2

(-2) The operation you called is not implemented on the server (application) you connect to. Make sure you run the fitting applications.

const int InvalidRegion = 0x7FFF - 11

(32756) Authorization on the **Photon** Cloud failed because the app's subscription does not allow to use a particular region's server.

Some subscription plans for the **Photon** Cloud are region-bound. Servers of other regions can't be used then. Check your master server address and compare it with your **Photon** Cloud Dashboard's info. <https://dashboard.photonengine.com>

OpAuthorize is part of connection workflow but only on the **Photon** Cloud, this error can happen. Self-hosted **Photon** servers with a CCU limited license won't let a client connect at all.

const int JoinFailedFoundActiveJoiner = 32746

(32746) for join requests. Indicates the list of ActiveActors already contains an actor with the requested ActorNr or UserId.

const int JoinFailedFoundExcludedUserId = 32747

(32747) for join requests. Note: for future use - Indicates the requested UserId was found in the ExcludedList.

const int JoinFailedFoundInactiveJoiner = 32749

(32749) for join requests. Indicates the list of InactiveActors already contains an actor with the requested ActorNr or UserId.

const int JoinFailedPeerAlreadyJoined = 32750

(32750) for join requests. Indicates the current peer already called join and is joined to the room.

const int JoinFailedWithRejoinerNotFound = 32748

(32748) for join requests. Indicates the list of Actors (active and inactive) did not contain an actor with the requested ActorNr or UserId.

const int MaxCcuReached = 0x7FFF - 10

(32757) Authorization on the **Photon** Cloud failed because the concurrent users (CCU) limit of the app's subscription is reached.

Unless you have a plan with "CCU Burst", clients might fail the authentication step during connect. Affected client are unable to call operations. Please note that players who end a game and return to the master server will disconnect and re-connect, which means that they just played and are rejected in the next minute / re-connect. This is a temporary measure. Once the CCU is below the limit, players will be able to connect and play again.

OpAuthorize is part of connection workflow but only on the **Photon** Cloud, this error can happen. Self-hosted **Photon** servers with a CCU limited license won't let a client connect at all.

const int NoRandomMatchFound = 0x7FFF - 7

(32760) Random matchmaking only succeeds if a room exists that's neither closed nor full. Repeat in a few seconds or create a new room.

const int Ok = 0

(0) is always "OK", anything else an error or specific situation.

const int OperationNotAllowedInCurrentState = -3

(-3) Operation can't be executed yet (e.g. OpJoin can't be called before being authenticated, RaiseEvent can't be used before getting into a room).

Before you call any operations on the Cloud servers, the automated client workflow must complete its authorization. In PUN, wait until State is: JoinedLobby or ConnectedToMasterserver

const int PluginMismatch = 0x7FFF - 16

(32751) CreateGame/JoinGame/Join operation fails if expected plugin does not correspond to loaded one.

const int PluginReportedError = 0x7FFF - 15

(32752) A server-side plugin (or webhook) failed to execute and reported an error. Check the OperationResponse.DebugMessage.

const int ServerFull = 0x7FFF - 5

(32762) Not in use currently.

const int SlotError = 32742

(32742) Server error during matchmaking with slot reservation. E.g. the reserved slots can not exceed MaxPlayers.

const int UserBlocked = 0x7FFF - 6

(32761) Not in use currently.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	EventCode		

[Public Attributes](#) | [List of all members](#)

EventCode Class Reference

Class for constants. These values are for events defined by **Photon** Loadbalancing. [More...](#)

Public Attributes

const byte **GameList** = 230
(230) Initial list of RoomInfos (in lobby on Master) [More...](#)

const byte **GameListUpdate** = 229
(229) Update of RoomInfos to be merged into "initial" list (in lobby on Master) [More...](#)

const byte **QueueState** = 228
(228) Currently not used. State of queueing in case of server-full [More...](#)

const byte **Match** = 227
(227) Currently not used. Event for matchmaking [More...](#)

const byte **AppStats** = 226
(226) Event with stats about this application (players, rooms, etc) [More...](#)

const byte **LobbyStats** = 224
(224) This event provides a list of lobbies with their player and game counts. [More...](#)

const byte **AzureNodeInfo** = 210
(210) Internally used in case of hosting by Azure [More...](#)

const byte **Join** = (byte)255
(255) Event Join: someone joined the game. The new actorNumber is provided as well as the properties of that actor (if set in OpJoin). [More...](#)

const byte **Leave** = (byte)254
(254) Event Leave: The player who left the game can be identified by the actorNumber. [More...](#)

const byte **PropertiesChanged** = (byte)253
(253) When you call OpSetProperties with the broadcast option "on", this event is fired. It contains the properties being set. [More...](#)

const byte **SetProperties** = (byte)253
(253) When you call OpSetProperties with the broadcast option "on", this event is fired. It contains the properties being set. [More...](#)

const byte **ErrorInfo** = 251
(252) When player left game unexpected and the room has a playerTtl != 0, this event is fired to let everyone know about the timeout. [More...](#)

const byte **CacheSliceChanged** = 250
(250) Sent by **Photon** when the event cache slice was changed. Done by OpRaiseEvent. [More...](#)

const byte **AuthEvent** = 223
(223) Sent by **Photon** to update a token before it times out. [More...](#)

Detailed Description

Class for constants. These values are for events defined by **Photon** Loadbalancing.

They start at 255 and go DOWN. Your own in-game events can start at 0. **Pun** uses these constants internally.

Member Data Documentation

const byte AppStats = 226

(226) Event with stats about this application (players, rooms, etc)

const byte AuthEvent = 223

(223) Sent by **Photon** to update a token before it times out.

const byte AzureNodeInfo = 210

(210) Internally used in case of hosting by Azure

const byte CacheSliceChanged = 250

(250) Sent by **Photon** when the event cache slice was changed.
Done by OpRaiseEvent.

const byte ErrorInfo = 251

(252) When player left game unexpected and the room has a
playerTtl != 0, this event is fired to let everyone know about the
timeout.

Obsolete. Replaced by Leave. public const byte Disconnect =
LiteEventCode.Disconnect;

(251) Sent by **Photon** Cloud when a plugin-call or webhook-call
failed. Usually, the execution on the server continues, despite the

issue. Contains: **ParameterCode.Info**.

See also

<https://doc.photonengine.com/en-us/realtime/current/reference/webhooks::options>

const byte GameList = 230

(230) Initial list of RoomInfos (in lobby on Master)

const byte GameListUpdate = 229

(229) Update of RoomInfos to be merged into "initial" list (in lobby on Master)

const byte Join = (byte)255

(255) Event Join: someone joined the game. The new actorNumber is provided as well as the properties of that actor (if set in OpJoin).

const byte Leave = (byte)254

(254) Event Leave: The player who left the game can be identified by the actorNumber.

const byte LobbyStats = 224

(224) This event provides a list of lobbies with their player and game counts.

const byte Match = 227

(227) Currently not used. Event for matchmaking

const byte PropertiesChanged = (byte)253

(253) When you call OpSetProperties with the broadcast option "on", this event is fired. It contains the properties being set.

const byte QueueState = 228

(228) Currently not used. State of queueing in case of server-full

const byte SetPropertyes = (byte)253

(253) When you call OpSetProperties with the broadcast option "on", this event is fired. It contains the properties being set.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	EventExt		

EventExt Class Reference

[Static Public Member Functions](#) |
[List of all members](#)

Static Public Member Functions

static int **Sender** (this eventdata ev)

static object **CustomData** (this eventdata ev)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	Extensions		

[Static Public Member Functions](#) |

[List of all members](#)

Extensions Class Reference

This static class defines some useful extension methods for several existing classes (e.g. Vector3, float and others). [More...](#)

Static Public Member Functions

static void **Merge** (this IDictionary target, IDictionary addHash)
Merges all keys from addHash into the target. Adds new keys and updates the values of existing keys in target. [More...](#)

static void **MergeStringKeys** (this IDictionary target, IDictionary addHash)
Merges keys of type string to target Hashtable. [More...](#)

static string **ToStringFull** (this IDictionary origin)
Helper method for debugging of IDictionary content, including type-information. Using this is not performant. [More...](#)

static string **ToStringFull< T >** (this List< T > data)
Helper method for debugging of List<T> content. Using this is not performant. [More...](#)

static string **ToStringFull** (this object[] data)
Helper method for debugging of object[] content. Using this is not performant. [More...](#)

static Hashtable **StripToStringKeys** (this IDictionary original)
This method copies all string-typed keys of the original into a new Hashtable. [More...](#)

static void **StripKeysWithNullValues** (this IDictionary original)
This removes all key-value pairs that have a null-reference as value. **Photon** properties are removed by setting their value to null. Changes the original passed IDictionary! [More...](#)

static bool **Contains** (this int[] target, int nr)

Checks if a particular integer value is in an int-array. [More...](#)

Detailed Description

This static class defines some useful extension methods for several existing classes (e.g. Vector3, float and others).

Member Function Documentation

```
static bool Contains ( this int[] target,  
                      int      nr  
                      )
```

static

Checks if a particular integer value is in an int-array.

This might be useful to look up if a particular actorNumber is in the list of players of a room.

Parameters

target The array of ints to check.

nr The number to lookup in target.

Returns

True if nr was found in target.

```
static void Merge ( this IDictionary target,  
                   IDictionary      addHash  
                   )
```

static

Merges all keys from addHash into the target. Adds new keys and updates the values of existing keys in target.

Parameters

target The IDictionary to update.

addHash The IDictionary containing data to merge into target.

```
static void MergeStringKeys ( this IDictionary target,  
                              IDictionary      addHash  
                              )
```

static

Merges keys of type string to target Hashtable.

Does not remove keys from target (so non-string keys CAN be in target if they were before).

Parameters

target The target IDictionary passed in plus all string-typed keys from the addHash.

addHash A IDictionary that should be merged partly into target to update it.

static void
StripKeysWithNullValues (this IDictionary **original**) static

This removes all key-value pairs that have a null-reference as value. **Photon** properties are removed by setting their value to null. Changes the original passed IDictionary!

Parameters

original The IDictionary to strip of keys with null-values.

static Hashtable
StripToStringKeys (this IDictionary **original**) static

This method copies all string-typed keys of the original into a new Hashtable.

Does not recurse (!) into hashes that might be values in the root-hash. This does not modify the original.

Parameters

original The original IDictionary to get string-typed keys from.

Returns

New Hashtable containing only string-typed keys of the original.

static string ToStringFull (this IDictionary **origin)**

static

Helper method for debugging of IDictionary content, including type-information. Using this is not performant.

Should only be used for debugging as necessary.

Parameters

origin Some Dictionary or Hashtable.

Returns

String of the content of the IDictionary.

static string ToStringFull (this object[] **data)**

static

Helper method for debugging of object[] content. Using this is not performant.

Should only be used for debugging as necessary.

Parameters

data Any object[].

Returns

A comma-separated string containing each value's ToString().

static string ToStringFull< T > (this List< T > **data)**

static

Helper method for debugging of List<T> content. Using this is not performant.

Should only be used for debugging as necessary.

Parameters

data Any List<T> where T implements ToString().

Returns

A comma-separated string containing each value's ToString().



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	FriendInfo		

[Public Member Functions](#) | [Properties](#) |

[List of all members](#)

FriendInfo Class Reference

Used to store info about a friend's online state and in which room he/she is. [More...](#)

Public Member Functions

override string **ToString** ()

Properties

string **Name** [get]

string **UserId** [get, protected set]

bool **IsOnline** [get, protected set]

string **Room** [get, protected set]

bool **IsInRoom** [get]

Detailed Description

Used to store info about a friend's online state and in which room he/she is.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	GamePropertyKey		

[Public Attributes](#) | [List of all members](#)

GamePropertyKey Class Reference

Class for constants. These (byte) values are for "well known" room/game properties used in **Photon** Loadbalancing. [More...](#)

Public Attributes

const byte **MaxPlayers** = 255
(255) Max number of players that "fit" into this room. 0 is for "unlimited". [More...](#)

const byte **IsVisible** = 254
(254) Makes this room listed or not in the lobby on master. [More...](#)

const byte **IsOpen** = 253
(253) Allows more players to join a room (or not). [More...](#)

const byte **PlayerCount** = 252
(252) Current count of players in the room. Used only in the lobby on master. [More...](#)

const byte **Removed** = 251
(251) True if the room is to be removed from room listing (used in update to room list in lobby on master) [More...](#)

const byte **PropsListedInLobby** = 250
(250) A list of the room properties to pass to the **RoomInfo** list in a lobby. This is used in CreateRoom, which defines this list once per room. [More...](#)

const byte **CleanupCacheOnLeave** = 249
(249) Equivalent of Operation Join parameter CleanupCacheOnLeave. [More...](#)

const byte **MasterClientId** = (byte)248
(248) Code for MasterClientId, which is synced by server. When sent as op-parameter this is (byte)203. As room property this is (byte)248. [More...](#)

const byte **ExpectedUsers** = (byte)247
(247) Code for ExpectedUsers in a room. Matchmaking keeps a slot open for the players with these userIDs. [More...](#)

const byte **PlayerTtl** = (byte)246
(246) **Player** Time To Live. How long any player can be inactive (due to disconnect or leave) before the user gets removed from the playerlist (freeing a slot). [More...](#)

const byte **EmptyRoomTtl** = (byte)245
(245) **Room** Time To Live. How long a room stays available (and in server-memory), after the last player becomes inactive. After this time, the room gets persisted or destroyed. [More...](#)

Detailed Description

Class for constants. These (byte) values are for "well known" room/game properties used in **Photon** Loadbalancing.

Pun uses these constants internally. "Custom properties" have to use a string-type as key. They can be assigned at will.

Member Data Documentation

const byte CleanupCacheOnLeave = 249

(249) Equivalent of Operation Join parameter CleanupCacheOnLeave.

const byte EmptyRoomTtl = (byte)245

(245) **Room** Time To Live. How long a room stays available (and in server-memory), after the last player becomes inactive. After this time, the room gets persisted or destroyed.

const byte ExpectedUsers = (byte)247

(247) Code for ExpectedUsers in a room. Matchmaking keeps a slot open for the players with these userIDs.

const byte IsOpen = 253

(253) Allows more players to join a room (or not).

const byte IsVisible = 254

(254) Makes this room listed or not in the lobby on master.

const byte MasterClientId = (byte)248

(248) Code for MasterClientId, which is synced by server. When sent as op-parameter this is (byte)203. As room property this is (byte)248.

Tightly related to **ParameterCode.MasterClientId**.

const byte MaxPlayers = 255

(255) Max number of players that "fit" into this room. 0 is for "unlimited".

const byte PlayerCount = 252

(252) Current count of players in the room. Used only in the lobby on master.

const byte PlayerTtl = (byte)246

(246) **Player** Time To Live. How long any player can be inactive (due to disconnect or leave) before the user gets removed from the playerlist (freeing a slot).

const byte PropsListedInLobby = 250

(250) A list of the room properties to pass to the **RoomInfo** list in a lobby. This is used in CreateRoom, which defines this list once per room.

const byte Removed = 251

(251) True if the room is to be removed from room listing (used in

update to room list in lobby on master)

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	LoadBalancingClient		

LoadBalancingClient Class Reference

[Public Member Functions](#) | [Public Attributes](#) |
[Properties](#) | [Events](#) | [List of all members](#)

This class implements the **Photon** LoadBalancing workflow by using a **LoadBalancingPeer**. It keeps a state and will automatically execute transitions between the Master and Game Servers. [More...](#)

Inherits IPhotonPeerListener.

Public Member Functions

LoadBalancingClient (ConnectionProtocol protocol=ConnectionProtocol.Udp)
Creates a **LoadBalancingClient** with UDP protocol or the one specified. [More...](#)

LoadBalancingClient (string masterAddress, string appld, string gameVersion, ConnectionProtocol protocol=ConnectionProtocol.Udp)
Creates a **LoadBalancingClient**, setting various values needed before connecting. [More...](#)

virtual bool **Connect** ()
Starts the "process" to connect to a Master Server, using MasterServerAddress and Appld properties. [More...](#)

bool **ConnectToNameServer** ()
Connects to the NameServer for **Photon** Cloud, where a region and server list can be obtained. [More...](#)

bool **ConnectToRegionMaster** (string region)
Connects you to a specific region's Master Server, using the Name Server to find the IP. [More...](#)

bool **ReconnectToMaster** ()
Can be used to reconnect to the master server after a disconnect. [More...](#)

bool **ReconnectAndRejoin** ()
Can be used to return to a room quickly, by directly reconnecting to a game server to rejoin a room. [More...](#)

void **Disconnect** ()
Disconnects this client from any server and sets this.State if the connection is successfully closed. [More...](#)

void **Service ()**

This method dispatches all available incoming commands and then sends this client's outgoing commands. It uses DispatchIncomingCommands and SendOutgoingCommands to do that. [More...](#)

bool **OpFindFriends** (string[] friendsToFind)

Request the rooms and online status for a list of friends. All clients should set a unique UserId before connecting. The result is available in this.FriendList. [More...](#)

bool **OpJoinLobby** (TypedLobby lobby)

If already connected to a Master Server, this joins the specified lobby. This request triggers an **OnOperationResponse()** call and the callback OnJoinedLobby(). [More...](#)

bool **OpLeaveLobby** ()

Opposite of joining a lobby. You don't have to explicitly leave a lobby to join another (client can be in one max, at any time). [More...](#)

bool **OpJoinRandomRoom** (OpJoinRandomRoomParams opJoinRandomRoomParams=null)

Joins a random room that matches the filter. Will callback: OnJoinedRoom or OnJoinRandomFailed. [More...](#)

bool **OpCreateRoom** (EnterRoomParams enterRoomParams)

Creates a new room. Will callback: OnCreatedRoom and OnJoinedRoom or OnCreateRoomFailed. [More...](#)

bool **OpJoinOrCreateRoom** (EnterRoomParams enterRoomParams)

Joins a specific room by name and creates it on demand. Will callback: OnJoinedRoom or

OnJoinRoomFailed. [More...](#)

bool **OpJoinRoom** (**EnterRoomParams** enterRoomParams)
Joins a room by name. Will callback: OnJoinedRoom or OnJoinRoomFailed. [More...](#)

bool **OpRejoinRoom** (string roomName)
Rejoins a room by roomName (using the userID internally to return). Will callback: OnJoinedRoom or OnJoinRoomFailed. [More...](#)

bool **OpLeaveRoom** (bool becomeInactive, bool sendAuthCookie=false)
Leaves the current room, optionally telling the server that the user is just becoming inactive. Will callback: OnLeftRoom. [More...](#)

bool **OpGetGameList** (**TypedLobby** typedLobby, string sqlLobbyFilter)
Gets a list of games matching a SQL-like where clause. [More...](#)

bool **OpSetCustomPropertiesOfActor** (int actorNr, Hashtable propertiesToSet, Hashtable expectedProperties=null, **WebFlags** webFlags=null)
Updates and synchronizes a **Player's** Custom Properties. Optionally, expectedProperties can be provided as condition. [More...](#)

bool **OpSetCustomPropertiesOfRoom** (Hashtable propertiesToSet, Hashtable expectedProperties=null, **WebFlags** webFlags=null)
Updates and synchronizes this **Room's** Custom Properties. Optionally, expectedProperties can be provided as condition. [More...](#)

bool **OpSetPropertiesOfRoom** (Hashtable gameProperties,

Hashtable expectedProperties=null, **WebFlags** webFlags=null)
Internally used to cache and set properties (including well known properties). [More...](#)

virtual bool **OpRaiseEvent** (byte eventCode, object customEventContent, **RaiseEventOptions** raiseEventOptions, SendOptions sendOptions)
Send an event with custom code/type and any content to the other players in the same room. [More...](#)

virtual bool **OpChangeGroups** (byte[] groupsToRemove, byte[] groupsToAdd)
Operation to handle this client's interest groups (for events in room). [More...](#)

void **ChangeLocalID** (int newID)
Internally used to set the LocalPlayer's ID (from -1 to the actual in-room ID). [More...](#)

virtual void **DebugReturn** (DebugLevel level, string message)
Debug output of low level api (and this client). [More...](#)

virtual void **OnOperationResponse** (OperationResponse operationResponse)
Uses the OperationResponses provided by the server to advance the internal state and call ops as needed. [More...](#)

virtual void **OnStatusChanged** (StatusCode statusCode)
Uses the connection's statusCodes to advance the internal state and call operations as needed. [More...](#)

virtual void **OnEvent** (EventData photonEvent)
Uses the photonEvent's provided by the server to advance the internal state and call ops as needed. [More...](#)

virtual void **OnMessage** (object message)

In **Photon** 4, "raw messages" will get their own callback method in the interface. Not used yet. [More...](#)

bool **OpWebRpc** (string uriPath, object parameters, bool sendAuthCookie=false)

This operation makes **Photon** call your custom web-service by path/name with the given parameters (converted into Json). [More...](#)

void **AddCallbackTarget** (object target)

Registers an object for callbacks for the implemented callback-interfaces. [More...](#)

void **RemoveCallbackTarget** (object target)

Unregisters an object from callbacks for the implemented callback-interfaces. [More...](#)

Public Attributes

AuthModeOption	AuthMode = AuthModeOption.AuthModeOption Enables the new Authentication workflow. More...
EncryptionMode	EncryptionMode = EncryptionMode.PayloadEncryptionMode Defines how the communication gets encrypted. More...
ConnectionProtocol	ExpectedProtocol = ConnectionProtocol.Udp The protocol which will be used on Master- and GameServer. More...
string	NameServerHost = "ns.exitgames.com" Name Server Host Name for Photon Cloud. Without port and without any prefix. More...
string	NameServerHttp = "http://ns.exitgames.com:80/photons" Name Server for HTTP connection to the Photon Cloud. Includes prefix and port. More...
ConnectionCallbacksContainer	ConnectionCallbackTargets = new ConnectionCallbacksContainer() Wraps up the target objects for a group of callbacks, so they can be called conveniently. More...
MatchMakingCallbacksContainer	MatchMakingCallbackTargets = new MatchMakingCallbacksContainer() Wraps up the target objects for a group of callbacks, so they can be called conveniently. More...

group of callbacks, so they can be called conveniently. [More...](#)

bool EnableLobbyStatistics

If enabled, the client will get a list of available lobbies from the Master Server. [More...](#)

RegionHandler RegionHandler

Contains the list of enabled regions this client may use. Null, unless the client got a response to OpGetRegions. [More...](#)

Properties

LoadBalancingPeer **LoadBalancingPeer** [get]

The client uses a **LoadBalancingPeer** as API to communicate with the server. This is public for ease-of-use: Some methods like `OpRaiseEvent` are not relevant for the connection state and don't need a override. [More...](#)

string **AppVersion** [get, set]

The version of your client. A new version also creates a new "virtual app" to separate players from older client versions. [More...](#)

string **AppId** [get, set]

The AppID as assigned from the **Photon** Cloud. If you host yourself, this is the "regular" **Photon** Server Application Name (most likely: "LoadBalancing"). [More...](#)

AuthenticationValues **AuthValues** [get, set]

User authentication values to be sent to the **Photon** server right after connecting. [More...](#)

bool **IsUsingNameServer** [get, set]

True if this client uses a NameServer to get the Master Server address. [More...](#)

string **NameServerAddress** [get]

Name Server Address for **Photon** Cloud (based on current protocol). You can use the default values and usually won't have to set this value. [More...](#)

bool **UseAlternativeUdpPorts** [get, set]
Use the alternative ports for UDP connections in the Public Cloud (27000 to 27003). [More...](#)

string **CurrentServerAddress** [get]
The currently used server address (if any). The type of server is define by Server property. [More...](#)

string **MasterServerAddress** [get, set]
Your Master Server address. In PhotonCloud, call **ConnectToRegionMaster()** to find your Master Server. [More...](#)

string **GameServerAddress** [get, set]
The game server's address for a particular room. In use temporarily, as assigned by master. [More...](#)

ServerConnection **Server** [get]
The server this client is currently connected or connecting to. [More...](#)

ClientState **State** [get, set]
Current state this client is in. Careful: several states are "transitions" that lead to other states. [More...](#)

bool **IsConnected** [get]
Returns if this client is currently connected or connecting to some type of server. [More...](#)

bool **IsConnectedAndReady** [get]
A refined version of IsConnected which is true only if your connection is ready to send

operations. [More...](#)

DisconnectCause **DisconnectedCause** [get, protected set]
Summarizes (aggregates) the different causes for disconnects of a client. [More...](#)

bool **InLobby** [get]
Internal value if the client is in a lobby. [More...](#)

TypedLobby **CurrentLobby** [get, set]
The lobby this client currently uses. Defined when joining a lobby or creating rooms [More...](#)

Player **LocalPlayer** [get, set]
The local player is never null but not valid unless the client is in a room, too. The ID will be -1 outside of rooms. [More...](#)

string **NickName** [get, set]
The nickname of the player (synced with others). Same as client.LocalPlayer.NickName. [More...](#)

string **UserId** [get, set]
An ID for this user. Sent in OpAuthenticate when you connect. If not set, the PlayerName is applied during connect. [More...](#)

Room **CurrentRoom** [get]
The current room this client is connected to (null if none available). [More...](#)

bool **InRoom** [get]
Is true while being in a room (this.state == **ClientState.Joined**). [More...](#)

int **PlayersOnMasterCount** [get, set]
Statistic value available on master server: Players on master (looking for games). [More...](#)

int **PlayersInRoomsCount** [get, set]
Statistic value available on master server: Players in rooms (playing). [More...](#)

int **RoomsCount** [get, set]
Statistic value available on master server: Rooms currently created. [More...](#)

bool **IsFetchingFriendList** [get]
Internal flag to know if the client currently fetches a friend list. [More...](#)

string **CloudRegion** [get]
The cloud region this client connects to. Set by **ConnectToRegionMaster()**. Not set if you don't use a NameServer! [More...](#)

Events

Action< **ClientState**, **ClientState** > **StateChanged**
Register a method to be called when this client's ClientState gets set. [More...](#)

Action< **EventData** > **EventReceived**
Register a method to be called when an event got dispatched. Gets called after the **LoadBalancingClient** handled the internal events first. [More...](#)

Action< **OperationResponse** > **OpResponseReceived**
Register a method to be called when an operation response is received. [More...](#)

Detailed Description

This class implements the **Photon** LoadBalancing workflow by using a **LoadBalancingPeer**. It keeps a state and will automatically execute transitions between the Master and Game Servers.

This class (and the **Player** class) should be extended to implement your own game logic. You can override `CreatePlayer` as "factory" method for Players and return your own **Player** instances. The State of this class is essential to know when a client is in a lobby (or just on the master) and when in a game where the actual gameplay should take place. Extension notes: An extension of this class should override the methods of the `IPhotonPeerListener`, as they are called when the state changes. Call `base.method` first, then pick the operation or state you want to react to and put it in a switch-case. We try to provide demo to each platform where this api can be used, so lookout for those.

Constructor & Destructor Documentation

LoadBalancingClient (**ConnectionProtocol** **protocol** = **ConnectionPr**

Creates a **LoadBalancingClient** with UDP protocol or the one specified

Parameters

protocol Specifies the network protocol to use for connections.

LoadBalancingClient (**string** **masterAddress**,
 string **appld**,
 string **gameVersion**,
 ConnectionProtocol **protocol** = **ConnectionPr**
)

Creates a **LoadBalancingClient**, setting various values needed before connecting.

Parameters

masterAddress The Master Server's address to connect to. Used Connect.

appld The Appld of this title. Needed for the **Photon** Clk in the Dashboard.

gameVersion A version for this client/build. In the **Photon** Cloud are separated by Appld, GameVersion and **Region**

protocol Specifies the network protocol to use for connecti

Member Function Documentation

void AddCallbackTarget (object **target)**

Registers an object for callbacks for the implemented callback-interfaces.

The covered callback interfaces are: **ICallbacks**, **IMatchmakingCallbacks**, **ILobbyCallbacks**, **IInRoomCallbacks**, **IOnEventCallback** and **IWebRpcCallback**.

See: [The object that registers to get callbacks from this client.](#)

void ChangeLocalID (int **newID)**

Internally used to set the LocalPlayer's ID (from -1 to the actual in-room ID).

Parameters

newID New actor ID (a.k.a actorNr) assigned when joining a room.

virtual bool Connect ()

virtual

Starts the "process" to connect to a Master Server, using MasterServerAddress and AppId properties.

To connect to the **Photon** Cloud, use **ConnectToRegionMaster()**.

The process to connect includes several steps: the actual connecting, establishing encryption, authentication (of app and optionally the user) and connecting to the MasterServer

Users can connect either anonymously or use "Custom Authentication" to verify each individual player's login. Custom Authentication in **Photon** uses external services and communities to verify users. While the client provides a user's info, the service setup is done in the **Photon** Cloud Dashboard. The parameter `authValues` will set this.`AuthValues` and use them in the connect process.

Connecting to the **Photon** Cloud might fail due to:

- Network issues (`OnStatusChanged()` `StatusCode.ExceptionOnConnect`)
- **Region** not available (`OnOperationResponse()` for `OpAuthenticate` with `ReturnCode == ErrorCode.InvalidRegion`)
- Subscription CCU limit reached (`OnOperationResponse()` for `OpAuthenticate` with `ReturnCode == ErrorCode.MaxCcuReached`)

bool ConnectToNameServer ()

Connects to the NameServer for **Photon** Cloud, where a region and server list can be obtained.

`OpGetRegions`

Returns

If the workflow was started or failed right away.

bool ConnectToRegionMaster (string **region)**

Connects you to a specific region's Master Server, using the Name Server to find the IP.

Returns

If the operation could be sent. If false, no operation was sent.

virtual void DebugReturn (DebugLevel **level,**

string message
) virtual

Debug output of low level api (and this client).

This method is not responsible to keep up the state of a **LoadBalancingClient**. Calling `base.DebugReturn` on overrides is optional.

void Disconnect ()

Disconnects this client from any server and sets `this.State` if the connection is successfully closed.

virtual void OnEvent (EventArgs photonEvent) virtual

Uses the `photonEvent`'s provided by the server to advance the internal state and call ops as needed.

This method is essential to update the internal state of a **LoadBalancingClient**. Overriding methods must call `base.OnEvent`.

virtual void OnMessage (object message) virtual

In **Photon 4**, "raw messages" will get their own callback method in the interface. Not used yet.

virtual void OnOperationResponse (OperationResponse operationResponse)

Uses the `OperationResponses` provided by the server to advance the internal state and call ops as needed.

When this method finishes, it will call your `OnOpResponseAction` (if any)

This way, you can get any operation response without overriding this client.

To implement a more complex game/app logic, you should implement your own class that inherits the **LoadBalancingClient**. Override this method to use your own operation-responses easily.

This method is essential to update the internal state of a **LoadBalancingClient**, so overriding methods must call `base.OnOperationResponse()`.

Parameters

operationResponse Contains the server's response for an operation called by this peer.

virtual void OnStatusChanged (StatusCode **statusCode)** virtual

Uses the connection's statusCodes to advance the internal state and call operations as needed.

This method is essential to update the internal state of a **LoadBalancingClient**. Overriding methods must call `base.OnStatusChanged`.

virtual bool OpChangeGroups (byte[] **groupsToRemove,
byte[] **groupsToAdd**
)** virtual

Operation to handle this client's interest groups (for events in room).

Note the difference between passing null and `byte[0]`: null won't add/remove any groups. `byte[0]` will add/remove all (existing) groups. First, removing groups is executed. This way, you could leave all groups and join only the ones provided.

Changes become active not immediately but when the server executes this operation (approximately $RTT/2$).

Parameters

groupsToRemove Groups to remove from interest. Null will not remove any. A byte[0] will remove all.

groupsToAdd Groups to add to interest. Null will not add any. A byte[0] will add all current.

Returns

If operation could be enqueued for sending. Sent when calling: Service or SendOutgoingCommands.

bool OpCreateRoom (EnterRoomParams **enterRoomParams**)

Creates a new room. Will callback: OnCreatedRoom and OnJoinedRoom or OnCreateRoomFailed.

When successful, the client will enter the specified room and callback both OnCreatedRoom and OnJoinedRoom. In all error cases, OnCreateRoomFailed gets called.

Creating a room will fail if the room name is already in use or when the **RoomOptions** clashing with one another. Check the **EnterRoomParams** reference for the various room creation options.

This method can only be called while the client is connected to a Master Server so you should implement the callback OnConnectedToMaster. Check the return value to make sure the operation will be called on the server. Note: There will be no callbacks if this method returned false.

When you're in the room, this client's State will become **ClientState.Joined**.

When entering a room, this client's **Player** Custom Properties will be sent to the room. Use LocalPlayer.SetCustomProperties to set them, even while not yet in the room. Note that the player properties will be cached locally and are not wiped when leaving a room.

You can define an array of expectedUsers, to block player slots in the room for these users. The corresponding feature in **Photon** is

called "Slot Reservation" and can be found in the doc pages.

Parameters

enterRoomParams Definition of properties for the room to create.

Returns

If the operation could be sent currently (requires connection to Master Server).

bool OpFindFriends (string[] friendsToFind)

Request the rooms and online status for a list of friends. All clients should set a unique UserId before connecting. The result is available in this.FriendList.

Used on Master Server to find the rooms played by a selected list of users. The result will be stored in LoadBalancingClient.FriendList, which is null before the first server response.

Users identify themselves by setting a UserId in the **LoadBalancingClient** instance. This will send the ID in OpAuthenticate during connect (to master and game servers). Note: Changing a player's name doesn't make sense when using a friend list.

The list of usernames must be fetched from some other source (not provided by **Photon**).

Internal: The server response includes 2 arrays of info (each index matching a friend from the request):

ParameterCode.FindFriendsResponseOnlineList = bool[] of online states **ParameterCode.FindFriendsResponseRoomIdList** = string[] of room names (empty string if not in a room)

Parameters

friendsToFind Array of friend's names (make sure they are unique).

Returns

If the operation could be sent (requires connection).

```
bool OpGetGameList ( TypedLobby typedLobby,  
                    string      sqlLobbyFilter  
                    )
```

Gets a list of games matching a SQL-like where clause.

Operation is only available for lobbies of type SqlLobby. This is an async request which triggers a **OnOperationResponse()** call.

https://doc.photonengine.com/en-us/realtime/current/reference/matchmaking-and-lobby::sql_lobby_type

Parameters

typedLobby The lobby to query. Has to be of type SqlLobby.
sqlLobbyFilter The sql query statement.

Returns

If the operation could be sent (has to be connected).

```
bool OpJoinLobby ( TypedLobby lobby )
```

If already connected to a Master Server, this joins the specified lobby. This request triggers an **OnOperationResponse()** call and the callback **OnJoinedLobby()**.

Parameters

lobby The lobby to join. Use null for default lobby.

Returns

If the operation could be sent. False, if the client is not **IsConnectedAndReady** or when it's not connected to a Master Server.

bool

OpJoinOrCreateRoom (EnterRoomParams enterRoomParams)

Joins a specific room by name and creates it on demand. Will callback: OnJoinedRoom or OnJoinRoomFailed.

Useful when players make up a room name to meet in: All involved clients call the same method and whoever is first, also creates the room.

When successful, the client will enter the specified room. The client which creates the room, will callback both OnCreatedRoom and OnJoinedRoom. Clients that join an existing room will only callback OnJoinedRoom. In all error cases, OnJoinRoomFailed gets called.

Joining a room will fail, if the room is full, closed or when the user already is present in the room (checked by userId).

To return to a room, use OpRejoinRoom.

This method can only be called while the client is connected to a Master Server so you should implement the callback OnConnectedToMaster. Check the return value to make sure the operation will be called on the server. Note: There will be no callbacks if this method returned false.

This client's State is set to **ClientState.Joining** immediately, when the operation could be called. In the background, the client will switch servers and call various related operations.

When you're in the room, this client's State will become **ClientState.Joined**.

If you set room properties in roomOptions, they get ignored when the room is existing already. This avoids changing the room properties by late joining players.

When entering a room, this client's **Player** Custom Properties will be sent to the room. Use LocalPlayer.SetCustomProperties to set them, even while not yet in the room. Note that the player properties will be

cached locally and are not wiped when leaving a room.

You can define an array of `expectedUsers`, to block player slots in the room for these users. The corresponding feature in **Photon** is called "Slot Reservation" and can be found in the doc pages.

Parameters

enterRoomParams Definition of properties for the room to create or join.

Returns

If the operation could be sent currently (requires connection to Master Server).

bool

OpJoinRandomRoom (OpJoinRandomRoomParams opJoinRandomRoomParams)

Joins a random room that matches the filter. Will callback: `OnJoinedRoom`, `OnJoinRandomFailed`.

Used for random matchmaking. You can join any room or one with specific `opJoinRandomRoomParams`.

You can use `expectedCustomRoomProperties` and `expectedMaxPlayerCount` to filter rooms. If you set `expectedCustomRoomProperties`, a room must have 1 or more values set at Custom Properties. You need to define which Custom Properties are available for matchmaking when you create a room. See: `OpCreateRoom`, `RoomOptions`, `roomOptions`, `TypedLobby`, `lobby`)

This operation fails if no rooms are fitting or available (all full, closed or fail when actually joining the room which was found. Rooms may close, anytime).

This method can only be called while the client is connected to a Master Server. Implement the callback `OnConnectedToMaster`. Check the return value. If the operation will be called on the server. Note: There will be no callbacks if the operation fails.

This client's State is set to **ClientState.Joining** immediately, when the operation is called.

called. In the background, the client will switch servers and call various

When you're in the room, this client's State will become **ClientState.Joining**

When entering a room, this client's **Player** Custom Properties will be set via LocalPlayer.SetCustomProperties to set them, even while not yet in the room. Player properties will be cached locally and are not wiped when leaving

More about matchmaking: <https://doc.photonengine.com/en-us/realtime/current/reference/matchmaking-and-lobby>

You can define an array of expectedUsers, to block player slots in the room. The corresponding feature in **Photon** is called "Slot Reservation" and can be found in the Photon documentation.

Parameters

opJoinRandomRoomParams Optional definition of properties to use for matchmaking.

Returns

If the operation could be sent currently (requires connection to MasterServer)

bool OpJoinRoom (EnterRoomParams enterRoomParams)

Joins a room by name. Will callback: OnJoinedRoom or OnJoinRoomFailed.

Useful when using lobbies or when players follow friends or invite each other.

When successful, the client will enter the specified room and callback via OnJoinedRoom. In all error cases, OnJoinRoomFailed gets called.

Joining a room will fail if the room is full, closed, not existing or when the user already is present in the room (checked by userId).

To return to a room, use OpRejoinRoom. When players invite each other and it's unclear who's first to respond, use OpJoinOrCreateRoom instead.

This method can only be called while the client is connected to a Master Server so you should implement the callback `OnConnectedToMaster`. Check the return value to make sure the operation will be called on the server. Note: There will be no callbacks if this method returned false.

A room's name has to be unique (per region, appid and gameversion). When your title uses a global matchmaking or invitations (e.g. an external solution), keep regions and the game versions in mind to join a room.

This client's State is set to **ClientState.Joining** immediately, when the operation could be called. In the background, the client will switch servers and call various related operations.

When you're in the room, this client's State will become **ClientState.Joined**.

When entering a room, this client's **Player** Custom Properties will be sent to the room. Use `LocalPlayer.SetCustomProperties` to set them, even while not yet in the room. Note that the player properties will be cached locally and are not wiped when leaving a room.

You can define an array of `expectedUsers`, to reserve player slots in the room for friends or party members. The corresponding feature in **Photon** is called "Slot Reservation" and can be found in the doc pages.

Parameters

enterRoomParams Definition of properties for the room to join.

Returns

If the operation could be sent currently (requires connection to Master Server).

bool OpLeaveLobby ()

Opposite of joining a lobby. You don't have to explicitly leave a lobby to join another (client can be in one max, at any time).

Returns

If the operation could be sent (has to be connected).

```
bool OpLeaveRoom ( bool becomeInactive,  
                  bool sendAuthCookie = false  
                  )
```

Leaves the current room, optionally telling the server that the user is just becoming inactive. Will callback: OnLeftRoom.

OpLeaveRoom skips execution when the room is null or the server is not GameServer or the client is disconnecting from GS already. OpLeaveRoom returns false in those cases and won't change the state, so check return of this method.

In some cases, this method will skip the OpLeave call and just call **Disconnect()**, which not only leaves the room but also the server. Disconnect also triggers a leave and so that workflow is quicker.

Parameters

becomeInactive If true, this player becomes inactive in the game and can return later (if PlayerTTL of the room is != 0).

sendAuthCookie WebFlag: Securely transmit the encrypted object AuthCookie to the web service in PathLeave webhook when available

Returns

If the current room could be left (impossible while not in a room).

virtual bool

OpRaiseEvent (byte

object

RaiseEventOptions

SendOptions

eventCode,

customEventContent,

raiseEventOptions,

sendOptions

)

virtual

Send an event with custom code/type and any content to the other players in the same room.

Parameters

eventCode	Identifies this type of event (and the content). Your game's event codes can start with 0.
customEventContent	Any serializable datatype (including Hashtable like the other OpRaiseEvent overloads).
raiseEventOptions	Contains used send options. If you pass null, the default options will be used.
sendOptions	Send options for reliable, encryption etc

Returns

If operation could be enqueued for sending. Sent when calling: Service or SendOutgoingCommands.

bool OpRejoinRoom (string **roomName**)

Rejoins a room by roomName (using the userID internally to return). Will callback: OnJoinedRoom or OnJoinRoomFailed.

Used to return to a room, before this user was removed from the players list. Internally, the userID will be checked by the server, to make sure this user is in the room (active or inactive).

In contrast to join, this operation never adds a players to a room. It will attempt to retake an existing spot in the playerlist or fail. This makes sure the client doesn't accidentally join a room when the game logic meant to re-activate an existing actor in an existing room.

This method will fail on the server, when the room does not exist, can't be loaded (persistent rooms) or when the userID is not in the player list of this room. This will lead to a callback

OnJoinRoomFailed.

Rejoining room will not send any player properties. Instead client will receive up-to-date ones from server. If you want to set new player properties, do it once rejoined.

```
bool  
OpSetCustomPropertiesOfActor ( int          actorNr,  
                               Hashtable propertiesToSet,  
                               Hashtable expectedProperties =  
                               WebFlags webFlags = null  
                               )
```

Updates and synchronizes a **Player's** Custom Properties. Optionally, expectedProperties can be provided as condition.

Custom Properties are a set of string keys and arbitrary values which is synchronized for the players in a **Room**. They are available when the c enters the room, as they are in the response of OpJoin and OpCreate.

Custom Properties either relate to the (current) **Room** or a **Player** (in the **Room**).

Both classes locally cache the current key/values and make them available as property: CustomProperties. This is provided only to read them. You must use the method SetCustomProperties to set/modify them.

Any client can set any Custom Properties anytime (when in a room). It's to the game logic to organize how they are best used.

You should call SetCustomProperties only with key/values that are new changed. This reduces traffic and performance.

Unless you define some expectedProperties, setting key/values is always permitted. In this case, the property-setting client will not receive the new values from the server but instead update its local cache in SetCustomProperties.

If you define expectedProperties, the server will skip updates if the server

property-cache does not contain all expectedProperties with the same values. In this case, the property-setting client will get an update from the server and update its cached key/values at about the same time as everyone else.

The benefit of using expectedProperties can be only one client success sets a key from one known value to another. As example: Store who owns an item in a Custom Property "ownedBy". It's 0 initially. When multiple players reach the item, they all attempt to change "ownedBy" from 0 to actorNumber. If you use expectedProperties {"ownedBy", 0} as condition the first player to take the item will have it (and the others fail to set the ownership).

Properties get saved with the game state for Turnbased games (which has IsPersistent = true).

Parameters

actorNr	Defines which player the Custom Properties belong to. ActorID of a player.
propertiesToSet	Hashtable of Custom Properties that change
expectedProperties	Provide some keys/values to use as condition setting the new values. Client must be in room
webFlags	Defines if the set properties should be forwarded to a WebHook. Client must be in room.

bool

```
OpSetCustomPropertiesOfRoom ( Hashtable propertiesToSet,  
                             Hashtable expectedProperties :  
                             WebFlags webFlags = null  
                             )
```

Updates and synchronizes this **Room**'s Custom Properties. Optionally, expectedProperties can be provided as condition.

Custom Properties are a set of string keys and arbitrary values which is synchronized for the players in a **Room**. They are available when the client enters the room, as they are in the response of OpJoin and OpCreate.

Custom Properties either relate to the (current) **Room** or a **Player** (in the **Room**).

Both classes locally cache the current key/values and make them available as property: CustomProperties. This is provided only to read them. You use the method SetCustomProperties to set/modify them.

Any client can set any Custom Properties anytime (when in a room). It's up to the game logic to organize how they are best used.

You should call SetCustomProperties only with key/values that are newly changed. This reduces traffic and performance.

Unless you define some expectedProperties, setting key/values is always permitted. In this case, the property-setting client will not receive the new values from the server but instead update its local cache in SetCustomProperties.

If you define expectedProperties, the server will skip updates if the server's property-cache does not contain all expectedProperties with the same values. In this case, the property-setting client will get an update from the server and update its cached key/values at about the same time as everyone else.

The benefit of using expectedProperties can be only one client successfully sets a key from one known value to another. As example: Store who owns an item in a Custom Property "ownedBy". It's 0 initially. When multiple players reach the item, they all attempt to change "ownedBy" from 0 to actorNumber. If you use expectedProperties {"ownedBy", 0} as condition, the first player to take the item will have it (and the others fail to set the ownership).

Properties get saved with the game state for Turnbased games (which has IsPersistent = true).

Parameters

propertiesToSet	Hashtable of Custom Properties that change
expectedProperties	Provide some keys/values to use as condition setting the new values.
webFlags	Defines web flags for an optional PathProperty webhook.

```
bool  
OpSetPropertiesOfRoom ( Hashtable gameProperties,  
                        Hashtable expectedProperties = null,  
                        WebFlags webFlags = null  
                        )
```

Internally used to cache and set properties (including well known properties).

Requires being in a room (because this attempts to send an operation which will fail otherwise).

```
bool OpWebRpc ( string uriPath,  
                object parameters,  
                bool sendAuthCookie = false  
                )
```

This operation makes **Photon** call your custom web-service by path/na

A WebRPC calls a custom, http-based function on a server you provide configured server-side. The sent parameters get converted from C# type to be converted to C# types and sent back as normal operation response.

To use this feature, you have to setup your server:

For a **Photon** Cloud application, [visit the Dashboard](#) and setup "WebH

The response by **Photon** will call **OnOperationResponse()** with Code: derive the **LoadBalancingClient**, or (much easier) you set a suitable C

It's important to understand that the OperationResponse tells you if the response will contain the values the web-service sent (if any). If the web-service returns another error and a message. This is inside the OperationResponse.

The class **WebRpcResponse** is a helper-class that extracts the most v

To get a WebRPC response, set a OnOpResponseAction:

```
this.OnOpResponseAction = this.OpResponseHandler;
```

It could look like this:

```
public void OpResponseHandler(OperationResponse operationResponse)
{
    if (operationResponse.OperationCode == OperationCode.WebRpcResponse)
    {
        if (operationResponse.ReturnCode != 0)
        {
            Console.WriteLine("WebRpc failed. Response code: " + operationResponse.ReturnCode);
        }
        else
        {
            WebRpcResponse webResponse = new WebRpcResponse(operationResponse.Data);
            Console.WriteLine(webResponse.DebugMessage);

            // do something with the response...
        }
    }
}
```

Parameters

uriPath	The url path to call, relative to the baseUrl configuration.
parameters	The parameters to send to the web-service method.
sendAuthCookie	Defines if the authentication cookie gets sent to the server.

bool ReconnectAndRejoin ()

Can be used to return to a room quickly, by directly reconnecting to a game server to rejoin a room.

Rejoining room will not send any player properties. Instead client will receive up-to-date ones from server. If you want to set new player

properties, do it once rejoined.

Returns

False, if the conditions are not met. Then, this client does not attempt the ReconnectAndRejoin.

bool ReconnectToMaster ()

Can be used to reconnect to the master server after a disconnect.

Common use case: Press the Lock Button on a iOS device and you get disconnected immediately.

void RemoveCallbackTarget (object **target**)

Unregisters an object from callbacks for the implemented callback-interfaces.

The covered callback interfaces are: **IConnectionCallbacks**, **IMatchmakingCallbacks**, **ILobbyCallbacks**, **IInRoomCallbacks**, **IOnEventCallback** and **IWebRpcCallback**.

See: [.Net Callbacks](#)

Parameters

target The object that unregisters from getting callbacks.

void Service ()

This method dispatches all available incoming commands and then server uses DispatchIncomingCommands and SendOutgoingCommands to do

The **Photon** client libraries are designed to fit easily into a game or app the context (thread) in which incoming events and responses are executed of UDP/TCP packages.

Sending packages and dispatching received messages are two separate methods at the cost of control. It calls `DispatchIncomingCommands`

Call this method regularly (2..20 times a second).

This will Dispatch ANY received commands (unless a reliable command) AND will send queued outgoing commands. Fewer calls might be more packets per second, as multiple operations might be combined into one

You could replace `Service` by:

```
while (DispatchIncomingCommands()); //Dispatch until  
SendOutgoingCommands(); //Send a UDP/TCP package with
```

See also

`PhotonPeer.DispatchIncomingCommands`, `PhotonPeer.SendOutgoingCommands`

Member Data Documentation

AuthModeOption AuthMode = AuthModeOption.Auth

Enables the new Authentication workflow.

ConnectionCallbacksContainer ConnectionCallbackTargets = new ConnectionCallbacksContainer()

Wraps up the target objects for a group of callbacks, so they can be called conveniently.

By using Add or Remove, objects can "subscribe" or "unsubscribe" for this group of callbacks.

bool EnableLobbyStatistics

If enabled, the client will get a list of available lobbies from the Master Server.

Set this value before the client connects to the Master Server. While connected to the Master Server, a change has no effect.

Implement OptionalInfoCallbacks.OnLobbyStatisticsUpdate, to get the list of used lobbies.

The lobby statistics can be useful if your title dynamically uses lobbies, depending (e.g.) on current player activity or such. In this case, getting a list of available lobbies, their room-count and player-count can be useful info.

ConnectUsingSettings sets this to the PhotonServerSettings value.

**EncryptionMode EncryptionMode =
EncryptionMode.PayloadEncryption**

Defines how the communication gets encrypted.

**ConnectionProtocol ExpectedProtocol =
ConnectionProtocol.Udp**

The protocol which will be used on Master- and GameServer.

When using AuthMode = AuthModeOption.AuthOnceWss, the client uses a wss-connection on the NameServer but another protocol on the other servers. As the NameServer sends an address, which is different per protocol, it needs to know the expected protocol.

summary>Simplifies getting the token for connect/init requests, if this feature is enabled.

**MatchMakingCallbacksContainer MatchMakingCallbackTargets =
new MatchMakingCallbacksContainer()**

Wraps up the target objects for a group of callbacks, so they can be called conveniently.

By using Add or Remove, objects can "subscribe" or "unsubscribe" for this group of callbacks.

string NameServerHost = "ns.exitgames.com"

Name Server Host Name for **Photon** Cloud. Without port and without any prefix.

string NameServerHttp = "http://ns.exitgames.com:80/photon/n"

Name Server for HTTP connections to the **Photon** Cloud. Includes prefix and port.

RegionHandler RegionHandler

Contains the list of enabled regions this client may use. Null, unless the client got a response to `OpGetRegions`.

Property Documentation

string AppId

get set

The AppID as assigned from the **Photon** Cloud. If you host yourself, this is the "regular" **Photon** Server Application Name (most likely: "LoadBalancing").

string AppVersion

get set

The version of your client. A new version also creates a new "virtual app" to separate players from older client versions.

AuthenticationValues AuthValues

get set

User authentication values to be sent to the **Photon** server right after connecting.

Set this property or pass **AuthenticationValues** by `Connect(..., authValues)`.

string CloudRegion

get

The cloud region this client connects to. Set by **ConnectToRegionMaster()**. Not set if you don't use a NameServer!

TypedLobby CurrentLobby

get set

The lobby this client currently uses. Defined when joining a lobby or creating rooms

Room CurrentRoom

get

The current room this client is connected to (null if none available).

string CurrentServerAddress

get

The currently used server address (if any). The type of server is define by Server property.

DisconnectCause DisconnectedCause

get

protected set

Summarizes (aggregates) the different causes for disconnects of a client.

A disconnect can be caused by: errors in the network connection or some vital operation failing (which is considered "high level"). While operations always trigger a call to OnOperationResponse, connection related changes are treated in OnStatusChanged. The DisconnectCause is set in either case and summarizes the causes for any disconnect in a single state value which can be used to display (or debug) the cause for disconnection.

string GameServerAddress

get

set

The game server's address for a particular room. In use temporarily, as assigned by master.

bool InLobby

get

Internal value if the client is in a lobby.

This is used to re-set this.State, when joining/creating a room fails.

bool InRoom

get

Is true while being in a room (this.state == **ClientState.Joined**).

Aside from polling this value, game logic should implement **IMatchmakingCallbacks** in some class and react when that gets called.

OpRaiseEvent, OpLeave and some other operations can only be used (successfully) when the client is in a room. PUN's PhotonNetwork.InRoom will support being InRoom in it's offline mode (by providing it's own property).

bool IsConnected

get

Returns if this client is currently connected or connecting to some type of server.

This is even true while switching servers. Use IsConnectedAndReady to check only for those states that enable you to send Operations.

bool IsConnectedAndReady

get

A refined version of IsConnected which is true only if your connection is ready to send operations.

Not all operations can be called on all types of servers. If an operation is unavailable on the currently connected server, this will result in a OperationResponse with **ErrorCode** != 0.

Examples: The NameServer allows OpGetRegions which is not available anywhere else. The MasterServer does not allow you to send events (OpRaiseEvent) and on the GameServer you are unable to join a lobby (OpJoinLobby).

To check which server you are on, use: PhotonNetwork.Server.

bool IsFetchingFriendList

get

Internal flag to know if the client currently fetches a friend list.

bool IsUsingNameServer

get set

True if this client uses a NameServer to get the Master Server address.

This value is public, despite being an internal value, which should only be set by this client or PUN.

LoadBalancingPeer LoadBalancingPeer

get

The client uses a **LoadBalancingPeer** as API to communicate with the server. This is public for ease-of-use: Some methods like `OpRaiseEvent` are not relevant for the connection state and don't need a override.

Player LocalPlayer

get set

The local player is never null but not valid unless the client is in a room, too. The ID will be -1 outside of rooms.

string MasterServerAddress

get set

Your Master Server address. In PhotonCloud, call **ConnectToRegionMaster()** to find your Master Server.

In the **Photon** Cloud, explicit definition of a Master Server Address is not best practice. The **Photon** Cloud has a "Name Server" which redirects clients to a specific Master Server (per **Region** and AppId).

string NameServerAddress

get

Name Server Address for **Photon** Cloud (based on current protocol). You can use the default values and usually won't have to set this value.

string NickName

get set

The nickname of the player (synced with others). Same as client.LocalPlayer.NickName.

int PlayersInRoomsCount

get set

Statistic value available on master server: Players in rooms (playing).

int PlayersOnMasterCount

get set

Statistic value available on master server: Players on master (looking for games).

int RoomsCount

get set

Statistic value available on master server: Rooms currently created.

ServerConnection Server

get

The server this client is currently connected or connecting to.

Each server (NameServer, MasterServer, GameServer) allow some operations and reject others.

ClientState State

get set

Current state this client is in. Careful: several states are "transitions" that lead to other states.

bool UseAlternativeUdpPorts

get set

Use the alternative ports for UDP connections in the Public Cloud (27000 to 27003).

This should be used when players have issues with connection stability. Some players reported better connectivity for Steam games. The effect might vary, which is why the alternative ports are not the new default.

The alternative (server) ports are 27000 up to 27003.

The values are applied by replacing any incoming server-address string accordingly. You only need to set this to true though.

This value does not affect TCP or WebSocket connections.

string UserId

get set

An ID for this user. Sent in OpAuthenticate when you connect. If not set, the PlayerName is applied during connect.

On connect, if the UserId is null or empty, the client will copy the PlayName to UserId. If PlayerName is not set either (before connect), the server applies a temporary ID which stays unknown to this client and other clients.

The UserId is what's used in FindFriends and for fetching data for your account (with WebHooks e.g.).

By convention, set this ID before you connect, not while being

connected. There is no error but the ID won't change while being connected.

Event Documentation

Action<EventData> EventReceived

Register a method to be called when an event got dispatched. Gets called after the **LoadBalancingClient** handled the internal events first.

This is an alternative to extending **LoadBalancingClient** to override **OnEvent()**.

Note that OnEvent is calling EventReceived after it handled internal events first. That means for example: Joining players will already be in the player list but leaving players will already be removed from the room.

Action<OperationResponse> OpResponseReceived

Register a method to be called when an operation response is received.

This is an alternative to extending **LoadBalancingClient** to override **OnOperationResponse()**.

Note that OnOperationResponse gets executed before your Action is called. That means for example: The OpJoinLobby response already set the state to "JoinedLobby" and the response to OpLeave already triggered the Disconnect before this is called.

Action<ClientState, ClientState> StateChanged

Register a method to be called when this client's ClientState gets set.

This can be useful to react to being connected, joined into a room, etc.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	LoadBalancingPeer		

LoadBalancingPeer Class Reference

[Public Member Functions](#) |
[Protected Member Functions](#) |
[List of all members](#)

A LoadbalancingPeer provides the operations and enum definitions needed to use the loadbalancing server application which is also used in **Photon** Cloud. [More...](#)

Inherits PhotonPeer.

Public Member Functions

LoadBalancingPeer (ConnectionProtocol protocolType)

Creates a Peer with specified connection protocol. You need to set the Listener before using the peer. [More...](#)

LoadBalancingPeer (IPhotonPeerListener listener, ConnectionProtocol protocolType)

Creates a Peer with specified connection protocol and a Listener for callbacks. [More...](#)

virtual bool **OpGetRegions** (string appId)

virtual bool **OpJoinLobby** (TypedLobby lobby=null)

Joins the lobby on the Master Server, where you get a list of RoomInfos of currently open rooms. This is an async request which triggers a OnOperationResponse() call. [More...](#)

virtual bool **OpLeaveLobby** ()

Leaves the lobby on the Master Server. This is an async request which triggers a OnOperationResponse() call. [More...](#)

virtual bool **OpCreateRoom** (EnterRoomParams opParams)

Creates a room (on either Master or Game Server). The OperationResponse depends on the server the peer is connected to: Master will return a Game Server to connect to. Game Server will return the joined **Room's** data. This is an async request which triggers a OnOperationResponse() call. [More...](#)

virtual bool **OpJoinRoom** (EnterRoomParams opParams)

Joins a room by name or creates new room if room with given name not exists. The OperationResponse depends on the server the peer is connected to: Master will return a Game Server to connect to. Game Server

will return the joined **Room**'s data. This is an async request which triggers a `OnOperationResponse()` call. [More...](#)

virtual bool **OpJoinRandomRoom** (**OpJoinRandomRoomParams** opJoinRandomRoomParams)
Operation to join a random, available room. Overloads take additional player properties. This is an async request which triggers a `OnOperationResponse()` call. If all rooms are closed or full, the `OperationResponse` will have a `returnCode` of **ErrorCode.NoRandomMatchFound**. If successful, the `OperationResponse` contains a gameserver address and the name of some room. [More...](#)

virtual bool **OpLeaveRoom** (bool becomeInactive, bool sendAuthCookie=false)
Leaves a room with option to come back later or "for good". [More...](#)

virtual bool **OpGetGameList** (**TypedLobby** lobby, string queryData)
Gets a list of games matching a SQL-like where clause. [More...](#)

virtual bool **OpFindFriends** (string[] friendsToFind)
Request the rooms and online status for a list of friends (each client must set a unique username via `OpAuthenticate`). [More...](#)

bool **OpSetCustomPropertiesOfActor** (int actorNr, Hashtable actorProperties)

bool **OpSetCustomPropertiesOfRoom** (Hashtable gameProperties)

virtual bool **OpAuthenticate** (string appld, string appVersion, **AuthenticationValues** authValues, string regionCode, bool getLobbyStatistics)

Sends this app's appId and appVersion to identify this application server side. This is an async request which triggers a OnOperationResponse() call. [More...](#)

virtual bool **OpAuthenticateOnce** (string appId, string appVersion, **AuthenticationValues** authValues, string regionCode, **EncryptionMode** encryptionMode, ConnectionProtocol expectedProtocol)

Sends this app's appId and appVersion to identify this application server side. This is an async request which triggers a OnOperationResponse() call. [More...](#)

virtual bool **OpChangeGroups** (byte[] groupsToRemove, byte[] groupsToAdd)

Operation to handle this client's interest groups (for events in room). [More...](#)

virtual bool **OpRaiseEvent** (byte eventCode, object customEventContent, **RaiseEventOptions** raiseEventOptions, SendOptions sendOptions)

Send an event with custom code/type and any content to the other players in the same room. [More...](#)

virtual bool **OpSettings** (bool receiveLobbyStats)

Internally used operation to set some "per server" settings. This is for the Master Server. [More...](#)

Protected Member Functions

void **OpSetPropertyOfRoom** (byte propCode, object value)

Detailed Description

A `LoadbalancingPeer` provides the operations and enum definitions needed to use the loadbalancing server application which is also used in **Photon** Cloud.

Internally used by PUN. The **`LoadBalancingPeer`** does not keep a state, instead this is done by a **`LoadBalancingClient`**.

Constructor & Destructor Documentation

LoadBalancingPeer (ConnectionProtocol *protocolType*)

Creates a Peer with specified connection protocol. You need to set the Listener before using the peer.

Each connection protocol has it's own default networking ports for **Photon**.

Parameters

protocolType The preferred option is UDP.

LoadBalancingPeer (IPhotonPeerListener *listener*, ConnectionProtocol *protocolType*)

Creates a Peer with specified connection protocol and a Listener for callbacks.

Member Function Documentation

virtual bool

```
OpAuthenticate ( string      appld,
                  string      appVersion,
                  AuthenticationValues authValues,
                  string      regionCode,
                  bool        getLobbyStatistics
                )
```

virtual

Sends this app's **appld** and **appVersion** to identify this application server side. This is an async request which triggers a **OnOperationResponse()** call.

This operation makes use of encryption, if that is established before. See: **EstablishEncryption()**. Check encryption with **IsEncryptionAvailable**. This operation is allowed only once per connection (multiple calls will have **ErrorCode** != Ok).

Parameters

appld	Your application's name or ID to authenticate. This is assigned by Photon Cloud (webpage).
appVersion	The client's version (clients with differing client appVersions are separated and players don't meet).
authValues	Contains all values relevant for authentication. Even without account system (external Custom Auth), the clients are allowed to identify themselves.
regionCode	Optional region code, if the client should connect to a specific Photon Cloud Region .
getLobbyStatistics	Set to true on Master Server to receive

"Lobby Statistics" events.

Returns

If the operation could be sent (has to be connected).

virtual bool

```
OpAuthenticateOnce ( string          appld,
                    string          appVersion,
                    AuthenticationValues authValues,
                    string          regionCode,
                    EncryptionMode   encryptionMode,
                    ConnectionProtocol expectedProtocol
                    )
```

vi

Sends this app's appld and appVersion to identify this application server side. This is an async request which triggers a OnOperationResponse() call.

This operation makes use of encryption, if that is established before. See EstablishEncryption(). Check encryption with IsEncryptionAvailable. The operation is allowed only once per connection (multiple calls will have **ErrorCode != Ok**).

Parameters

appld	Your application's name or ID to authenticate. This is assigned by Photon Cloud (webpage).
appVersion	The client's version (clients with differing client appVersions are separated and players don't meet).
authValues	Optional authentication values. The client can set no values or a UserId or some parameters for Custom Authentication by a server.
regionCode	Optional region code, if the client should connect to a specific Photon Cloud Region .
encryptionMode	
expectedProtocol	

Returns

If the operation could be sent (has to be connected).

```
virtual bool OpChangeGroups ( byte[] groupsToRemove,  
                             byte[] groupsToAdd  
                             )
```

virtual

Operation to handle this client's interest groups (for events in room).

Note the difference between passing null and byte[0]: null won't add/remove any groups. byte[0] will add/remove all (existing) groups. First, removing groups is executed. This way, you could leave all groups and join only the ones provided.

Changes become active not immediately but when the server executes this operation (approximately RTT/2).

Parameters

groupsToRemove Groups to remove from interest. Null will not remove any. A byte[0] will remove all.

groupsToAdd Groups to add to interest. Null will not add any. A byte[0] will add all current.

Returns

If operation could be enqueued for sending. Sent when calling: Service or SendOutgoingCommands.

```
virtual bool  
OpCreateRoom ( EnterRoomParams opParams )
```

virtual

Creates a room (on either Master or Game Server). The OperationResponse depends on the server the peer is connected to: Master will return a Game Server to connect to. Game Server will return the joined **Room**'s data. This is an async request which triggers a OnOperationResponse() call.

If the room is already existing, the OperationResponse will have a

returnCode of ErrorCode.GameAlreadyExists.

virtual bool OpFindFriends (string[] friendsToFind)

virtual

Request the rooms and online status for a list of friends (each client must set a unique username via OpAuthenticate).

Used on Master Server to find the rooms played by a selected list of users. Users identify themselves by using OpAuthenticate with a unique username. The list of usernames must be fetched from some other source (not provided by **Photon**).

The server response includes 2 arrays of info (each index matching a friend from the request):

ParameterCode.FindFriendsResponseOnlineList = bool[] of online states
ParameterCode.FindFriendsResponseRoomIdList = string[] of room names (empty string if not in a room)

Parameters

friendsToFind Array of friend's names (make sure they are unique).

Returns

If the operation could be sent (requires connection).

virtual bool OpGetGameList (TypedLobby lobby, string queryData)

virtual

Gets a list of games matching a SQL-like where clause.

Operation is only available in lobbies of type SqlLobby. This is an async request which triggers a OnOperationResponse() call. Returned game list is stored in RoomInfoList.

<https://doc.photonengine.com/en-us/realtime/current/reference/matchmaking-and->

lobby::sql_lobby_type

Parameters

lobby The lobby to query. Has to be of type SqlLobby.
queryData The sql query statement.

Returns

If the operation could be sent (has to be connected).

virtual bool OpJoinLobby (TypedLobby lobby = null)

virtual

Joins the lobby on the Master Server, where you get a list of RoomInfos of currently open rooms. This is an async request which triggers a OnOperationResponse() call.

Parameters

lobby The lobby join to.

Returns

If the operation could be sent (has to be connected).

virtual bool

OpJoinRandomRoom (OpJoinRandomRoomParams opJoinRand

Operation to join a random, available room. Overloads take additional p an async request which triggers a OnOperationResponse() call. If all ro the OperationResponse will have a returnCode of **ErrorCode.NoRand** successful, the OperationResponse contains a gameserver address an room.

Returns

If the operation could be sent currently (requires connection).

virtual bool OpJoinRoom (EnterRoomParams opParams)

virtual

Joins a room by name or creates new room if room with given name not exists. The OperationResponse depends on the server the peer is connected to: Master will return a Game Server to connect to. Game Server will return the joined **Room**'s data. This is an async request which triggers a OnOperationResponse() call.

If the room is not existing (anymore), the OperationResponse will have a returnCode of **ErrorCode.GameDoesNotExist**. Other possible ErrorCodes are: GameClosed, GameFull.

Returns

If the operation could be sent (requires connection).

virtual bool OpLeaveLobby ()

virtual

Leaves the lobby on the Master Server. This is an async request which triggers a OnOperationResponse() call.

Returns

If the operation could be sent (requires connection).

virtual bool OpLeaveRoom (bool **becomeInactive**, bool **sendAuthCookie** = false)

virtual

Leaves a room with option to come back later or "for good".

Parameters

becomeInactive Async games can be re-joined (loaded) later on. Set to false, if you want to abandon a game entirely.

sendAuthCookie WebFlag: Securely transmit the encrypted object AuthCookie to the web service in PathLeave webhook when available

Returns

If the operation can be send currently.

virtual bool

OpRaiseEvent (byte
 object
 RaiseEventOptions
 SendOptions
) **eventCode,**
 customEventContent,
 raiseEventOptions,
 sendOptions

virtual

Send an event with custom code/type and any content to the other players in the same room.

This override explicitly uses another parameter order to not mix it up with the implementation for Hashtable only.

Parameters

eventCode	Identifies this type of event (and the content). Your game's event codes can start with 0.
customEventContent	Any serializable datatype (including Hashtable like the other OpRaiseEvent overloads).
raiseEventOptions	Contains (slightly) less often used options. If you pass null, the default options will be used.
sendOptions	Send options for reliable, encryption etc

Returns

If operation could be enqueued for sending. Sent when calling: Service or SendOutgoingCommands.

virtual bool OpSettings (bool **receiveLobbyStats**)

virtual

Internally used operation to set some "per server" settings. This is for the Master Server.

Parameters

receiveLobbyStats Set to true, to get Lobby Statistics (lists of existing lobbies).

Returns

False if the operation could not be sent.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	MatchMakingCallbacksContainer		

[Public Member Functions](#) | [List of all members](#)

MatchMakingCallbacksContainer Class Reference

Container type for callbacks defined by **IMatchmakingCallbacks**. See **MatchMakingCallbackTargets**. [More...](#)

Inherits `List< IMatchmakingCallbacks >`, and **IMatchmakingCallbacks**.

Public Member Functions

void **AddCallbackTarget** (IMatchmakingCallbacks target)

void **RemoveCallbackTarget** (IMatchmakingCallbacks target)

void **OnCreatedRoom** ()
Called when this client created a room and entered it.
OnJoinedRoom() will be called as well. [More...](#)

void **OnJoinedRoom** ()
Called when the **LoadBalancingClient** entered a room, no matter if this client created it or simply joined. [More...](#)

void **OnCreateRoomFailed** (short returnCode, string message)
Called when the server couldn't create a room (OpCreateRoom failed). [More...](#)

void **OnJoinRandomFailed** (short returnCode, string message)
Called when a previous OpJoinRandom call failed on the server. [More...](#)

void **OnJoinRoomFailed** (short returnCode, string message)
Called when a previous OpJoinRoom call failed on the server.
[More...](#)

void **OnLeftRoom** ()
Called when the local user/client left a room, so the game's logic can clean up it's internal state. [More...](#)

void **OnFriendListUpdate** (List< **FriendInfo** > friendList)
Called when the server sent the response to a FindFriends request. [More...](#)

Detailed Description

Container type for callbacks defined by **IMatchmakingCallbacks**. See **MatchMakingCallbackTargets**.

While the interfaces of callbacks wrap up the methods that will be called, the container classes implement a simple way to call a method on all registered objects.

Member Function Documentation

void OnCreatedRoom ()

Called when this client created a room and entered it. **OnJoinedRoom()** will be called as well.

This callback is only called on the client which created a room (see `OpCreateRoom`).

As any client might close (or drop connection) anytime, there is a chance that the creator of a room does not execute `OnCreatedRoom`.

If you need specific room properties or a "start signal", implement `OnMasterClientSwitched()` and make each new `MasterClient` check the room's state.

Implements **IMatchmakingCallbacks**.

void OnCreateRoomFailed (short `returnCode`, string `message`)

Called when the server couldn't create a room (`OpCreateRoom` failed).

Creating a room may fail for various reasons. Most often, the room already exists (roomname in use) or the **RoomOptions** clash and it's impossible to create the room.

When creating a room fails on a Game Server: The client will cache the failure internally and returns to the Master Server before it calls the fail-callback. This way, the client is ready to find/create a room at the moment of the callback. In this case, the client skips calling

OnConnectedToMaster but returning to the Master Server will still call OnConnected. Treat callbacks of OnConnected as pure information that the client could connect.

Parameters

returnCode Operation ReturnCode from the server.

message Debug message for the error.

Implements **IMatchmakingCallbacks**.

void OnFriendListUpdate (List< FriendInfo > friendList)

Called when the server sent the response to a FindFriends request.

After calling OpFindFriends, the Master Server will cache the friend list and send updates to the friend list. The friends includes the name, userId, online state and the room (if any) for each requested user/friend.

Use the friendList to update your UI and store it, if the UI should highlight changes.

Implements **IMatchmakingCallbacks**.

void OnJoinedRoom ()

Called when the **LoadBalancingClient** entered a room, no matter if this client created it or simply joined.

When this is called, you can access the existing players in **Room.Players**, their custom properties and **Room.CustomProperties**.

In this callback, you could create player objects. For example in Unity, instantiate a prefab for the player.

If you want a match to be started "actively", enable the user to signal

"ready" (using OpRaiseEvent or a Custom Property).

Implements **IMatchmakingCallbacks**.

```
void OnJoinRandomFailed ( short returnCode,  
                        string message  
                        )
```

Called when a previous OpJoinRandom call failed on the server.

The most common causes are that a room is full or does not exist (due to someone else being faster or closing the room).

This operation is only ever sent to the Master Server. Once a room is found by the Master Server, the client will head off to the designated Game Server and use the operation Join on the Game Server.

When using multiple lobbies (via OpJoinLobby or a **TypedLobby** parameter), another lobby might have more/fitting rooms.

Parameters

returnCode Operation ReturnCode from the server.

message Debug message for the error.

Implements **IMatchmakingCallbacks**.

```
void OnJoinRoomFailed ( short returnCode,  
                      string message  
                      )
```

Called when a previous OpJoinRoom call failed on the server.

Joining a room may fail for various reasons. Most often, the room is full or does not exist anymore (due to someone else being faster or closing the room).

When joining a room fails on a Game Server: The client will cache the failure internally and returns to the Master Server before it calls the fail-callback. This way, the client is ready to find/create a room at the moment of the callback. In this case, the client skips calling `OnConnectedToMaster` but returning to the Master Server will still call `OnConnected`. Treat callbacks of `OnConnected` as pure information that the client could connect.

Parameters

returnCode Operation ReturnCode from the server.

message Debug message for the error.

Implements **IMatchmakingCallbacks**.

void OnLeftRoom ()

Called when the local user/client left a room, so the game's logic can clean up it's internal state.

When leaving a room, the **LoadBalancingClient** will disconnect the Game Server and connect to the Master Server. This wraps up multiple internal actions.

Wait for the callback `OnConnectedToMaster`, before you use lobbies and join or create rooms.

Implements **IMatchmakingCallbacks**.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	OperationCode		

[Public Attributes](#) | [List of all members](#)

OperationCode Class Reference

Class for constants. Contains operation codes. **Pun** uses these constants internally. [More...](#)

Public Attributes

const byte **ExchangeKeysForEncryption** = 250

const byte **Join** = 255
(255) Code for OpJoin, to get into a room. [More...](#)

const byte **AuthenticateOnce** = 231
(231) Authenticates this peer and connects to a virtual application [More...](#)

const byte **Authenticate** = 230
(230) Authenticates this peer and connects to a virtual application [More...](#)

const byte **JoinLobby** = 229
(229) Joins lobby (on master) [More...](#)

const byte **LeaveLobby** = 228
(228) Leaves lobby (on master) [More...](#)

const byte **CreateGame** = 227
(227) Creates a game (or fails if name exists) [More...](#)

const byte **JoinGame** = 226
(226) Join game (by name) [More...](#)

const byte **JoinRandomGame** = 225
(225) Joins random game (on master) [More...](#)

const byte **Leave** = (byte)254
(254) Code for OpLeave, to get out of a room. [More...](#)

const byte **RaiseEvent** = (byte)253
(253) Raise event (in a room, for other actors/players) [More...](#)

const byte **SetProperties** = (byte)252
(252) Set Properties (of room or actor/player) [More...](#)

const byte **GetProperties** = (byte)251
(251) Get Properties [More...](#)

const byte **ChangeGroups** = (byte)248
(248) Operation code to change interest groups in Rooms (Lite application and extending ones). [More...](#)

const byte **FindFriends** = 222
(222) Request the rooms and online status for a list of friends (by name, which should be unique). [More...](#)

const byte **GetLobbyStats** = 221
(221) Request statistics about a specific list of lobbies (their user and game count). [More...](#)

const byte **GetRegions** = 220
(220) Get list of regional servers from a NameServer. [More...](#)

const byte **WebRpc** = 219
(219) WebRpc Operation. [More...](#)

const byte **ServerSettings** = 218
(218) Operation to set some server settings. Used with different parameters on various servers. [More...](#)

const byte **GetGameList** = 217
(217) Get the game list matching a supplied sql filter (SqlListLobby only) [More...](#)

Detailed Description

Class for constants. Contains operation codes. **Pun** uses these constants internally.

Member Data Documentation

const byte Authenticate = 230

(230) Authenticates this peer and connects to a virtual application

const byte AuthenticateOnce = 231

(231) Authenticates this peer and connects to a virtual application

const byte ChangeGroups = (byte)248

(248) Operation code to change interest groups in Rooms (Lite application and extending ones).

const byte CreateGame = 227

(227) Creates a game (or fails if name exists)

const byte FindFriends = 222

(222) Request the rooms and online status for a list of friends (by name, which should be unique).

const byte GetGameList = 217

(217) Get the game list matching a supplied sql filter (SqlListLobby only)

const byte GetLobbyStats = 221

(221) Request statistics about a specific list of lobbies (their user and game count).

const byte GetProperties = (byte)251

(251) Get Properties

const byte GetRegions = 220

(220) Get list of regional servers from a NameServer.

const byte Join = 255

(255) Code for OpJoin, to get into a room.

const byte JoinGame = 226

(226) Join game (by name)

const byte JoinLobby = 229

(229) Joins lobby (on master)

const byte JoinRandomGame = 225

(225) Joins random game (on master)

const byte Leave = (byte)254

(254) Code for OpLeave, to get out of a room.

const byte LeaveLobby = 228

(228) Leaves lobby (on master)

const byte RaiseEvent = (byte)253

(253) Raise event (in a room, for other actors/players)

const byte ServerSettings = 218

(218) Operation to set some server settings. Used with different parameters on various servers.

const byte SetPropertyes = (byte)252

(252) Set Properties (of room or actor/player)

const byte WebRpc = 219

(219) WebRpc Operation.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	OpJoinRandomRoomParams		

[Public Attributes](#) | [List of all members](#)

OpJoinRandomRoomParams Class Reference

Public Attributes

Hashtable	ExpectedCustomRoomProperties
byte	ExpectedMaxPlayers
MatchmakingMode	MatchingType
TypedLobby	TypedLobby
string	SqlLobbyFilter
string[]	ExpectedUsers



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	ParameterCode		

[Public Attributes](#) | [List of all members](#)

ParameterCode Class Reference

Class for constants. Codes for parameters of Operations and Events.
[More...](#)

Public Attributes

const byte **SuppressRoomEvents** = 237

(237) A bool parameter for creating games. If set to true, no room events are sent to the clients on join and leave. Default: false (and not sent). [More...](#)

const byte **EmptyRoomTTL** = 236

(236) Time To Live (TTL) for a room when the last player leaves. Keeps room in memory for case a player re-joins soon. In milliseconds. [More...](#)

const byte **PlayerTTL** = 235

(235) Time To Live (TTL) for an 'actor' in a room. If a client disconnects, this actor is inactive first and removed after this timeout. In milliseconds. [More...](#)

const byte **EventForward** = 234

(234) Optional parameter of OpRaiseEvent and OpSetCustomProperties to forward the event/operation to a web-service. [More...](#)

const byte **IsComingBack** = (byte)233

(233) Optional parameter of OpLeave in async games. If false, the player does abandons the game (forever). By default players become inactive and can re-join. [More...](#)

const byte **IsInactive** = (byte)233

(233) Used in EvLeave to describe if a user is inactive (and might come back) or not. In rooms with PlayerTTL, becoming inactive is the default case. [More...](#)

const byte **CheckUserOnJoin** = (byte)232

(232) Used when creating rooms to define if any userid can join the room only once. [More...](#)

const byte **ExpectedValues** = (byte)231
(231) Code for "Check And Swap" (CAS) when changing properties. [More...](#)

const byte **Address** = 230
(230) Address of a (game) server to use. [More...](#)

const byte **PeerCount** = 229
(229) Count of players in this application in a rooms (used in stats event) [More...](#)

const byte **GameCount** = 228
(228) Count of games in this application (used in stats event) [More...](#)

const byte **MasterPeerCount** = 227
(227) Count of players on the master server (in this app, looking for rooms) [More...](#)

const byte **UserId** = 225
(225) User's ID [More...](#)

const byte **ApplicationId** = 224
(224) Your application's ID: a name on your own **Photon** or a GUID on the **Photon** Cloud [More...](#)

const byte **Position** = 223
(223) Not used currently (as "Position"). If you get queued before connect, this is your position [More...](#)

const byte **MatchMakingType** = 223
(223) Modifies the matchmaking algorithm used for OpJoinRandom. Allowed parameter values are defined in enum MatchmakingMode. [More...](#)

const byte **GameList** = 222
(222) List of RoomInfos about open / listed rooms [More...](#)

const byte **Secret** = 221
(221) Internally used to establish encryption [More...](#)

const byte **AppVersion** = 220
(220) Version of your application [More...](#)

const byte **AzureNodeInfo** = 210
(210) Internally used in case of hosting by Azure [More...](#)

const byte **AzureLocalNodeId** = 209
(209) Internally used in case of hosting by Azure [More...](#)

const byte **AzureMasterNodeId** = 208
(208) Internally used in case of hosting by Azure [More...](#)

const byte **RoomName** = (byte)255
(255) Code for the gameId/roomName (a unique name per room). Used in OpJoin and similar. [More...](#)

const byte **Broadcast** = (byte)250
(250) Code for broadcast parameter of OpSetProperties method. [More...](#)

const byte **ActorList** = (byte)252
(252) Code for list of players in a room. Currently not used. [More...](#)

const byte **ActorNr** = (byte)254
(254) Code of the Actor of an operation. Used for property get and set. [More...](#)

const byte **PlayerProperties** = (byte)249
(249) Code for property set (Hashtable). [More...](#)

const byte **CustomEventContent** = (byte)245

(245) Code of data/custom content of an event. Used in OpRaiseEvent. [More...](#)

const byte **Data** = (byte)245
(245) Code of data of an event. Used in OpRaiseEvent. [More...](#)

const byte **Code** = (byte)244
(244) Code used when sending some code-related parameter, like OpRaiseEvent's event-code. [More...](#)

const byte **GameProperties** = (byte)248
(248) Code for property set (Hashtable). [More...](#)

const byte **Properties** = (byte)251
(251) Code for property-set (Hashtable). This key is used when sending only one set of properties. If either **ActorProperties** or GameProperties are used (or both), check those keys. [More...](#)

const byte **TargetActorNr** = (byte)253
(253) Code of the target Actor of an operation. Used for property set. Is 0 for game [More...](#)

const byte **ReceiverGroup** = (byte)246
(246) Code to select the receivers of events (used in Lite, Operation RaiseEvent). [More...](#)

const byte **Cache** = (byte)247
(247) Code for caching events while raising them. [More...](#)

const byte **CleanupCacheOnLeave** = (byte)241
(241) Bool parameter of CreateGame Operation. If true, server cleans up roomcache of leaving players (their cached events get removed). [More...](#)

const byte **Group** = 240

(240) Code for "group" operation-parameter (as used in Op RaiseEvent). [More...](#)

const byte **Remove** = 239

(239) The "Remove" operation-parameter can be used to remove something from a list. E.g. remove groups from player's interest groups. [More...](#)

const byte **PublishUserId** = 239

(239) Used in Op Join to define if UserIds of the players are broadcast in the room. Useful for FindFriends and reserving slots for expected users. [More...](#)

const byte **Add** = 238

(238) The "Add" operation-parameter can be used to add something to some list or set. E.g. add groups to player's interest groups. [More...](#)

const byte **Info** = 218

(218) Content for **EventCode.ErrorInfo** and internal debug operations. [More...](#)

const byte **ClientAuthenticationType** = 217

(217) This key's (byte) value defines the target custom authentication type/service the client connects with. Used in OpAuthenticate [More...](#)

const byte **ClientAuthenticationParams** = 216

(216) This key's (string) value provides parameters sent to the custom authentication type/service the client connects with. Used in OpAuthenticate [More...](#)

const byte **JoinMode** = 215

(215) Makes the server create a room if it doesn't exist. OpJoin uses this to always enter a room, unless it exists and is full/closed. [More...](#)

const byte **ClientAuthenticationData** = 214
(214) This key's (string or byte[]) value provides parameters sent to the custom authentication service setup in **Photon** Dashboard. Used in OpAuthenticate [More...](#)

const byte **MasterClientId** = (byte)203
(203) Code for MasterClientId, which is synced by server. When sent as op-parameter this is code 203. [More...](#)

const byte **FindFriendsRequestList** = (byte)1
(1) Used in Op FindFriends request. Value must be string[] of friends to look up. [More...](#)

const byte **FindFriendsResponseOnlineList** = (byte)1
(1) Used in Op FindFriends response. Contains bool[] list of online states (false if not online). [More...](#)

const byte **FindFriendsResponseRoomIdList** = (byte)2
(2) Used in Op FindFriends response. Contains string[] of room names ("" where not known or no room joined). [More...](#)

const byte **LobbyName** = (byte)213
(213) Used in matchmaking-related methods and when creating a room to name a lobby (to join or to attach a room to). [More...](#)

const byte **LobbyType** = (byte)212
(212) Used in matchmaking-related methods and when creating a room to define the type of a lobby. Combined with the lobby name this identifies the lobby. [More...](#)

const byte **LobbyStats** = (byte)211
(211) This (optional) parameter can be sent in Op Authenticate to turn on Lobby Stats (info about lobby names and their user- and game-counts). See: PhotonNetwork.Lobbies [More...](#)

const byte **Region** = (byte)210
(210) Used for region values in OpAuth and OpGetRegions. [More...](#)

const byte **UriPath** = 209
(209) Path of the WebRPC that got called. Also known as "WebRpc Name". Type: string. [More...](#)

const byte **WebRpcParameters** = 208
(208) Parameters for a WebRPC as: Dictionary<string, object>. This will get serialized to JSon. [More...](#)

const byte **WebRpcReturnCode** = 207
(207) ReturnCode for the WebRPC, as sent by the web service (not by **Photon**, which uses **ErrorCode**). Type: byte. [More...](#)

const byte **WebRpcReturnMessage** = 206
(206) Message returned by WebRPC server. Analog to **Photon**'s debug message. Type: string. [More...](#)

const byte **CacheSliceIndex** = 205
(205) Used to define a "slice" for cached events. Slices can easily be removed from cache. Type: int. [More...](#)

const byte **Plugins** = 204
(204) Informs the server of the expected plugin setup. [More...](#)

const byte **NickName** = 202
(202) Used by the server in Operation Responses, when it sends the nickname of the client (the user's nickname). [More...](#)

const byte **PluginName** = 201
(201) Informs user about name of plugin load to game

[More...](#)

const byte **PluginVersion** = 200
(200) Informs user about version of plugin load to game
[More...](#)

const byte **ExpectedProtocol** = 195
(195) Protocol which will be used by client to connect master/game servers. Used for nameserver. [More...](#)

const byte **CustomInitData** = 194
(194) Set of custom parameters which are sent in auth request. [More...](#)

const byte **EncryptionMode** = 193
(193) How are we going to encrypt data. [More...](#)

const byte **EncryptionData** = 192
(192) Parameter of Authentication, which contains encryption keys (depends on AuthMode and EncryptionMode). [More...](#)

const byte **RoomOptionFlags** = 191
(191) An int parameter summarizing several boolean room-options with bit-flags. [More...](#)

Detailed Description

Class for constants. Codes for parameters of Operations and Events.

Pun uses these constants internally.

Member Data Documentation

const byte ActorList = (byte)252

(252) Code for list of players in a room. Currently not used.

const byte ActorNr = (byte)254

(254) Code of the Actor of an operation. Used for property get and set.

const byte Add = 238

(238) The "Add" operation-parameter can be used to add something to some list or set. E.g. add groups to player's interest groups.

const byte Address = 230

(230) Address of a (game) server to use.

const byte ApplicationId = 224

(224) Your application's ID: a name on your own **Photon** or a GUID on the **Photon** Cloud

const byte AppVersion = 220

(220) Version of your application

const byte AzureLocalNodeId = 209

(209) Internally used in case of hosting by Azure

const byte AzureMasterNodeId = 208

(208) Internally used in case of hosting by Azure

const byte AzureNodeInfo = 210

(210) Internally used in case of hosting by Azure

const byte Broadcast = (byte)250

(250) Code for broadcast parameter of OpSetProperties method.

const byte Cache = (byte)247

(247) Code for caching events while raising them.

const byte CacheSliceIndex = 205

(205) Used to define a "slice" for cached events. Slices can easily be removed from cache. Type: int.

const byte CheckUserOnJoin = (byte)232

(232) Used when creating rooms to define if any userid can join the room only once.

const byte CleanupCacheOnLeave = (byte)241

(241) Bool parameter of CreateGame Operation. If true, server cleans up roomcache of leaving players (their cached events get removed).

const byte ClientAuthenticationData = 214

(214) This key's (string or byte[]) value provides parameters sent to the custom authentication service setup in **Photon** Dashboard. Used in OpAuthenticate

const byte ClientAuthenticationParams = 216

(216) This key's (string) value provides parameters sent to the custom authentication type/service the client connects with. Used in OpAuthenticate

const byte ClientAuthenticationType = 217

(217) This key's (byte) value defines the target custom authentication type/service the client connects with. Used in OpAuthenticate

const byte Code = (byte)244

(244) Code used when sending some code-related parameter, like OpRaiseEvent's event-code.

This is not the same as the Operation's code, which is no longer sent as part of the parameter Dictionary in **Photon 3**.

const byte CustomEventContent = (byte)245

(245) Code of data/custom content of an event. Used in OpRaiseEvent.

const byte CustomInitData = 194

(194) Set of custom parameters which are sent in auth request.

const byte Data = (byte)245

(245) Code of data of an event. Used in OpRaiseEvent.

const byte EmptyRoomTTL = 236

(236) Time To Live (TTL) for a room when the last player leaves. Keeps room in memory for case a player re-joins soon. In milliseconds.

const byte EncryptionData = 192

(192) Parameter of Authentication, which contains encryption keys (depends on AuthMode and EncryptionMode).

const byte EncryptionMode = 193

(193) How are we going to encrypt data.

const byte EventForward = 234

(234) Optional parameter of OpRaiseEvent and OpSetCustomProperties to forward the event/operation to a web-service.

const byte ExpectedProtocol = 195

(195) Protocol which will be used by client to connect master/game servers. Used for nameserver.

const byte ExpectedValues = (byte)231

(231) Code for "Check And Swap" (CAS) when changing properties.

const byte FindFriendsRequestList = (byte)1

(1) Used in Op FindFriends request. Value must be string[] of friends to look up.

const byte FindFriendsResponseOnlineList = (byte)1

(1) Used in Op FindFriends response. Contains bool[] list of online states (false if not online).

const byte FindFriendsResponseRoomIdList = (byte)2

(2) Used in Op FindFriends response. Contains string[] of room names ("" where not known or no room joined).

const byte GameCount = 228

(228) Count of games in this application (used in stats event)

const byte GameList = 222

(222) List of RoomInfos about open / listed rooms

const byte GameProperties = (byte)248

(248) Code for property set (Hashtable).

const byte Group = 240

(240) Code for "group" operation-parameter (as used in OpRaiseEvent).

const byte Info = 218

(218) Content for **EventCode.ErrorInfo** and internal debug operations.

const byte IsComingBack = (byte)233

(233) Optional parameter of OpLeave in async games. If false, the player does abandons the game (forever). By default players become inactive and can re-join.

const byte IsInactive = (byte)233

(233) Used in EvLeave to describe if a user is inactive (and might come back) or not. In rooms with PlayerTTL, becoming inactive is

the default case.

const byte JoinMode = 215

(215) Makes the server create a room if it doesn't exist. OpJoin uses this to always enter a room, unless it exists and is full/closed.

(215) The JoinMode enum defines which variant of joining a room will be executed: Join only if available, create if not exists or re-join.

Replaces CreateIfNotExists which was only a bool-value.

const byte LobbyName = (byte)213

(213) Used in matchmaking-related methods and when creating a room to name a lobby (to join or to attach a room to).

const byte LobbyStats = (byte)211

(211) This (optional) parameter can be sent in Op Authenticate to turn on Lobby Stats (info about lobby names and their user- and game-counts). See: PhotonNetwork.Lobbies

const byte LobbyType = (byte)212

(212) Used in matchmaking-related methods and when creating a room to define the type of a lobby. Combined with the lobby name this identifies the lobby.

const byte MasterClientId = (byte)203

(203) Code for MasterClientId, which is synced by server. When sent as op-parameter this is code 203.

Tightly related to **GamePropertyKey.MasterClientId**.

const byte MasterPeerCount = 227

(227) Count of players on the master server (in this app, looking for rooms)

const byte MatchMakingType = 223

(223) Modifies the matchmaking algorithm used for OpJoinRandom. Allowed parameter values are defined in enum MatchmakingMode.

const byte NickName = 202

(202) Used by the server in Operation Responses, when it sends the nickname of the client (the user's nickname).

const byte PeerCount = 229

(229) Count of players in this application in a rooms (used in stats event)

const byte PlayerProperties = (byte)249

(249) Code for property set (Hashtable).

const byte PlayerTTL = 235

(235) Time To Live (TTL) for an 'actor' in a room. If a client disconnects, this actor is inactive first and removed after this timeout. In milliseconds.

const byte PluginName = 201

(201) Informs user about name of plugin load to game

const byte Plugins = 204

(204) Informs the server of the expected plugin setup.

The operation will fail in case of a plugin mismatch returning error code PluginMismatch 32751(0x7FFF - 16). Setting string[]{} means the client expects no plugin to be setup. Note: for backwards compatibility null omits any check.

const byte PluginVersion = 200

(200) Informs user about version of plugin load to game

const byte Position = 223

(223) Not used currently (as "Position"). If you get queued before connect, this is your position

const byte Properties = (byte)251

(251) Code for property-set (Hashtable). This key is used when sending only one set of properties. If either **ActorProperties** or **GameProperties** are used (or both), check those keys.

const byte PublishUserId = 239

(239) Used in Op Join to define if UserIds of the players are

broadcast in the room. Useful for FindFriends and reserving slots for expected users.

const byte ReceiverGroup = (byte)246

(246) Code to select the receivers of events (used in Lite, Operation RaiseEvent).

const byte Region = (byte)210

(210) Used for region values in OpAuth and OpGetRegions.

const byte Remove = 239

(239) The "Remove" operation-parameter can be used to remove something from a list. E.g. remove groups from player's interest groups.

const byte RoomName = (byte)255

(255) Code for the gameId/roomName (a unique name per room). Used in OpJoin and similar.

const byte RoomOptionFlags = 191

(191) An int parameter summarizing several boolean room-options with bit-flags.

const byte Secret = 221

(221) Internally used to establish encryption

const byte SuppressRoomEvents = 237

(237) A bool parameter for creating games. If set to true, no room events are sent to the clients on join and leave. Default: false (and not sent).

const byte TargetActorNr = (byte)253

(253) Code of the target Actor of an operation. Used for property set. Is 0 for game

const byte UriPath = 209

(209) Path of the WebRPC that got called. Also known as "WebRpc Name". Type: string.

const byte UserId = 225

(225) User's ID

const byte WebRpcParameters = 208

(208) Parameters for a WebRPC as: Dictionary<string, object>. This will get serialized to JSon.

const byte WebRpcReturnCode = 207

(207) ReturnCode for the WebRPC, as sent by the web service (not by **Photon**, which uses **ErrorCode**). Type: byte.

const byte WebRpcReturnMessage = 206

(206) Message returned by WebRPC server. Analog to **Photon's** debug message. Type: string.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	PhotonPing		

[Public Member Functions](#) | [Public Attributes](#) |
[List of all members](#)

PhotonPing Class Reference

Inherits IDisposable.

Inherited by **PingMono**.

Public Member Functions

virtual bool **StartPing** (string ip)

virtual bool **Done** ()

virtual void **Dispose** ()

Public Attributes

```
string DebugString = ""
```

```
bool Successful
```

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	PingMono		

[Public Member Functions](#) | [List of all members](#)

PingMono Class Reference

Uses C# Socket class from System.Net.Sockets (as Unity usually does). [More...](#)

Inherits **PhotonPing**.

Public Member Functions

override bool **StartPing** (string ip)
Sends a "Photon Ping" to a server. [More...](#)

override bool **Done** ()

override void **Dispose** ()

Additional Inherited Members

‣ **Public Attributes inherited from PhotonPing**

```
string DebugString = ""
```

```
bool Successful
```

Detailed Description

Uses C# Socket class from System.Net.Sockets (as Unity usually does).

Incompatible with Windows 8 Store/Phone API.

Member Function Documentation

override bool StartPing (string **ip)**

virtual

Sends a "Photon Ping" to a server.

Parameters

ip Address in IPv4 or IPv6 format. An address containing a '.' will be interpreted as IPv4.

Returns

True if the **Photon** Ping could be sent.

Reimplemented from **PhotonPing**.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	Player		

Player Class Reference

[Public Member Functions](#) | [Public Attributes](#) |
[Properties](#) | [List of all members](#)

Summarizes a "player" within a room, identified (in that room) by ID (or "actorNumber"). [More...](#)

Public Member Functions

Player **Get** (int id)
Get a **Player** by ActorNumber (Player.ID). [More...](#)

Player **GetNext** ()
Gets this **Player**'s next **Player**, as sorted by ActorNumber (Player.ID). Wraps around. [More...](#)

Player **GetNextFor** (**Player** currentPlayer)
Gets a **Player**'s next **Player**, as sorted by ActorNumber (Player.ID). Wraps around. [More...](#)

Player **GetNextFor** (int currentPlayerId)
Gets a **Player**'s next **Player**, as sorted by ActorNumber (Player.ID). Wraps around. [More...](#)

virtual void **InternalCacheProperties** (Hashtable properties)
Caches properties for new Players or when updates of remote players are received. Use **SetCustomProperties()** for a synced update. [More...](#)

override string **ToString** ()
Brief summary string of the **Player**. Includes name or player.ID and if it's the Master Client. [More...](#)

string **ToStringFull** ()
String summary of the **Player**: player.ID, name and all custom properties of this user. [More...](#)

override bool **Equals** (object p)
If players are equal (by GetHashCode, which returns this.ID). [More...](#)

override int **GetHashCode** ()
Accompanies Equals, using the ID (actorNumber) as

HashCode to return. [More...](#)

void **SetCustomProperties** (Hashtable propertiesToSet, Hashtable expectedValues=null, **WebFlags** webFlags=null)

Updates and synchronizes this **Player**'s Custom Properties. Optionally, expectedProperties can be provided as condition. [More...](#)

Public Attributes

readonly bool **IsLocal**

Only one player is controlled by each client. Others are not local. [More...](#)

object **TagObject**

Can be used to store a reference that's useful to know "by player". [More...](#)

Properties

int **ActorNumber** [get]

Identifier of this player in current room. Also known as: actorNumber or actorNumber. It's -1 outside of rooms. [More...](#)

string **NickName** [get, set]

Non-unique nickname of this player. Synced automatically in a room. [More...](#)

string **UserId** [get, set]

UserId of the player, available when the room got created with **RoomOptions.PublishUserId** = true. [More...](#)

bool **IsMasterClient** [get]

True if this player is the Master Client of the current room. [More...](#)

bool **IsInactive** [get, set]

If this player is active in the room (and getting events which are currently being sent). [More...](#)

Hashtable **CustomProperties** [get, set]

Read-only cache for custom properties of player. Set via **Player.SetCustomProperties**. [More...](#)

Detailed Description

Summarizes a "player" within a room, identified (in that room) by ID (or "actorNumber").

Each player has a actorNumber, valid for that room. It's -1 until assigned by server (and client logic).

Member Function Documentation

override bool Equals (object **p)**

If players are equal (by GetHashCode, which returns this.ID).

Player Get (int **id)**

Get a **Player** by ActorNumber (Player.ID).

Parameters

id ActorNumber of the a player in this room.

Returns

Player or null.

override int GetHashCode ()

Accompanies Equals, using the ID (actorNumber) as GetHashCode to return.

Player GetNext ()

Gets this **Player**'s next **Player**, as sorted by ActorNumber (Player.ID). Wraps around.

Returns

Player or null.

Player GetNextFor (Player **currentPlayer)**

Gets a **Player**'s next **Player**, as sorted by ActorNumber (Player.ID). Wraps around.

Useful when you pass something to the next player. For example: passing the turn to the next player.

Parameters

currentPlayer The **Player** for which the next is being needed.

Returns

Player or null.

Player GetNextFor (int **currentPlayerId**)

Gets a **Player**'s next **Player**, as sorted by ActorNumber (Player.ID). Wraps around.

Useful when you pass something to the next player. For example: passing the turn to the next player.

Parameters

currentPlayerId The ActorNumber (Player.ID) for which the next is being needed.

Returns

Player or null.

virtual void InternalCacheProperties (Hashtable **properties**) virtual

Caches properties for new Players or when updates of remote players are received. Use **SetCustomProperties()** for a synced update.

This only updates the CustomProperties and doesn't send them to the server. Mostly used when creating new remote players, where

the server sends their properties.

```
void SetCustomProperties ( Hashtable propertiesToSet,  
                          Hashtable expectedValues = null,  
                          WebFlags webFlags = null  
                          )
```

Updates and synchronizes this **Player's** Custom Properties. Optionally, expectedProperties can be provided as condition.

Custom Properties are a set of string keys and arbitrary values which is synchronized for the players in a **Room**. They are available when the client enters the room, as they are in the response of OpJoin and OpCreate.

Custom Properties either relate to the (current) **Room** or a **Player** (in that **Room**).

Both classes locally cache the current key/values and make them available as property: CustomProperties. This is provided only to read them. You must use the method SetCustomProperties to set/modify them.

Any client can set any Custom Properties anytime (when in a room). It's up to the game logic to organize how they are best used.

You should call SetCustomProperties only with key/values that are new or changed. This reduces traffic and performance.

Unless you define some expectedProperties, setting key/values is always permitted. In this case, the property-setting client will not receive the new values from the server but instead update its local cache in SetCustomProperties.

If you define expectedProperties, the server will skip updates if the server property-cache does not contain all expectedProperties with the same values. In this case, the property-setting client will get an update from the server and update it's cached key/values at about the same time as everyone else.

The benefit of using `expectedProperties` can be only one client successfully sets a key from one known value to another. As example: Store who owns an item in a Custom Property `"ownedBy"`. It's 0 initially. When multiple players reach the item, they all attempt to change `"ownedBy"` from 0 to their `actorNumber`. If you use `expectedProperties {"ownedBy", 0}` as condition, the first player to take the item will have it (and the others fail to set the ownership).

Properties get saved with the game state for Turnbased games (which use `IsPersistent = true`).

Parameters

propertiesToSet Hashtable of Custom Properties to be set.

expectedValues If non-null, these are the property-values the server will check as condition for this update.

webFlags Defines if this `SetCustomProperties`-operation gets forwarded to your WebHooks. Client must be in room.

override string ToString ()

Brief summary string of the **Player**. Includes name or `player.ID` and if it's the Master Client.

string ToStringFull ()

String summary of the **Player**: `player.ID`, name and all custom properties of this user.

Use with care and not every frame! Converts the `customProperties` to a String on every single call.

Member Data Documentation

readonly bool IsLocal

Only one player is controlled by each client. Others are not local.

object TagObject

Can be used to store a reference that's useful to know "by player".

Example: Set a player's character as Tag by assigning the GameObject on Instantiate.

Property Documentation

int ActorNumber

get

Identifier of this player in current room. Also known as: actorNumber or actorNumber. It's -1 outside of rooms.

The ID is assigned per room and only valid in that context. It will change even on leave and re-join. IDs are never re-used per room.

Hashtable CustomProperties

get set

Read-only cache for custom properties of player. Set via **Player.SetCustomProperties**.

Don't modify the content of this Hashtable. Use SetCustomProperties and the properties of this class to modify values. When you use those, the client will sync values with the server.

SetCustomProperties

bool IsInactive

get set

If this player is active in the room (and getting events which are currently being sent).

Inactive players keep their spot in a room but otherwise behave as if offline (no matter what their actual connection status is). The room needs a PlayerTTL != 0. If a player is inactive for longer than PlayerTTL, the server will remove this player from the room. For a client "rejoining" a room, is the same as joining it: It gets properties, cached events and then the live events.

bool IsMasterClient

get

True if this player is the Master Client of the current room.

See also: **PhotonNetwork.MasterClient**.

string NickName

get

set

Non-unique nickname of this player. Synced automatically in a room.

A player might change his own playername in a room (it's only a property). Setting this value updates the server and other players (using an operation).

string UserId

get

set

UserId of the player, available when the room got created with **RoomOptions.PublishUserId** = true.

Useful for PhotonNetwork.FindFriends and blocking slots in a room for expected players (e.g. in **PhotonNetwork.CreateRoom**).



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	RaiseEventOptions		

[Public Attributes](#) | [Static Public Attributes](#) |

[List of all members](#)

RaiseEventOptions Class Reference

Aggregates several less-often used options for operation RaiseEvent.
See field descriptions for usage details. [More...](#)

Public Attributes

EventCaching **CachingOption**

Defines if the server should simply send the event, put it in the cache or remove events that are like this one. [More...](#)

byte **InterestGroup**

The number of the Interest Group to send this to. 0 goes to all users but to get 1 and up, clients must subscribe to the group first. [More...](#)

int[] **TargetActors**

A list of Player.ActorNumbers to send this event to. You can implement events that just go to specific users this way. [More...](#)

ReceiverGroup **Receivers**

Sends the event to All, MasterClient or Others (default). Be careful with MasterClient, as the client might disconnect before it got the event and it gets lost. [More...](#)

byte **SequenceChannel**

Events are ordered per "channel". If you have events that are independent of others, they can go into another sequence or channel. [More...](#)

WebFlags **Flags = WebFlags.Default**

Optional flags to be used in **Photon** client SDKs with Op RaiseEvent and Op SetProperties. [More...](#)

Static Public Attributes

static readonly **RaiseEventOptions** **Default** = new **RaiseEventOptions()**
Default options:
CachingOption: DoNotCache,
InterestGroup: 0, targetActors:
null, receivers: Others,
sequenceChannel: 0. [More...](#)

Detailed Description

Aggregates several less-often used options for operation RaiseEvent. See field descriptions for usage details.

Member Data Documentation

EventCaching CachingOption

Defines if the server should simply send the event, put it in the cache or remove events that are like this one.

When using option: SliceSetIndex, SlicePurgeIndex or SlicePurgeUpToIndex, set a CacheSliceIndex. All other options except SequenceChannel get ignored.

readonly RaiseEventOptions Default = new RaiseEventOptions()

static

Default options: CachingOption: DoNotCache, InterestGroup: 0, targetActors: null, receivers: Others, sequenceChannel: 0.

WebFlags Flags = WebFlags.Default

Optional flags to be used in **Photon** client SDKs with Op RaiseEvent and Op SetProperties.

Introduced mainly for webhooks 1.2 to control behavior of forwarded HTTP requests.

byte InterestGroup

The number of the Interest Group to send this to. 0 goes to all users but to get 1 and up, clients must subscribe to the group first.

ReceiverGroup Receivers

Sends the event to All, MasterClient or Others (default). Be careful with MasterClient, as the client might disconnect before it got the event and it gets lost.

byte SequenceChannel

Events are ordered per "channel". If you have events that are independent of others, they can go into another sequence or channel.

int [] TargetActors

A list of Player.ActorNumbers to send this event to. You can implement events that just go to specific users this way.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	Region		

[Public Member Functions](#) | [Properties](#) |

[List of all members](#)

Region Class Reference

Public Member Functions

Region (string code, string address)

Region (string code, int ping)

override string **ToString** ()

string **ToString** (bool compact=false)

Properties

string **Code** [get]

string **Cluster** [get]

Unlike the CloudRegionCode, this may contain cluster information. [More...](#)

string **HostAndPort** [get, set]

int **Ping** [get, set]

bool **WasPinged** [get]

Property Documentation

string Cluster

get

Unlike the CloudRegionCode, this may contain cluster information.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	RegionHandler		

[Public Member Functions](#) | [Properties](#) |

[List of all members](#)

RegionHandler Class Reference

Provides methods to work with **Photon's** regions (**Photon Cloud**) and can be use to find the one with best ping. [More...](#)

Public Member Functions

void **SetRegions** (OperationResponse opGetRegions)

bool **PingMinimumOfRegions** (Action< **RegionHandler** >
onCompleteCallback, string previousSummary)

Properties

List< **Region** > **EnabledRegions** [get, set]
A list of region names for the **Photon** Cloud. Set by the result of OpGetRegions(). [More...](#)

Region **BestRegion** [get]
When PingMinimumOfRegions was called and completed, the BestRegion is identified by best ping. [More...](#)

string **SummaryToCache** [get]
This value summarizes the results of pinging the currently available EnabledRegions (after PingMinimumOfRegions finished). [More...](#)

bool **IsPinging** [get]

Detailed Description

Provides methods to work with **Photon**'s regions (**Photon Cloud**) and can be use to find the one with best ping.

When a client uses a Name Server to fetch the list of available regions, the **LoadBalancingClient** will create a **RegionHandler** and provide it via the OnRegionListReceived callback.

Your logic can decide to either connect to one of those regional servers, or it may use PingMinimumOfRegions to test which region provides the best ping.

It makes sense to make clients "sticky" to a region when one gets selected. This can be achieved by storing the SummaryToCache value, once pinging was done. When the client connects again, the previous SummaryToCache helps limiting the number of regions to ping. In best case, only the previously selected region gets re-pinged and if the current ping is not much worse, this one region is used again.

Property Documentation

Region BestRegion

get

When PingMinimumOfRegions was called and completed, the BestRegion is identified by best ping.

List<Region> EnabledRegions

get

set

A list of region names for the **Photon** Cloud. Set by the result of OpGetRegions().

Implement ILoadBalancingCallbacks and register for the callbacks to get OnRegionListReceived(RegionHandler regionHandler). You can also put a "case OperationCode.GetRegions:" into your OnOperationResponse method to notice when the result is available.

string SummaryToCache

get

This value summarizes the results of pinging the currently available EnabledRegions (after PingMinimumOfRegions finished).



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	RegionPinger		

RegionPinger Class Reference

[Public Member Functions](#) |
[Static Public Member Functions](#) |
[Public Attributes](#) | [Static Public Attributes](#) |
[Properties](#) | [List of all members](#)

Public Member Functions

RegionPinger (**Region** region, Action< **Region** >
onDoneCallback)

bool **Start** ()

Static Public Member Functions

static string **ResolveHost** (string hostName)

Attempts to resolve a hostname into an IP string or returns empty string if that fails. [More...](#)

Public Attributes

```
int CurrentAttempt = 0
```

Static Public Attributes

```
static int Attempts = 5
```

```
static bool IgnoreInitialAttempt = true
```

```
static int MaxMilliseconsPerPing = 800
```

```
static int PingWhenFailed = Attempts * MaxMilliseconsPerPing
```

Properties

bool **Done** [get]

Member Function Documentation

static string ResolveHost (string **hostName)**

static

Attempts to resolve a hostname into an IP string or returns empty string if that fails.

To be compatible with most platforms, the address family is checked like this:

if (ipAddress.AddressFamily.ToString().Contains("6")) // ipv6...

Parameters

hostName Hostname to resolve.

Returns

IP string or empty string if resolution fails



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	Room		

Room Class Reference

[Public Member Functions](#) | [Properties](#) |

[List of all members](#)

This class represents a room a client joins/joined. [More...](#)

Inherits **RoomInfo**.

Public Member Functions

Room (string roomName, **RoomOptions** options, bool isOffline=false)
Creates a **Room** (representation) with given name and properties and the "listing options" as provided by parameters. [More...](#)

virtual void **SetCustomProperties** (Hashtable propertiesToSet, Hashtable expectedProperties=null, **WebFlags** webFlags=null)
Updates and synchronizes this **Room**'s Custom Properties. Optionally, expectedProperties can be provided as condition. [More...](#)

void **SetPropertiesListedInLobby** (string[] propertiesListedInLobby)
Enables you to define the properties available in the lobby if not all properties are needed to pick a room. [More...](#)

bool **SetMasterClient** (**Player** masterClientPlayer)
Asks the server to assign another player as Master Client of your current room. [More...](#)

virtual bool **AddPlayer** (**Player** player)
Checks if the player is in the room's list already and calls **StorePlayer()** if not. [More...](#)

virtual **Player** **StorePlayer** (**Player** player)
Updates a player reference in the Players dictionary (no matter if it existed before or not). [More...](#)

virtual **Player** **GetPlayer** (int id)
Tries to find the player with given actorNumber (a.k.a. ID). Only useful when in a **Room**, as IDs are only valid per **Room**. [More...](#)

void **ClearExpectedUsers ()**

Attempts to remove all current expected users from the server's Slot Reservation list. [More...](#)

override string **ToString ()**

Returns a summary of this **Room** instance as string. [More...](#)

new string **ToStringFull ()**

Returns a summary of this **Room** instance as longer string, including Custom Properties. [More...](#)

► **Public Member Functions inherited from RoomInfo**

override bool **Equals (object other)**

Makes **RoomInfo** comparable (by name). [More...](#)

override int **GetHashCode ()**

Accompanies Equals, using the name's GetHashCode as return. [More...](#)

override string **ToString ()**

Returns most interesting room values as string. [More...](#)

string **ToStringFull ()**

Returns most interesting room values as string, including custom properties. [More...](#)

Properties

LoadBalancingClient **LoadBalancingClient** [get, set]
A reference to the **LoadBalancingClient** which is currently keeping the connection and state. [More...](#)

new string **Name** [get, set]
The name of a room. Unique identifier (per region and virtual appid) for a room/match. [More...](#)

bool **IsOffline** [get]

new bool **IsOpen** [get, set]
Defines if the room can be joined. [More...](#)

new bool **IsVisible** [get, set]
Defines if the room is listed in its lobby. [More...](#)

new byte **MaxPlayers** [get, set]
Sets a limit of players to this room. This property is synced and shown in lobby, too. If the room is full (players count == maxplayers), joining this room will fail. [More...](#)

new byte **PlayerCount** [get]
The count of players in this **Room** (using this.Players.Count). [More...](#)

Dictionary< int, **Player** > **Players** [get]
While inside a **Room**, this is the list of players who are also in that room. [More...](#)

string[] **ExpectedUsers** [get]
List of users who are expected to join this room. In matchmaking, **Photon** blocks a slot for each of these UserIDs out of the MaxPlayers. [More...](#)

int **PlayerTtl** [get, set]
Player Time To Live. How long any player can be inactive (due to disconnect or leave) before the user gets removed from the playerlist (freeing a slot). [More...](#)

int **EmptyRoomTtl** [get, set]
Room Time To Live. How long a room stays available (and in server-memory), after the last player becomes inactive. After this time, the room gets persisted or destroyed. [More...](#)

int **MasterClientId** [get]
The ID (actorNumber, actorNumber) of the player who's the master of this **Room**. Note: This changes when the current master leaves the room. [More...](#)

string[] **PropertiesListedInLobby** [get]
Gets a list of custom properties that are in the **RoomInfo** of the Lobby. This list is defined when creating the room and can't be changed afterwards. Compare: **LoadBalancingClient.OpCreateRoom()** [More...](#)

bool **AutoCleanUp** [get]
Gets if this room uses autoCleanUp to remove all (buffered) RPCs and instantiated GameObjects when a player

leaves. [More...](#)

► Properties inherited from **RoomInfo**

Hashtable **CustomProperties** [get]

Read-only "cache" of custom properties of a room. Set via

Room.SetCustomProperties (not available for **RoomInfo** class!). [More...](#)

string **Name** [get]

The name of a room. Unique identifier for a room/match (per AppId + game-Version). [More...](#)

int **PlayerCount** [get]

Count of players currently in room. This property is overwritten by the **Room** class (used when you're in a **Room**). [More...](#)

byte **MaxPlayers** [get]

The limit of players for this room. This property is shown in lobby, too. If the room is full (players count == maxplayers), joining this room will fail. [More...](#)

bool **IsOpen** [get]

Defines if the room can be joined. This does not affect listing in a lobby but joining the room will fail if not open. If not open, the room is excluded from random matchmaking. Due to racing conditions, found matches might become closed even while you join them. Simply re-connect to master and find another. Use property "IsVisible" to not list the room. [More...](#)

bool IsVisible [get]

Defines if the room is listed in its lobby.
Rooms can be created invisible, or
changed to invisible. To change if a room
can be joined, use property: open. [More...](#)

Additional Inherited Members

‣ Public Attributes inherited from RoomInfo

bool **RemovedFromList**

Used in lobby, to mark rooms that are no longer listed (for being full, closed or hidden). [More...](#)

int **masterClientId**

Backing field for master client id (actorNumber). defined by server in room props and ev leave. [More...](#)

‣ Protected Attributes inherited from RoomInfo

byte **maxPlayers** = 0

Backing field for property. [More...](#)

int **emptyRoomTtl** = 0

Backing field for property. [More...](#)

int **playerTtl** = 0

Backing field for property. [More...](#)

string[] **expectedUsers**

Backing field for property. [More...](#)

bool **isOpen** = true

Backing field for property. [More...](#)

bool **isVisible** = true

Backing field for property. [More...](#)

bool **autoCleanUp** = true

Backing field for property. False unless the GameProperty is set to true (else it's not sent). [More...](#)

string **name**

Backing field for property. [More...](#)

string[] **propertiesListedInLobby**
Backing field for property. [More...](#)

Detailed Description

This class represents a room a client joins/joined.

Contains a list of current players, their properties and those of this room, too. A room instance has a number of "well known" properties like `IsOpen`, `MaxPlayers` which can be changed. Your own, custom properties can be set via **`SetCustomProperties()`** while being in the room.

Typically, this class should be extended by a game-specific implementation with logic and extra features.

Constructor & Destructor Documentation

```
Room ( string      roomName,  
      RoomOptions options,  
      bool        isOffline = false  
    )
```

Creates a **Room** (representation) with given name and properties and the "listing options" as provided by parameters.

Parameters

roomName Name of the room (can be null until it's actually created on server).
options **Room** options.

Member Function Documentation

virtual bool AddPlayer (Player **player**)

virtual

Checks if the player is in the room's list already and calls **StorePlayer()** if not.

Parameters

player The new player - identified by ID.

Returns

False if the player could not be added (cause it was in the list already).

void ClearExpectedUsers ()

Attempts to remove all current expected users from the server's Slot Reservation list.

Note that this operation can conflict with new/other users joining. They might be adding users to the list of expected users before or after this client called ClearExpectedUsers.

This room's expectedUsers value will update, when the server sends a successful update.

Internals: This methods wraps up setting the ExpectedUsers property of a room.

virtual Player GetPlayer (int **id**)

virtual

Tries to find the player with given actorNumber (a.k.a. ID). Only useful when in a **Room**, as IDs are only valid per **Room**.

Parameters

id ID to look for.

Returns

The player with the ID or null.

virtual void

```
SetCustomProperties ( Hashtable propertiesToSet,  
                    Hashtable expectedProperties = null,  
                    WebFlags webFlags = null  
                    )
```

virtua

Updates and synchronizes this **Room**'s Custom Properties. Optionally, **expectedProperties** can be provided as condition.

Custom Properties are a set of string keys and arbitrary values which is synchronized for the players in a **Room**. They are available when the client enters the room, as they are in the response of OpJoin and OpCreate.

Custom Properties either relate to the (current) **Room** or a **Player** (in that **Room**).

Both classes locally cache the current key/values and make them available as property: CustomProperties. This is provided only to read them. You must use the method SetCustomProperties to set/modify them.

Any client can set any Custom Properties anytime (when in a room). It's up to the game logic to organize how they are best used.

You should call SetCustomProperties only with key/values that are new or changed. This reduces traffic and performance.

Unless you define some **expectedProperties**, setting key/values is always permitted. In this case, the property-setting client will not receive the new values from the server but instead update its local cache in SetCustomProperties.

If you define `expectedProperties`, the server will skip updates if the server property-cache does not contain all `expectedProperties` with the same values. In this case, the property-setting client will get an update from the server and update its cached key/values at about the same time as everyone else.

The benefit of using `expectedProperties` can be only one client successfully sets a key from one known value to another. As example: Store who owns an item in a Custom Property "ownedBy". It's 0 initially. When multiple players reach the item, they all attempt to change "ownedBy" from 0 to their `actorNumber`. If you use `expectedProperties {"ownedBy", 0}` as condition, the first player to take the item will have it (and the others fail to set the ownership).

Properties get saved with the game state for Turnbased games (which use `IsPersistent = true`).

Parameters

propertiesToSet	Hashtable of Custom Properties that changes.
expectedProperties	Provide some keys/values to use as condition for setting the new values. Client must be in room.
webFlags	Defines if this <code>SetCustomProperties</code> -operation gets forwarded to your WebHooks. Client must be in room.

bool SetMasterClient (Player **masterClientPlayer)**

Asks the server to assign another player as Master Client of your current room.

`RaiseEvent` has the option to send messages only to the Master Client of a room. `SetMasterClient` affects which client gets those messages.

This method calls an operation on the server to set a new Master Client, which takes a roundtrip. In case of success, this client and the others get the new Master Client from the server.

SetMasterClient tells the server which current Master Client should be replaced with the new one. It will fail, if anything switches the Master Client moments earlier. There is no callback for this error. All clients should get the new Master Client assigned by the server anyways.

See also: MasterClientId

Parameters

masterClientPlayer The player to become the next Master Client.

Returns

False when this operation couldn't be done currently. Requires a v4 **Photon** Server.

void

SetPropertiesListedInLobby (string[] **propertiesListedInLobby)**

Enables you to define the properties available in the lobby if not all properties are needed to pick a room.

Limit the amount of properties sent to users in the lobby to improve speed and stability.

Parameters

propertiesListedInLobby An array of custom room property names to forward to the lobby.

virtual Player StorePlayer (Player **player)**

virtual

Updates a player reference in the Players dictionary (no matter if it existed before or not).

Parameters

player The **Player** instance to insert into the room.

override string ToString ()

Returns a summary of this **Room** instance as string.

Returns

Summary of this **Room** instance.

new string ToStringFull ()

Returns a summary of this **Room** instance as longer string, including Custom Properties.

Returns

Summary of this **Room** instance.

Property Documentation

bool AutoCleanUp

get

Gets if this room uses autoCleanUp to remove all (buffered) RPCs and instantiated GameObjects when a player leaves.

int EmptyRoomTtl

get set

Room Time To Live. How long a room stays available (and in server-memory), after the last player becomes inactive. After this time, the room gets persisted or destroyed.

string [] ExpectedUsers

get

List of users who are expected to join this room. In matchmaking, **Photon** blocks a slot for each of these UserIDs out of the MaxPlayers.

The corresponding feature in **Photon** is called "Slot Reservation" and can be found in the doc pages. Define expected players in the PhotonNetwork methods: CreateRoom, JoinRoom and JoinOrCreateRoom.

new bool IsOpen

get set

Defines if the room can be joined.

This does not affect listing in a lobby but joining the room will fail if not open. If not open, the room is excluded from random matchmaking. Due to racing conditions, found matches might

become closed while users are trying to join. Simply re-connect to master and find another. Use property "IsVisible" to not list the room.

As part of **RoomInfo** this can't be set. As part of a **Room** (which the player joined), the setter will update the server and all clients.

new bool IsVisible

get set

Defines if the room is listed in its lobby.

Rooms can be created invisible, or changed to invisible. To change if a room can be joined, use property: open.

As part of **RoomInfo** this can't be set. As part of a **Room** (which the player joined), the setter will update the server and all clients.

LoadBalancingClient LoadBalancingClient

get set

A reference to the **LoadBalancingClient** which is currently keeping the connection and state.

int MasterClientId

get

The ID (actorNumber, actorNumber) of the player who's the master of this **Room**. Note: This changes when the current master leaves the room.

new byte MaxPlayers

get set

Sets a limit of players to this room. This property is synced and shown in lobby, too. If the room is full (players count == maxplayers), joining this room will fail.

As part of **RoomInfo** this can't be set. As part of a **Room** (which the

player joined), the setter will update the server and all clients.

new string Name

get set

The name of a room. Unique identifier (per region and virtual appid) for a room/match.

The name can't be changed once it's set by the server.

new byte PlayerCount

get

The count of players in this **Room** (using this.Players.Count).

Dictionary<int, Player> Players

get

While inside a **Room**, this is the list of players who are also in that room.

int PlayerTtl

get set

Player Time To Live. How long any player can be inactive (due to disconnect or leave) before the user gets removed from the playerlist (freeing a slot).

string [] PropertiesListedInLobby

get

Gets a list of custom properties that are in the **RoomInfo** of the Lobby. This list is defined when creating the room and can't be changed afterwards. Compare:

LoadBalancingClient.OpCreateRoom()

You could name properties that are not set from the beginning. Those will be synced with the lobby when added later on.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	RoomInfo		

RoomInfo Class Reference

[Public Member Functions](#) | [Public Attributes](#) |
[Protected Attributes](#) | [Properties](#) |
[List of all members](#)

A simplified room with just the info required to list and join, used for the room listing in the lobby. The properties are not settable (IsOpen, MaxPlayers, etc). [More...](#)

Inherited by **Room**.

Public Member Functions

override bool **Equals** (object other)
Makes **RoomInfo** comparable (by name). [More...](#)

override int **GetHashCode** ()
Accompanies Equals, using the name's HashCode as return. [More...](#)

override string **ToString** ()
Returns most interesting room values as string.
[More...](#)

string **ToStringFull** ()
Returns most interesting room values as string, including custom properties. [More...](#)

Public Attributes

bool **RemovedFromList**

Used in lobby, to mark rooms that are no longer listed (for being full, closed or hidden). [More...](#)

int **masterClientId**

Backing field for master client id (actorNumber). defined by server in room props and ev leave. [More...](#)

Protected Attributes

byte **maxPlayers** = 0
Backing field for property. [More...](#)

int **emptyRoomTtl** = 0
Backing field for property. [More...](#)

int **playerTtl** = 0
Backing field for property. [More...](#)

string[] **expectedUsers**
Backing field for property. [More...](#)

bool **isOpen** = true
Backing field for property. [More...](#)

bool **isVisible** = true
Backing field for property. [More...](#)

bool **autoCleanUp** = true
Backing field for property. False unless the GameProperty is set to true (else it's not sent). [More...](#)

string **name**
Backing field for property. [More...](#)

string[] **propertiesListedInLobby**
Backing field for property. [More...](#)

Properties

Hashtable **CustomProperties** [get]

Read-only "cache" of custom properties of a room. Set via **Room.SetCustomProperties** (not available for **RoomInfo** class!). [More...](#)

string **Name** [get]

The name of a room. Unique identifier for a room/match (per AppId + game-Version). [More...](#)

int **PlayerCount** [get]

Count of players currently in room. This property is overwritten by the **Room** class (used when you're in a **Room**). [More...](#)

byte **MaxPlayers** [get]

The limit of players for this room. This property is shown in lobby, too. If the room is full (players count == maxplayers), joining this room will fail. [More...](#)

bool **IsOpen** [get]

Defines if the room can be joined. This does not affect listing in a lobby but joining the room will fail if not open. If not open, the room is excluded from random matchmaking. Due to racing conditions, found matches might become closed even while you join them. Simply re-connect to master and find another. Use property "IsVisible" to not list the room. [More...](#)

bool **IsVisible** [get]

Defines if the room is listed in its lobby. Rooms can be created invisible, or changed to invisible. To change if a room can be joined, use property: open. [More...](#)

Detailed Description

A simplified room with just the info required to list and join, used for the room listing in the lobby. The properties are not settable (IsOpen, MaxPlayers, etc).

This class resembles info about available rooms, as sent by the Master server's lobby. Consider all values as readonly. None are synced (only updated by events by server).

Member Function Documentation

override bool Equals (object *other*)

Makes **RoomInfo** comparable (by name).

override int GetHashCode ()

Accompanies Equals, using the name's HashCode as return.

Returns

override string ToString ()

Returns most interesting room values as string.

Returns

Summary of this **RoomInfo** instance.

string ToStringFull ()

Returns most interesting room values as string, including custom properties.

Returns

Summary of this **RoomInfo** instance.

Member Data Documentation

bool autoCleanUp = true

protected

Backing field for property. False unless the GameProperty is set to true (else it's not sent).

int emptyRoomTtl = 0

protected

Backing field for property.

string [] expectedUsers

protected

Backing field for property.

bool isOpen = true

protected

Backing field for property.

bool isVisible = true

protected

Backing field for property.

int masterClientId

Backing field for master client id (actorNumber). defined by server in room props and ev leave.

byte maxPlayers = 0

protected

Backing field for property.

string name

protected

Backing field for property.

int playerTtl = 0

protected

Backing field for property.

string [] propertiesListedInLobby

protected

Backing field for property.

bool RemovedFromList

Used in lobby, to mark rooms that are no longer listed (for being full, closed or hidden).

Property Documentation

Hashtable CustomProperties

get

Read-only "cache" of custom properties of a room. Set via **Room.SetCustomProperties** (not available for **RoomInfo** class!).

All keys are string-typed and the values depend on the game/application.

Room.SetCustomProperties

bool IsOpen

get

Defines if the room can be joined. This does not affect listing in a lobby but joining the room will fail if not open. If not open, the room is excluded from random matchmaking. Due to racing conditions, found matches might become closed even while you join them. Simply re-connect to master and find another. Use property "IsVisible" to not list the room.

As part of **RoomInfo** this can't be set. As part of a **Room** (which the player joined), the setter will update the server and all clients.

bool IsVisible

get

Defines if the room is listed in its lobby. Rooms can be created invisible, or changed to invisible. To change if a room can be joined, use property: open.

As part of **RoomInfo** this can't be set. As part of a **Room** (which the player joined), the setter will update the server and all clients.

byte MaxPlayers

get

The limit of players for this room. This property is shown in lobby, too. If the room is full (players count == maxplayers), joining this room will fail.

As part of **RoomInfo** this can't be set. As part of a **Room** (which the player joined), the setter will update the server and all clients.

string Name

get

The name of a room. Unique identifier for a room/match (per AppId + game-Version).

int PlayerCount

get

Count of players currently in room. This property is overwritten by the **Room** class (used when you're in a **Room**).



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	RoomOptions		

[Public Attributes](#) | [Properties](#) |

[List of all members](#)

RoomOptions Class Reference

Wraps up common room properties needed when you create rooms.
Read the individual entries for more details. [More...](#)

Public Attributes

byte **MaxPlayers**

Max number of players that can be in the room at any time. 0 means "no limit". [More...](#)

int **PlayerTtl**

Time To Live (TTL) for an 'actor' in a room. If a client disconnects, this actor is inactive first and removed after this timeout. In milliseconds. [More...](#)

int **EmptyRoomTtl**

Time To Live (TTL) for a room when the last player leaves. Keeps room in memory for case a player re-joins soon. In milliseconds. [More...](#)

Hashtable **CustomRoomProperties**

The room's custom properties to set. Use string keys! [More...](#)

string[] **CustomRoomPropertiesForLobby** = new string[0]

Defines the custom room properties that get listed in the lobby. [More...](#)

string[] **Plugins**

Informs the server of the expected plugin setup. [More...](#)

Properties

bool **IsVisible** [get, set]

Defines if this room is listed in the lobby. If not, it also is not joined randomly. [More...](#)

bool **IsOpen** [get, set]

Defines if this room can be joined at all. [More...](#)

bool **CleanupCacheOnLeave** [get, set]

Removes a user's events and properties from the room when a user leaves. [More...](#)

bool **SuppressRoomEvents** [get, set]

Tells the server to skip room events for joining and leaving players. [More...](#)

bool **PublishUserId** [get, set]

Defines if the UserIds of players get "published" in the room. Useful for FindFriends, if players want to play another game together. [More...](#)

bool **DeleteNullProperties** [get, set]

Optionally, properties get deleted, when null gets assigned as value. Defaults to off / false. [More...](#)

bool **BroadcastPropsChangeToAll** [get, set]

By default, property changes are sent back to the client that's setting them to avoid de-sync when properties are set concurrently. [More...](#)

Detailed Description

Wraps up common room properties needed when you create rooms.
Read the individual entries for more details.

This directly maps to the fields in the **Room** class.

Member Data Documentation

Hashtable CustomRoomProperties

The room's custom properties to set. Use string keys!

Custom room properties are any key-values you need to define the game's setup. The shorter your keys are, the better. Example: Map, Mode (could be "m" when used with "Map"), TileSet (could be "t").

string [] CustomRoomPropertiesForLobby = new string[0]

Defines the custom room properties that get listed in the lobby.

Name the custom room properties that should be available to clients that are in a lobby. Use with care. Unless a custom property is essential for matchmaking or user info, it should not be sent to the lobby, which causes traffic and delays for clients in the lobby.

Default: No custom properties are sent to the lobby.

int EmptyRoomTtl

Time To Live (TTL) for a room when the last player leaves. Keeps room in memory for case a player re-joins soon. In milliseconds.

byte MaxPlayers

Max number of players that can be in the room at any time. 0 means "no limit".

int PlayerTtl

Time To Live (TTL) for an 'actor' in a room. If a client disconnects, this actor is inactive first and removed after this timeout. In milliseconds.

string [] Plugins

Informs the server of the expected plugin setup.

The operation will fail in case of a plugin mismatch returning error code `PluginMismatch 32757(0x7FFF - 10)`. Setting `string[]{}` means the client expects no plugin to be setup. Note: for backwards compatibility null omits any check.

Property Documentation

bool BroadcastPropsChangeToAll

get **set**

By default, property changes are sent back to the client that's setting them to avoid de-sync when properties are set concurrently.

This option is enabled by default to fix this scenario:

1) On server, room property ABC is set to value FOO, which triggers notifications to all the clients telling them that the property changed.
2) While that notification is in flight, a client sets the ABC property to value BAR.
3) Client receives notification from the server and changes its local copy of ABC to FOO.
4) Server receives the set operation and changes the official value of ABC to BAR, but never notifies the client that sent the set operation that the value is now BAR.

Without this option, the client that set the value to BAR never hears from the server that the official copy has been updated to BAR, and thus gets stuck with a value of FOO.

bool CleanupCacheOnLeave

get **set**

Removes a user's events and properties from the room when a user leaves.

This makes sense when in rooms where players can't place items in the room and just vanish entirely. When you disable this, the event history can become too long to load if the room stays in use indefinitely. Default: true. Cleans up the cache and props of leaving users.

bool DeleteNullProperties

get **set**

Optionally, properties get deleted, when null gets assigned as value. Defaults to off / false.

When Op SetProperty is setting a key's value to null, the server and clients should remove the key/value from the Custom Properties. By default, the server keeps the keys (and null values) and sends them to joining players.

Important: Only when SetProperty does a "broadcast", the change (key, value = null) is sent to clients to update accordingly. This applies to Custom Properties for rooms and actors/players.

bool IsOpen

get set

Defines if this room can be joined at all.

If a room is closed, no player can join this. As example this makes sense when 3 of 4 possible players start their gameplay early and don't want anyone to join during the game. The room can still be listed in the lobby (set isVisible to control lobby-visibility).

bool isVisible

get set

Defines if this room is listed in the lobby. If not, it also is not joined randomly.

A room that is not visible will be excluded from the room lists that are sent to the clients in lobbies. An invisible room can be joined by name but is excluded from random matchmaking.

Use this to "hide" a room and simulate "private rooms". Players can exchange a roomname and create it invisible to avoid anyone else joining it.

bool PublishUserId

get set

Defines if the UserIds of players get "published" in the room. Useful for FindFriends, if players want to play another game together.

When you set this to true, **Photon** will publish the UserIds of the players in that room. In that case, you can use PhotonPlayer.userId, to access any player's userID. This is useful for FindFriends and to set "expected users" to reserve slots in a room (see **PhotonNetwork.JoinRoom** e.g.).

bool SuppressRoomEvents

get set

Tells the server to skip room events for joining and leaving players.

Using this makes the client unaware of the other players in a room. That can save some traffic if you have some server logic that updates players but it can also limit the client's usability.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	SupportLogger		

SupportLogger Class Reference

[Classes](#) | [Public Member Functions](#) |
[Public Attributes](#) | [Properties](#) |
[List of all members](#)

Helper class to debug log basic information about **Photon** client and vital traffic statistics. [More...](#)

Inherits **ICallbacks**, **IInRoomCallbacks**, **IMatchmakingCallbacks**, and **ILobbyCallbacks**.

Public Member Functions

void **LogStats** ()

Debug logs vital traffic statistics about the attached **Photon** Client. [More...](#)

void **OnConnected** ()

Called to signal that the "low level connection" got established but before the client can call operation on the server. [More...](#)

void **OnConnectedToMaster** ()

Called when the client is connected to the Master Server and ready for matchmaking and other tasks. [More...](#)

void **OnFriendListUpdate** (List< **FriendInfo** > friendList)

Called when the server sent the response to a FindFriends request. [More...](#)

void **OnJoinedLobby** ()

Called on entering a lobby on the Master Server. The actual room-list updates will call OnRoomListUpdate. [More...](#)

void **OnLeftLobby** ()

Called after leaving a lobby. [More...](#)

void **OnCreateRoomFailed** (short returnCode, string message)

Called when the server couldn't create a room (OpCreateRoom failed). [More...](#)

void **OnJoinedRoom** ()

Called when the **LoadBalancingClient** entered a room, no matter if this client created it or simply joined. [More...](#)

void **OnJoinRoomFailed** (short returnCode, string message)

Called when a previous OpJoinRoom call failed on the server. [More...](#)

void **OnJoinRandomFailed** (short returnCode, string message)
Called when a previous OpJoinRandom call failed on the server. [More...](#)

void **OnCreatedRoom** ()
Called when this client created a room and entered it. **OnJoinedRoom()** will be called as well. [More...](#)

void **OnLeftRoom** ()
Called when the local user/client left a room, so the game's logic can clean up it's internal state. [More...](#)

void **OnDisconnected** (**DisconnectCause** cause)
Called after disconnecting from the **Photon** server. It could be a failure or an explicit disconnect call [More...](#)

void **OnRegionListReceived** (**RegionHandler** regionHandler)
Called when the Name Server provided a list of regions for your title. [More...](#)

void **OnRoomListUpdate** (List< **RoomInfo** > roomList)
Called for any update of the room-listing while in a lobby (InLobby) on the Master Server. [More...](#)

void **OnPlayerEnteredRoom** (**Player** newPlayer)
Called when a remote player entered the room. This **Player** is already added to the playerlist. [More...](#)

void **OnPlayerLeftRoom** (**Player** otherPlayer)
Called when a remote player left the room or became inactive. Check otherPlayer.IsInactive. [More...](#)

void **OnRoomPropertiesUpdate** (Hashtable propertiesThatChanged)
Called when a room's custom properties changed. The propertiesThatChanged contains all that was set via

Room.SetCustomProperties. [More...](#)

void **OnPlayerPropertiesUpdate** (**Player** targetPlayer, Hashtable changedProps)
Called when custom player-properties are changed. **Player** and the changed properties are passed as object[]. [More...](#)

void **OnMasterClientSwitched** (**Player** newMasterClient)
Called after switching to a new MasterClient when the current one leaves. [More...](#)

void **OnCustomAuthenticationResponse** (Dictionary< string, object > data)
Called when your Custom Authentication service responds with additional data. [More...](#)

void **OnCustomAuthenticationFailed** (string debugMessage)
Called when the custom authentication failed. Followed by disconnect! [More...](#)

void **OnLobbyStatisticsUpdate** (List< **TypedLobbyInfo** > lobbyStatistics)
Called when the Master Server sent an update for the Lobby Statistics, updating PhotonNetwork.LobbyStatistics. [More...](#)

Public Attributes

bool **LogTrafficStats**

Toggle to enable or disable traffic statistics logging. [More...](#)

Properties

LoadBalancingClient Client [get, set]

Photon client to log information and statistics from. [More...](#)

Detailed Description

Helper class to debug log basic information about **Photon** client and vital traffic statistics.

Set **SupportLogger.Client** for this to work.

Member Function Documentation

void LogStats ()

Debug logs vital traffic statistics about the attached **Photon** Client.

void OnConnected ()

Called to signal that the "low level connection" got established but before the client can call operation on the server.

After the (low level transport) connection is established, the client will automatically send the Authentication operation, which needs to get a response before the client can call other operations.

Your logic should wait for either: `OnRegionListReceived` or `OnConnectedToMaster`.

This callback is useful to detect if the server can be reached at all (technically). Most often, it's enough to implement **`OnDisconnected(DisconnectCause cause)`** and check for the cause.

This is not called for transitions from the masterserver to game servers.

Implements **`IConnectionCallbacks`**.

void OnConnectedToMaster ()

Called when the client is connected to the Master Server and ready for matchmaking and other tasks.

The list of available rooms won't become available unless you join a lobby via **LoadBalancingClient.OpJoinLobby**. You can join rooms and create them even without being in a lobby. The default lobby is used in that case.

Implements **IConnectionCallbacks**.

void OnCreatedRoom ()

Called when this client created a room and entered it. **OnJoinedRoom()** will be called as well.

This callback is only called on the client which created a room (see **OpCreateRoom**).

As any client might close (or drop connection) anytime, there is a chance that the creator of a room does not execute **OnCreatedRoom**.

If you need specific room properties or a "start signal", implement **OnMasterClientSwitched()** and make each new **MasterClient** check the room's state.

Implements **IMatchmakingCallbacks**.

void OnCreateRoomFailed (short **returnCode, string **message**)**

Called when the server couldn't create a room (**OpCreateRoom** failed).

Creating a room may fail for various reasons. Most often, the room already exists (roomname in use) or the **RoomOptions** clash and it's impossible to create the room.

When creating a room fails on a Game Server: The client will cache

the failure internally and returns to the Master Server before it calls the fail-callback. This way, the client is ready to find/create a room at the moment of the callback. In this case, the client skips calling `OnConnectedToMaster` but returning to the Master Server will still call `OnConnected`. Treat callbacks of `OnConnected` as pure information that the client could connect.

Parameters

returnCode Operation ReturnCode from the server.

message Debug message for the error.

Implements **IMatchmakingCallbacks**.

void OnCustomAuthenticationFailed (string debugMessage)

Called when the custom authentication failed. Followed by disconnect!

Custom Authentication can fail due to user-input, bad tokens/secrets. If authentication is successful, this method is not called. Implement **OnJoinedLobby()** or **OnConnectedToMaster()** (as usual).

During development of a game, it might also fail due to wrong configuration on the server side. In those cases, logging the `debugMessage` is very important.

Unless you setup a custom authentication service for your app (in the [Dashboard](#)), this won't be called!

Parameters

debugMessage Contains a debug message why authentication failed. This has to be fixed during development.

Implements **IConnectionCallbacks**.

void

OnCustomAuthenticationResponse (Dictionary< string, object > c

Called when your Custom Authentication service responds with additional data.

Custom Authentication services can include some custom data in their response. When present, that data is made available in this callback as Dictionary. While the keys of your data have to be strings, the values can be either string or a number (in Json). You need to make extra sure, that the value type is the one you expect. Numbers become (currently) int64

Example: void OnCustomAuthenticationResponse(Dictionary<string, object> data) { ... }

<https://doc.photonengine.com/en-us/realtime/current/reference/custom-authentication>

Implements **ICollectionCallbacks**.

void OnDisconnected (DisconnectCause cause)

Called after disconnecting from the **Photon** server. It could be a failure or an explicit disconnect call

The reason for this disconnect is provided as DisconnectCause.

Implements **ICollectionCallbacks**.

void OnFriendListUpdate (List< FriendInfo > friendList)

Called when the server sent the response to a FindFriends request.

After calling OpFindFriends, the Master Server will cache the friend list and send updates to the friend list. The friends includes the name, userId, online state and the room (if any) for each requested user/friend.

Use the `friendList` to update your UI and store it, if the UI should highlight changes.

Implements **IMatchmakingCallbacks**.

void OnJoinedLobby ()

Called on entering a lobby on the Master Server. The actual room-list updates will call `OnRoomListUpdate`.

While in the lobby, the roomlist is automatically updated in fixed intervals (which you can't modify in the public cloud). The room list gets available via `OnRoomListUpdate`.

Implements **ILobbyCallbacks**.

void OnJoinedRoom ()

Called when the **LoadBalancingClient** entered a room, no matter if this client created it or simply joined.

When this is called, you can access the existing players in **Room.Players**, their custom properties and **Room.CustomProperties**.

In this callback, you could create player objects. For example in Unity, instantiate a prefab for the player.

If you want a match to be started "actively", enable the user to signal "ready" (using `OpRaiseEvent` or a Custom Property).

Implements **IMatchmakingCallbacks**.

```
void OnJoinRandomFailed ( short returnCode,  
                        string message  
                        )
```

Called when a previous OpJoinRandom call failed on the server.

The most common causes are that a room is full or does not exist (due to someone else being faster or closing the room).

This operation is only ever sent to the Master Server. Once a room is found by the Master Server, the client will head off to the designated Game Server and use the operation Join on the Game Server.

When using multiple lobbies (via OpJoinLobby or a **TypedLobby** parameter), another lobby might have more/fitting rooms.

Parameters

returnCode Operation ReturnCode from the server.

message Debug message for the error.

Implements **IMatchmakingCallbacks**.

```
void OnJoinRoomFailed ( short returnCode,  
                        string message  
                      )
```

Called when a previous OpJoinRoom call failed on the server.

Joining a room may fail for various reasons. Most often, the room is full or does not exist anymore (due to someone else being faster or closing the room).

When joining a room fails on a Game Server: The client will cache the failure internally and returns to the Master Server before it calls the fail-callback. This way, the client is ready to find/create a room at the moment of the callback. In this case, the client skips calling OnConnectedToMaster but returning to the Master Server will still call OnConnected. Treat callbacks of OnConnected as pure information that the client could connect.

Parameters

returnCode Operation ReturnCode from the server.

message Debug message for the error.

Implements **IMatchmakingCallbacks**.

void OnLeftLobby ()

Called after leaving a lobby.

When you leave a lobby, `OpCreateRoom` and `OpJoinRandomRoom` automatically refer to the default lobby.

Implements **ILobbyCallbacks**.

void OnLeftRoom ()

Called when the local user/client left a room, so the game's logic can clean up it's internal state.

When leaving a room, the **LoadBalancingClient** will disconnect the Game Server and connect to the Master Server. This wraps up multiple internal actions.

Wait for the callback `OnConnectedToMaster`, before you use lobbies and join or create rooms.

Implements **IMatchmakingCallbacks**.

void OnLobbyStatisticsUpdate (List< TypedLobbyInfo > lobbyStatistics;

Called when the Master Server sent an update for the Lobby Statistics, updating `PhotonNetwork.LobbyStatistics`.

This callback has two preconditions: `EnableLobbyStatistics` must be set

to true, before this client connects. And the client has to be connected to the Master Server, which is providing the info about lobbies.

Implements **ILobbyCallbacks**.

void OnMasterClientSwitched (Player newMasterClient)

Called after switching to a new MasterClient when the current one leaves.

This is not called when this client enters a room. The former MasterClient is still in the player list when this method gets called.

Implements **IRoomCallbacks**.

void OnPlayerEnteredRoom (Player newPlayer)

Called when a remote player entered the room. This **Player** is already added to the playerlist.

If your game starts with a certain number of players, this callback can be useful to check the Room.playerCount and find out if you can start.

Implements **IRoomCallbacks**.

void OnPlayerLeftRoom (Player otherPlayer)

Called when a remote player left the room or became inactive. Check otherPlayer.IsInactive.

If another player leaves the room or if the server detects a lost connection, this callback will be used to notify your game logic.

Depending on the room's setup, players may become inactive, which means they may return and retake their spot in the room. In such

cases, the **Player** stays in the **Room.Players** dictionary.

If the player is not just inactive, it gets removed from the **Room.Players** dictionary, before the callback is called.

Implements **IInRoomCallbacks**.

```
void OnPlayerPropertiesUpdate ( Player      targetPlayer,  
                               Hashtable changedProps  
                               )
```

Called when custom player-properties are changed. **Player** and the changed properties are passed as object[].

Changing properties must be done by **Player.SetCustomProperties**, which causes this callback locally, too.

Parameters

targetPlayer Contains **Player** that changed.

changedProps Contains the properties that changed.

Implements **IInRoomCallbacks**.

```
void OnRegionListReceived ( RegionHandler regionHandler )
```

Called when the Name Server provided a list of regions for your title.

Check the **RegionHandler** class description, to make use of the provided values.

Parameters

regionHandler The currently used **RegionHandler**.

Implements **IConnectionCallbacks**.

void OnRoomListUpdate (List< RoomInfo > **roomList)**

Called for any update of the room-listing while in a lobby (InLobby) on the Master Server.

Each item is a **RoomInfo** which might include custom properties (provided you defined those as lobby-listed when creating a room). Not all types of lobbies provide a listing of rooms to the client. Some are silent and specialized for server-side matchmaking.

Implements **ILobbyCallbacks**.

void OnRoomPropertiesUpdate (Hashtable **propertiesThatChanged)**

Called when a room's custom properties changed. The **propertiesThatChanged** contains all that was set via **Room.SetCustomProperties**.

Since v1.25 this method has one parameter: Hashtable **propertiesThatChanged**.
Changing properties must be done by **Room.SetCustomProperties**, which causes this callback locally, too.

Parameters

propertiesThatChanged

Implements **IInRoomCallbacks**.

Member Data Documentation

bool LogTrafficStats
Toggle to enable or disable traffic statistics logging.

Property Documentation

LoadBalancingClient Client

get set

Photon client to log information and statistics from.

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	TypedLobby		

TypedLobby Class Reference

[Public Member Functions](#) | [Public Attributes](#) |
[Static Public Attributes](#) | [Properties](#) |
[List of all members](#)

Refers to a specific lobby (and type) on the server. [More...](#)

Inherited by **TypedLobbyInfo**.

Public Member Functions

TypedLobby (string name, **LobbyType** type)

override string **ToString** ()

Public Attributes

string **Name**

Name of the lobby this game gets added to. Default: null, attached to default lobby. Lobbies are unique per lobbyName plus lobbyType, so the same name can be used when several types are existing. [More...](#)

LobbyType **Type**

Type of the (named)lobby this game gets added to
[More...](#)

Static Public Attributes

```
static readonly TypedLobby Default = new TypedLobby()
```

Properties

bool **IsDefault** [get]

Detailed Description

Refers to a specific lobby (and type) on the server.

The name and type are the unique identifier for a lobby.

Join a lobby via `PhotonNetwork.JoinLobby(TypedLobby lobby)`.

The current lobby is stored in `PhotonNetwork.lobby`.

Member Data Documentation

string Name

Name of the lobby this game gets added to. Default: null, attached to default lobby. Lobbies are unique per lobbyName plus lobbyType, so the same name can be used when several types are existing.

LobbyType Type

Type of the (named)lobby this game gets added to



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	TypedLobbyInfo		

TypedLobbyInfo Class Reference

[Public Member Functions](#) | [Public Attributes](#) | [List of all members](#)

Inherits `TypedLobby`.

Public Member Functions

override string **ToString** ()

‣ **Public Member Functions inherited from TypedLobby**

TypedLobby (string name, **LobbyType** type)

override string **ToString** ()

Public Attributes

int **PlayerCount**

int **RoomCount**

▸ Public Attributes inherited from TypedLobby

string **Name**

Name of the lobby this game gets added to. Default: null, attached to default lobby. Lobbies are unique per lobbyName plus lobbyType, so the same name can be used when several types are existing. [More...](#)

LobbyType **Type**

Type of the (named)lobby this game gets added to
[More...](#)

Additional Inherited Members

▸ Static Public Attributes inherited from **TypedLobby**

static readonly **TypedLobby** **Default** = new **TypedLobby**()

▸ Properties inherited from **TypedLobby**

bool **IsDefault** [get]



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	WebFlags		

WebFlags Class Reference

[Public Member Functions](#) | [Public Attributes](#) |
[Static Public Attributes](#) | [Properties](#) |
[List of all members](#)

Optional flags to be used in **Photon** client SDKs with Op RaiseEvent and Op SetProperties. Introduced mainly for webhooks 1.2 to control behavior of forwarded HTTP requests. [More...](#)

Public Member Functions

WebFlags (byte webhookFlags)

Public Attributes

byte **WebhookFlags**

const byte **HttpForwardConst** = 0x01

const byte **SendAuthCookieConst** = 0x02

const byte **SendSyncConst** = 0x04

const byte **SendStateConst** = 0x08

Static Public Attributes

```
static readonly WebFlags Default = new WebFlags(0)
```

Properties

bool **HttpForward** [get, set]

Indicates whether to forward HTTP request to web service or not. [More...](#)

bool **SendAuthCookie** [get, set]

Indicates whether to send AuthCookie of actor in the HTTP request to web service or not. [More...](#)

bool **SendSync** [get, set]

Indicates whether to send HTTP request synchronously or asynchronously to web service. [More...](#)

bool **SendState** [get, set]

Indicates whether to send serialized game state in HTTP request to web service or not. [More...](#)

Detailed Description

Optional flags to be used in **Photon** client SDKs with Op RaiseEvent and Op SetProperties. Introduced mainly for webhooks 1.2 to control behavior of forwarded HTTP requests.

Property Documentation

bool HttpForward

get **set**

Indicates whether to forward HTTP request to web service or not.

bool SendAuthCookie

get **set**

Indicates whether to send AuthCookie of actor in the HTTP request to web service or not.

bool SendState

get **set**

Indicates whether to send serialized game state in HTTP request to web service or not.

bool SendSync

get **set**

Indicates whether to send HTTP request synchronously or asynchronously to web service.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	WebRpcResponse		

[Public Member Functions](#) | [Properties](#) |

[List of all members](#)

WebRpcResponse Class Reference

Reads an operation response of a WebRpc and provides convenient access to most common values. [More...](#)

Public Member Functions

WebRpcResponse (OperationResponse response)

An OperationResponse for a WebRpc is needed to read it's values. [More...](#)

string **ToStringFull** ()

Turns the response into an easier to read string. [More...](#)

Properties

string **Name** [get]
Name of the WebRpc that was called.
[More...](#)

int **ReturnCode** [get]
ReturnCode of the WebService that answered the WebRpc. [More...](#)

string **DebugMessage** [get]
Might be empty or null. [More...](#)

Dictionary< string, object > **Parameters** [get]
Other key/values returned by the webservice that answered the WebRpc.
[More...](#)

Detailed Description

Reads an operation response of a WebRpc and provides convenient access to most common values.

See **LoadBalancingClient.OpWebRpc**.

Create a **WebRpcResponse** to access common result values.

The operationResponse.OperationCode should be:

OperationCode.WebRpc.

Constructor & Destructor Documentation

WebRpcResponse (OperationResponse *response*)

An OperationResponse for a WebRpc is needed to read it's values.

Member Function Documentation

string ToStringFull ()

Turns the response into an easier to read string.

Returns

String resembling the result.

Property Documentation

string DebugMessage

get

Might be empty or null.

string Name

get

Name of the WebRpc that was called.

Dictionary<string, object> Parameters

get

Other key/values returned by the webservice that answered the WebRpc.

int ReturnCode

get

ReturnCode of the WebService that answered the WebRpc.

1 is: "OK" for WebRPCs.

-1 is: No ReturnCode by WebRpc service (check
OperationResponse.ReturnCode).

Other ReturnCodes are defined by the individual WebRpc and
service.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	

Class Index

[A](#) | [B](#) | [C](#) | [D](#) | [E](#) | [F](#) | [G](#) | [I](#) | [L](#) | [M](#) | [O](#) | [P](#) | [R](#) | [S](#) | [T](#) | [W](#)

A

[ActorProperties \(Photon.Realtime\)](#)
[AppSettings \(Photon.Realtime\)](#)
[AuthenticationValues \(Photon.Realtime\)](#)
[AuthenticationValues \(Photon.Chat\)](#)

E

[Encry](#)
[E](#)

B

[ButtonInsideScrollList \(Photon.Pun.UtilityScripts\)](#)

[EventS](#)

C

[CellTree \(Photon.Pun.UtilityScripts\)](#)
[CellTreeNode \(Photon.Pun.UtilityScripts\)](#)
[ChannelCreationOptions \(Photon.Chat\)](#)
[ChannelWellKnownProperties \(Photon.Chat\)](#)
[ChatChannel \(Photon.Chat\)](#)
[ChatClient \(Photon.Chat\)](#)
[ChatEventCode \(Photon.Chat\)](#)
[ChatOperationCode \(Photon.Chat\)](#)
[ChatParameterCode \(Photon.Chat\)](#)
[ChatPeer \(Photon.Chat\)](#)
[ChatUserStatus \(Photon.Chat\)](#)
[ConnectAndJoinRandom \(Photon.Pun.UtilityScripts\)](#)
[ConnectionCallbacksContainer \(Photon.Realtime\)](#)
[ConnectionHandler \(Photon.Realtime\)](#)

F

G

[G](#)
[GraphicTo](#)

I

[ICc](#)
[II](#)
[I](#)
[IMat](#)
[I](#)

CountdownTimer (Photon.Pun.UtilityScripts)

I

CullArea (Photon.Pun.UtilityScripts)

IPun

CullingHandler (Photon.Pun.UtilityScripts)

IP

D

DefaultPool (Photon.Pun)

IPunTurnM

[A](#) | [B](#) | [C](#) | [D](#) | [E](#) | [F](#) | [G](#) | [H](#) | [I](#) | [J](#) | [K](#) | [L](#) | [M](#) | [N](#) | [O](#) | [P](#) | [Q](#) | [R](#) | [S](#) | [T](#) | [U](#) | [V](#) | [W](#)

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	

Class Hierarchy

















This inheritance list is sorted roughly, but not completely, alphabetically:














		[detail level 1 2 3]
☐ ActorProperties		Class for constar define "well know Actor / Player .
☐ AppSettings		Settings for Phot server to connec
▶ ☐ Attribute		
☐ PunRPC		Replacement for different name. L remote-callable.
☐ AuthenticationValues		Container for use Photon . Set Autl connect - all else
☐ AuthenticationValues		Container for use Photon . Set Autl connect - all else
☐ CellTree		Represents the ti root node.
☐ CellTreeNode		Represents a sin
☐ ChannelCreationOptions		
☐ ChannelWellKnownProperties		
☐ ChatChannel		A channel of cor Chat , updated by provided as REA























☑ ChatEventCode	Wraps up internal Photon Chat events. You can use them directly.
☑ ChatOperationCode	Wraps up codes internally in Photon Chat to use them directly.
☑ ChatParameterCode	Wraps up codes for operations and events in Photon Chat . You can use them directly usually.
☑ ChatUserStatus	Contains common status codes for SetOnlineStatus.
☑ ConnectionHandler	
☑ EncryptionDataParameters	
☑ EnterRoomParams	
☑ ErrorCode	ErrorCode definition associated with F communication.
☑ ErrorCode	ErrorCode definition associated with F communication.
☑ EventCode	Class for constants for events defined by Photon .
☑ EventExt	
☑ Extensions	This static class contains extension methods for classes (e.g. Vector3).
☑ FriendInfo	Used to store friend info and in which state and in which room.
☑ GamePropertyKey	Class for constants for "well known" properties used in Loadbalancing.
☑ IChatClientListener	Callback interface for Photon Chat .

	Contains callback app about update new ChatClient
▶ IConnectionCallbacks	Collection of "org the Realtime Api and Regions.
▶ IDisposable	
▶ PhotonPing PingMono	Uses C# Socket System.Net.Sockets does).
▶ IInRoomCallbacks	Collection of "in r Realtime Api to (or leaving, proper Client switching.
▶ ILobbyCallbacks	Collection of "org the Realtime Api
MonoBehaviourPunCallbacks	This class provid callbacks/events Override the eve use.
OnJoinedInstantiate	This component ' GameObject whe
SupportLogger	Helper class to d information about traffic statistics.
▶ IMatchmakingCallbacks	Collection of "org the Realtime Api
InstantiateParameters	
▶ IOnEventCallback	Event callback fo Covers events fr sent by clients vi
▶ IPhotonPeerListener	
ChatClient	Central class of t connect, handle (

<ul style="list-style-type: none"> <ul style="list-style-type: none"> LoadBalancingClient 	<p>This class implements LoadBalancing with LoadBalancingFactory will automatically create between the Master and Slave.</p>
<ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> IPointerClickHandler 	
<ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> IPointerDownHandler 	
<ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> ButtonInsideScrollList 	<p>Button inside scroll view. It is responsible for the scrollability of scrollRect when pressing down or dragging up and down scrolling. this does not affect scrollRect composition hierarchy.</p>
<ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> IPointerEnterHandler 	
<ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> IPointerExitHandler 	
<ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> GraphicToggleIsOnTransition 	<p>Use this on toggle to change color transition on the isOn State.</p>
<ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> OnPointerOverTooltip 	<p>Set focus to a given point is over</p>
<ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> TextButtonTransition 	<p>Use this on Button to change color transition on corrupting button</p>
<ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> TextToggleIsOnTransition 	<p>Use this on toggle to change color transition on the isOn State.</p>
<ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> IPointerUpHandler 	
<ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> IPunInstantiateMagicCallback 	
<ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> IPunObservable 	<p>Defines the OnPhotonSerializeData method to make it work correctly for observable.</p>
<ul style="list-style-type: none"> <ul style="list-style-type: none"> <ul style="list-style-type: none"> IPunOwnershipCallbacks 	<p>This interface is used to define callback methods for OnPhotonSerializeData.</p>

	implement them
►  IPunPrefabPool	Defines an interface used in PhotonNetwork. PhotonNetwork
 IPunTurnManagerCallbacks	
 IWebRpcCallback	Interface for "Web Realtime Api". Contains responses for Web
►  List	
 ConnectionCallbacksContainer	Container type for IConnectionCall LoadBalancingCa
 MatchMakingCallbacksContainer	Container type for IMatchmakingCa MatchMakingCal
►  MonoBehaviour	
 OperationCode	Class for constant codes. Pun uses internally.
 OpJoinRandomRoomParams	
 ParameterCode	Class for constant parameters of Op
 ParameterCode	Class for constant parameters of Op
 PhotonMessageInfo	Container class for message, RPC o
 PhotonNetwork	The main class to PhotonNetwork static.
►  PhotonPeer	
 ChatPeer	Provides basic of Chat server. This public ChatClien
 LoadBalancingPeer	A Loadbalancing

	operations and e to use the loadba application which Cloud.
 PhotonStream	This container is OnPhotonSeriali; incoming data of to provide it.
 PhotonStreamQueue	The PhotonStre object states at h what PhotonNet and then sends a when Serialize() receiving end you and then the stre received object s and timeStep the
 PhotonTransformViewPositionControl	
 PhotonTransformViewPositionModel	
 PhotonTransformViewRotationControl	
 PhotonTransformViewRotationModel	
 PhotonTransformViewScaleControl	
 PhotonTransformViewScaleModel	
 Player	Summarizes a "p identified (in that "actorNumber").
 PlayerNumberingExtensions	Extension used f and Player class.
 PunExtensions	Small number of make it easier fo Unity-versions.
 RaiseEventOptions	Aggregates seve options for opera field descriptions
 Region	

 RegionHandler	Provides method regions (Photon to find the one wi
 RegionPinger	
  RoomInfo	A simplified room required to list ar listing in the lobb settable (IsOpen,
 Room	This class repres joins/joined.
 RoomOptions	Wraps up comm needed when you individual entries
 SceneManagerHelper	
 ScoreExtensions	
  ScriptableObject	
 ServerSettings	Collection of con used internally by PhotonNetwork .
 PhotonAnimatorView.SynchronizedLayer	
 PhotonAnimatorView.SynchronizedParameter	
 TabViewManager.Tab	
 TeamExtensions	Extension used f class. Wraps acc custom property.
 TurnExtensions	
  TypedLobby	Refers to a speci the server.
 TypedLobbyInfo	
  UnityEvent	
 WebFlags	Optional flags to SDKs with Op R& SetProperties. In webhooks 1.2 to

WebRpcResponse

forwarded HTTP
Reads an operat
WebRpc and pro
to most common

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes										
Class List					Class Index					Class Hierarchy					Class Members										
All			Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			

Here is a list of all documented class members with links to the class documentation for each member:

- a -

- ActorList : **ParameterCode**
- ActorNr : **ParameterCode**
- ActorNumber : **Player**
- Add() : **ChatChannel** , **ParameterCode**
- AddAuthParameter() : **AuthenticationValues**
- AddCallbackTarget() : **PhotonNetwork** , **LoadBalancingClient**
- AddChild() : **CellTreeNode**
- AddFriends() : **ChatClient** , **ChatOperationCode**
- AddPlayer() : **Room**
- Address : **ParameterCode**
- AllocateSceneViewID() : **PhotonNetwork**
- AllocateViewID() : **PhotonNetwork**
- AlmostEquals() : **PunExtensions**
- AppId : **ChatClient** , **LoadBalancingClient**
- AppIdChat : **AppSettings**
- AppIdRealtime : **AppSettings**
- AppIdVoice : **AppSettings**
- ApplicationId : **ParameterCode**
- AppStats : **EventCode**
- AppVersion : **ChatClient** , **ParameterCode** , **PhotonNetwork** , **AppSettings** , **LoadBalancingClient** , **ParameterCode**
- Authenticate : **ChatOperationCode** , **OperationCode**
- AuthenticateOnce : **OperationCode**
- AuthenticateOnNameServer() : **ChatPeer**
- AuthenticationTicketExpired : **ErrorCode**

- AuthenticationValues() : **AuthenticationValues**
- AuthEvent : **EventCode**
- AuthGetParameters : **AuthenticationValues**
- AuthMode : **LoadBalancingClient**
- AuthPostData : **AuthenticationValues**
- AuthType : **AuthenticationValues**
- AuthValues : **ChatClient** , **PhotonNetwork** , **LoadBalancingClient**
- AutoCleanUp : **Room**
- autoCleanUp : **RoomInfo**
- AutoConnect : **ConnectAndJoinRandom**
- AutomaticallySyncScene : **PhotonNetwork**
- Away : **ChatUserStatus**
- AzureLocalNodeId : **ParameterCode**
- AzureMasterNodeId : **ParameterCode**
- AzureNodeInfo : **EventCode** , **ParameterCode**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes			
Class List				Class Index				Class Hierarchy				Class Members			
All				Functions				Variables				Enumerations			
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
q	r	s	t	u	v	w									

Here is a list of all documented class members with links to the class documentation for each member:

- b -

- BeginTurn() : **PunTurnManager**
- BestRegion : **RegionHandler**
- BestRegionSummaryInPreferences : **PhotonNetwork** , **ServerSettings**
- Broadcast : **ParameterCode**
- BroadcastPropsChangeToAll : **RoomOptions**
- buttonsOn : **PhotonStatsGui**



Photon Unity Networking 2 2.12

Main Page				Related Pages					Modules				Classes										
Class List				Class Index			Class Hierarchy				Class Members												
All		Functions			Variables			Enumerations				Properties				Events							
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

Here is a list of all documented class members with links to the class documentation for each member:

- C -

- Cache : **ParameterCode**
- CacheDiscreteTriggers() : **PhotonAnimatorView**
- CacheSliceChanged : **EventCode**
- CacheSliceIndex : **ParameterCode**
- CachingOption : **RaiseEventOptions**
- CanChat : **ChatClient**
- CanChatInChannel() : **ChatClient**
- CellTree() : **CellTree**
- CellTreeNode() : **CellTreeNode**
- Center : **CellTreeNode**
- ChangeGroups : **OperationCode**
- ChangeLocalID() : **LoadBalancingClient**
- Channel : **ChatParameterCode**
- ChannelHistory : **ChatOperationCode**
- Channels : **ChatParameterCode**
- ChannelSubscribers : **ChatParameterCode**
- ChannelUserCount : **ChatParameterCode**
- ChatChannel() : **ChatChannel**
- ChatClient() : **ChatClient**
- ChatMessages : **ChatEventCode**
- chatPeer : **ChatClient**
- ChatPeer() : **ChatPeer**
- ChatRegion : **ChatClient**
- CheckUserOnJoin : **ParameterCode**
- Childs : **CellTreeNode**

- CleanupCacheOnLeave : **GamePropertyKey** , **ParameterCode** , **RoomOptions**
- ClearExpectedUsers() : **Room**
- ClearMessages() : **ChatChannel**
- Client : **ConnectionHandler** , **SupportLogger**
- ClientAuthenticationData : **ParameterCode**
- ClientAuthenticationParams : **ParameterCode**
- ClientAuthenticationType : **ParameterCode**
- CloseConnection() : **PhotonNetwork**
- CloudRegion : **PhotonNetwork** , **LoadBalancingClient**
- Cluster : **Region**
- Code : **ParameterCode**
- Connect() : **ChatClient** , **ChatPeer** , **LoadBalancingClient**
- ConnectAndSetStatus() : **ChatClient**
- ConnectionCallbackTargets : **LoadBalancingClient**
- ConnectMethod : **PhotonNetwork**
- ConnectToBestCloudServer() : **PhotonNetwork**
- ConnectToMaster() : **PhotonNetwork**
- ConnectToNameServer() : **LoadBalancingClient**
- ConnectToRegion() : **PhotonNetwork**
- ConnectToRegionMaster() : **LoadBalancingClient**
- ConnectUsingSettings() : **PhotonNetwork**
- Contains() : **Extensions**
- Count : **PhotonStream**
- CountdownTimerHasExpired() : **CountdownTimer**
- CountOfPlayers : **PhotonNetwork**
- CountOfPlayersInRooms : **PhotonNetwork**
- CountOfPlayersOnMaster : **PhotonNetwork**
- CountOfRooms : **PhotonNetwork**
- CountSendAcksOnly : **ConnectionHandler**
- CrcCheckEnabled : **PhotonNetwork**
- CreateGame : **OperationCode**
- CreateRoom() : **PhotonNetwork**
- CurrentLobby : **PhotonNetwork** , **LoadBalancingClient**
- CurrentRoom : **PhotonNetwork** , **LoadBalancingClient**
- CurrentServerAddress : **LoadBalancingClient**
- CustomAuthenticationFailed : **ErrorCode**
- CustomEventContent : **ParameterCode**
- CustomInitData : **ParameterCode**
- CustomProperties : **Player** , **RoomInfo**

- CustomRoomProperties : **RoomOptions**
- CustomRoomPropertiesForLobby : **RoomOptions**

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page				Related Pages						Modules					Classes									
Class List				Class Index				Class Hierarchy					Class Members											
All		Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

Here is a list of all documented class members with links to the class documentation for each member:

- d -

- Data : **ParameterCode**
- DebugMessage : **WebRpcResponse**
- DebugOut : **ChatClient**
- DebugReturn() : **IClientListener** , **LoadBalancingClient**
- Default : **ChannelCreationOptions** , **RaiseEventOptions**
- DefaultMaxSubscribers : **ChatClient**
- DeleteNullProperties : **RoomOptions**
- Deserialize() : **PhotonStreamQueue**
- Destroy() : **DefaultPool** , **IPunPrefabPool** , **PhotonNetwork**
- DestroyAll() : **PhotonNetwork**
- DestroyPlayerObjects() : **PhotonNetwork**
- Disconnect() : **ChatClient** , **PhotonNetwork** , **LoadBalancingClient**
- DisconnectedCause : **ChatClient** , **LoadBalancingClient**
- DND : **ChatUserStatus**
- DoesLayerSynchronizeTypeExist() : **PhotonAnimatorView**
- DoesParameterSynchronizeTypeExist() : **PhotonAnimatorView**
- dontDestroyOnLoad : **PlayerNumbering**
- Draw() : **CellTreeNode**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes									
Class List					Class Index					Class Hierarchy					Class Members									
All		Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

Here is a list of all documented class members with links to the class documentation for each member:

- e -

- ElapsedTimeInTurn : **PunTurnManager**
- EmptyRoomTtl : **GamePropertyKey**
- EmptyRoomTTL : **ParameterCode**
- EmptyRoomTtl : **Room**
- emptyRoomTtl : **RoomInfo**
- EmptyRoomTtl : **RoomOptions**
- EnabledRegions : **RegionHandler**
- EnableLobbyStatistics : **PhotonNetwork** , **AppSettings** , **LoadBalancingClient**
- EncryptionData : **ParameterCode**
- EncryptionMode : **LoadBalancingClient** , **ParameterCode**
- Equals() : **Player** , **RoomInfo**
- ErrorInfo : **EventCode**
- EventForward : **ParameterCode**
- EventReceived : **LoadBalancingClient**
- EvFinalMove : **PunTurnManager**
- EvMove : **PunTurnManager**
- ExpectedProtocol : **LoadBalancingClient** , **ParameterCode**
- ExpectedUsers : **GamePropertyKey** , **Room**
- expectedUsers : **RoomInfo**
- ExpectedValues : **ParameterCode**
- ExternalHttpCallFailed : **ErrorCode**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes												
Class List					Class Index					Class Hierarchy					Class Members												
All		Functions					Variables					Enumerations					Properties					Events					
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w					

Here is a list of all documented class members with links to the class documentation for each member:

- f -

- [FetchServerTimestamp\(\)](#) : **PhotonNetwork**
- [FindFriends\(\)](#) : **PhotonNetwork** , **OperationCode**
- [FindFriendsRequestList](#) : **ParameterCode**
- [FindFriendsResponseOnlineList](#) : **ParameterCode**
- [FindFriendsResponseRoomIdList](#) : **ParameterCode**
- [FindGameObjectsWithComponent\(\)](#) : **PhotonNetwork**
- [FinishedTurnPropKey](#) : **TurnExtensions**
- [FIRST_GROUP_ID](#) : **CullArea**
- [FixedRegion](#) : **AppSettings**
- [Flags](#) : **RaiseEventOptions**
- [Friends](#) : **ChatParameterCode**
- [FriendsList](#) : **ChatEventCode**
- [FrontendAddress](#) : **ChatClient**



Photon Unity Networking 2 2.12

Main Page				Related Pages					Modules				Classes										
Class List				Class Index			Class Hierarchy				Class Members												
All		Functions			Variables			Enumerations				Properties				Events							
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

Here is a list of all documented class members with links to the class documentation for each member:

- g -

- GameClosed : **ErrorCode**
- GameCount : **ParameterCode**
- GameDoesNotExist : **ErrorCode**
- GameFull : **ErrorCode**
- GameIdAlreadyExists : **ErrorCode**
- GameList : **EventCode** , **ParameterCode**
- GameListUpdate : **EventCode**
- GameProperties : **ParameterCode**
- GameServerAddress : **LoadBalancingClient**
- GameVersion : **PhotonNetwork**
- Get() : **Player**
- GetActiveCells() : **CellTreeNode** , **CullArea**
- GetCustomRoomList() : **PhotonNetwork**
- GetExtrapolatedPositionOffset() : **PhotonTransformViewPositionControl**
- GetFinishedTurn() : **TurnExtensions**
- GetGameList : **OperationCode**
- GetHashCode() : **Player** , **RoomInfo**
- GetLayerSynchronizeType() : **PhotonAnimatorView**
- GetLobbyStats : **OperationCode**
- GetNetworkPosition() : **PhotonTransformViewPositionControl**
- GetNetworkRotation() : **PhotonTransformViewRotationControl**
- GetNetworkScale() : **PhotonTransformViewScaleControl**
- GetNext() : **Player**
- GetNextFor() : **Player**

- **GetParameterSynchronizeType() : PhotonAnimatorView**
- **GetPing() : PhotonNetwork**
- **GetPlayer() : Room**
- **GetPlayerFinishedTurn() : PunTurnManager**
- **GetPlayerNumber() : PlayerNumberingExtensions**
- **GetPrivateChannelNameByUser() : ChatClient**
- **GetProperties : OperationCode**
- **GetRegions : OperationCode**
- **GetSynchronizedLayers() : PhotonAnimatorView**
- **GetSynchronizedParameters() : PhotonAnimatorView**
- **GetTeam() : TeamExtensions**
- **GetTurn() : TurnExtensions**
- **GetTurnStart() : TurnExtensions**
- **Group : ParameterCode**



Photon Unity Networking 2 2.12

Main Page				Related Pages						Modules					Classes									
Class List				Class Index				Class Hierarchy					Class Members											
All		Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

Here is a list of all documented class members with links to the class documentation for each member:

- h -

- HasQueuedObjects() : **PhotonStreamQueue**
- healthStatsVisible : **PhotonStatsGui**
- HistoryLength : **ChatParameterCode**
- HoverColor : **TextButtonTransition**
- HoverOffColor : **TextToggleIsOnTransition**
- HoverOnColor : **TextToggleIsOnTransition**
- HttpForward : **WebFlags**
- HttpLimitReached : **ErrorCode**



Photon Unity Networking 2 2.12

Main Page				Related Pages					Modules					Classes										
Class List				Class Index				Class Hierarchy					Class Members											
All		Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

Here is a list of all documented class members with links to the class documentation for each member:

- i -

- Id : **CellTreeNode**
- Info : **ParameterCode**
- InLobby : **PhotonNetwork** , **LoadBalancingClient**
- InRoom : **PhotonNetwork** , **LoadBalancingClient**
- instance : **PlayerNumbering**
- Instantiate() : **DefaultPool** , **IPunPrefabPool**
- InstantiationData : **PhotonView**
- InterestGroup : **RaiseEventOptions**
- InternalCacheProperties() : **Player**
- InternalServerError : **ErrorCode**
- InvalidAuthentication : **ErrorCode**
- InvalidEncryptionParameters : **ErrorCode**
- InvalidOperation : **ErrorCode**
- InvalidOperationCode : **ErrorCode**
- InvalidRegion : **ErrorCode**
- Invisible : **ChatUserStatus**
- IsAppld() : **ServerSettings**
- IsBestRegion : **AppSettings**
- IsComingBack : **ParameterCode**
- IsCompletedByAll : **PunTurnManager**
- IsConnected : **PhotonNetwork** , **LoadBalancingClient**
- IsConnectedAndReady : **PhotonNetwork** , **LoadBalancingClient**
- IsDefaultNameServer : **AppSettings**
- IsDefaultPort : **AppSettings**
- IsFetchingFriendList : **LoadBalancingClient**

- IsFinishedByMe : **PunTurnManager**
- IsInactive : **ActorProperties** , **ParameterCode** , **Player**
- IsLocal : **Player**
- IsMasterClient : **PhotonNetwork** , **Player**
- IsMasterServerAddress : **AppSettings**
- IsMessageQueueRunning : **PhotonNetwork**
- IsMine : **PhotonView**
- IsOpen : **GamePropertyKey** , **Room**
- isOpen : **RoomInfo**
- IsOpen : **RoomInfo** , **RoomOptions**
- IsOver : **PunTurnManager**
- IsPointInsideCell() : **CellTreeNode**
- IsPointNearCell() : **CellTreeNode**
- IsPrivate : **ChatChannel**
- IsReading : **PhotonStream**
- IsSceneView : **PhotonView**
- IsUsingNameServer : **LoadBalancingClient**
- IsVisible : **GamePropertyKey** , **Room**
- isVisible : **RoomInfo**
- IsVisible : **RoomInfo** , **RoomOptions**
- IsWriting : **PhotonStream**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes									
Class List					Class Index					Class Hierarchy					Class Members									
All		Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

Here is a list of all documented class members with links to the class documentation for each member:

- j -

- Join : **EventCode** , **OperationCode**
- JoinFailedFoundActiveJoiner : **ErrorCode**
- JoinFailedFoundExcludedUserId : **ErrorCode**
- JoinFailedFoundInactiveJoiner : **ErrorCode**
- JoinFailedPeerAlreadyJoined : **ErrorCode**
- JoinFailedWithRejoinerNotFound : **ErrorCode**
- JoinGame : **OperationCode**
- JoinLobby() : **PhotonNetwork** , **OperationCode**
- JoinMode : **ParameterCode**
- JoinOrCreateRoom() : **PhotonNetwork**
- JoinRandomGame : **OperationCode**
- JoinRandomRoom() : **PhotonNetwork**
- JoinRoom() : **PhotonNetwork**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes											
Class List				Class Index				Class Hierarchy				Class Members											
All		Functions				Variables				Enumerations				Properties				Events					
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

Here is a list of all documented class members with links to the class documentation for each member:

- k -

- `KeepAliveInBackground` : **PhotonNetwork** , **ConnectionHandler**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes									
Class List					Class Index					Class Hierarchy					Class Members									
All		Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

Here is a list of all documented class members with links to the class documentation for each member:

- | -

- LastMsgId : **ChatChannel**
- Leave : **EventCode** , **OperationCode**
- LeaveLobby() : **PhotonNetwork** , **OperationCode**
- LeaveRoom() : **PhotonNetwork**
- LevelLoadingProgress : **PhotonNetwork**
- LFG : **ChatUserStatus**
- LoadBalancingClient() : **LoadBalancingClient** , **Room**
- LoadBalancingPeer : **LoadBalancingClient** , **LoadBalancingPeer**
- LoadLevel() : **PhotonNetwork**
- LobbyName : **ParameterCode**
- LobbyStats : **EventCode** , **ParameterCode**
- LobbyType : **ParameterCode**
- LocalPlayer : **PhotonNetwork** , **LoadBalancingClient**
- LogLevel : **PhotonNetwork**
- LogStats() : **SupportLogger**
- LogTrafficStats : **SupportLogger**



Photon Unity Networking 2 2.12

Main Page				Related Pages					Modules				Classes										
Class List				Class Index			Class Hierarchy				Class Members												
All		Functions			Variables			Enumerations				Properties				Events							
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

Here is a list of all documented class members with links to the class documentation for each member:

- m -

- MasterClient : **PhotonNetwork**
 - MasterClientId : **GamePropertyKey** , **ParameterCode** , **Room**
 - masterClientId : **RoomInfo**
 - MasterPeerCount : **ParameterCode**
 - MasterServerAddress : **LoadBalancingClient**
 - Match : **EventCode**
 - MatchMakingCallbackTargets : **LoadBalancingClient**
 - MatchMakingType : **ParameterCode**
 - MAX_VIEW_IDS : **PhotonNetwork**
 - MaxCcuReached : **ErrorCode**
 - MaxPlayers : **GamePropertyKey** , **Room** , **RoomInfo**
 - maxPlayers : **RoomInfo**
 - MaxPlayers : **RoomOptions**
 - MaxResendsBeforeDisconnect : **PhotonNetwork**
 - MaxSubscribers : **ChannelCreationOptions** , **ChatChannel**
 - Merge() : **Extensions**
 - MergeStringKeys() : **Extensions**
 - Message : **ChatParameterCode**
 - MessageCount : **ChatChannel**
 - MessageLimit : **ChatChannel** , **ChatClient**
 - Messages : **ChatChannel** , **ChatParameterCode**
 - Mode : **EncryptionDataParameters**
 - MsgId : **ChatParameterCode**
 - MsgIds : **ChatParameterCode**
-



Photon Unity Networking 2 2.12

Main Page				Related Pages					Modules				Classes										
Class List				Class Index			Class Hierarchy				Class Members												
All		Functions			Variables			Enumerations				Properties				Events							
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

Here is a list of all documented class members with links to the class documentation for each member:

- n -

- Name : **ChatChannel** , **Room**
- name : **RoomInfo**
- Name : **RoomInfo** , **TypedLobby** , **WebRpcResponse**
- NameServerAddress : **ChatClient** , **ChatPeer** , **LoadBalancingClient**
- NameServerHost : **ChatPeer** , **LoadBalancingClient**
- NameServerHttp : **ChatPeer** , **LoadBalancingClient**
- NetworkClientState : **PhotonNetwork**
- NetworkingClient : **PhotonNetwork**
- NetworkLogging : **AppSettings**
- NetworkStatisticsEnabled : **PhotonNetwork**
- NetworkStatisticsReset() : **PhotonNetwork**
- NetworkStatisticsToString() : **PhotonNetwork**
- NickName : **PhotonNetwork** , **LoadBalancingClient** , **ParameterCode** , **Player**
- NodeType : **CellTreeNode**
- NoRandomMatchFound : **ErrorCode**
- NormalColor : **TextButtonTransition**
- NormalOffColor : **TextToggleIsOnTransition**
- NormalOnColor : **TextToggleIsOnTransition**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes										
Class List					Class Index					Class Hierarchy					Class Members										
All			Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			

Here is a list of all documented class members with links to the class documentation for each member:

- O -

- **ObjectsInOneUpdate** : **PhotonNetwork**
- **Offline** : **ChatUserStatus**
- **OfflineMode** : **PhotonNetwork**
- **Ok** : **ErrorCode**
- **OnChatStateChange()** : **IChatClientListener**
- **OnConnected()** : **IChatClientListener** , **MonoBehaviourPunCallbacks** , **OnJoinedInstantiate** , **ConnectionCallbacksContainer** , **IConnectionCallbacks** , **SupportLogger**
- **OnConnectedToMaster()** : **MonoBehaviourPunCallbacks** , **ConnectAndJoinRandom** , **OnJoinedInstantiate** , **ConnectionCallbacksContainer** , **IConnectionCallbacks** , **SupportLogger**
- **OnCountdownTimerHasExpired** : **CountdownTimer**
- **OnCreatedRoom()** : **MonoBehaviourPunCallbacks** , **OnJoinedInstantiate** , **IMatchmakingCallbacks** , **MatchMakingCallbacksContainer** , **SupportLogger**
- **OnCreateRoomFailed()** : **MonoBehaviourPunCallbacks** , **OnJoinedInstantiate** , **IMatchmakingCallbacks** , **MatchMakingCallbacksContainer** , **SupportLogger**
- **OnCustomAuthenticationFailed()** : **MonoBehaviourPunCallbacks** , **OnJoinedInstantiate** , **ConnectionCallbacksContainer** , **IConnectionCallbacks** , **SupportLogger**
- **OnCustomAuthenticationResponse()** : **MonoBehaviourPunCallbacks** , **OnJoinedInstantiate** ,

**ConnectionCallbacksContainer , IConnectionCallbacks ,
SupportLogger**

- **OnDisconnected() : IChatClientListener ,
MonoBehaviourPunCallbacks , ConnectAndJoinRandom ,
OnJoinedInstantiate , ConnectionCallbacksContainer ,
IConnectionCallbacks , SupportLogger**
- **OnDrawGizmos() : CullArea**
- **OnEvent() : PunTurnManager , IOnEventCallback ,
LoadBalancingClient**
- **OnFriendListUpdate() : MonoBehaviourPunCallbacks ,
OnJoinedInstantiate , IMatchmakingCallbacks ,
MatchMakingCallbacksContainer , SupportLogger**
- **OnGetMessages() : IChatClientListener**
- **OnJoinedLobby() : MonoBehaviourPunCallbacks ,
ConnectAndJoinRandom , OnJoinedInstantiate ,
ILobbyCallbacks , SupportLogger**
- **OnJoinedRoom() : MonoBehaviourPunCallbacks ,
ConnectAndJoinRandom , OnJoinedInstantiate ,
PlayerNumbering , PunTeams , IMatchmakingCallbacks ,
MatchMakingCallbacksContainer , SupportLogger**
- **OnJoinRandomFailed() : MonoBehaviourPunCallbacks ,
ConnectAndJoinRandom , OnJoinedInstantiate ,
IMatchmakingCallbacks , MatchMakingCallbacksContainer ,
SupportLogger**
- **OnJoinRoomFailed() : MonoBehaviourPunCallbacks ,
OnJoinedInstantiate , IMatchmakingCallbacks ,
MatchMakingCallbacksContainer , SupportLogger**
- **OnLeftLobby() : MonoBehaviourPunCallbacks ,
OnJoinedInstantiate , ILobbyCallbacks , SupportLogger**
- **OnLeftRoom() : MonoBehaviourPunCallbacks ,
OnJoinedInstantiate , PlayerNumbering , PunTeams ,
IMatchmakingCallbacks , MatchMakingCallbacksContainer ,
SupportLogger**
- **Online : ChatUserStatus**
- **OnLobbyStatisticsUpdate() : MonoBehaviourPunCallbacks ,
OnJoinedInstantiate , ILobbyCallbacks , SupportLogger**
- **OnMasterClientSwitched() : MonoBehaviourPunCallbacks ,
IInRoomCallbacks , SupportLogger**
- **OnMessage() : LoadBalancingClient**
- **OnOperationResponse() : LoadBalancingClient**

- OnOwnershipRequest() : **IPunOwnershipCallbacks**
- OnOwnershipTransferred() : **IPunOwnershipCallbacks**
- OnPhotonSerializeView() : **IPunObservable** ,
PhotonAnimatorView , **PhotonRigidbody2DView** ,
PhotonRigidbodyView , **PhotonTransformView** ,
PhotonTransformViewClassic , **CullingHandler** ,
SmoothSyncMovement
- OnPlayerEnteredRoom() : **MonoBehaviourPunCallbacks** ,
PlayerNumbering , **PunTeams** , **INRoomCallbacks** ,
SupportLogger
- OnPlayerFinished() : **IPunTurnManagerCallbacks**
- OnPlayerLeftRoom() : **MonoBehaviourPunCallbacks** ,
PlayerNumbering , **PunTeams** , **INRoomCallbacks** ,
SupportLogger
- OnPlayerMove() : **IPunTurnManagerCallbacks**
- OnPlayerNumberingChanged : **PlayerNumbering**
- OnPlayerPropertiesUpdate() : **MonoBehaviourPunCallbacks** ,
PlayerNumbering , **PunTeams** , **INRoomCallbacks** ,
SupportLogger
- OnPrivateMessage() : **IChatClientListener**
- OnRegionListReceived() : **MonoBehaviourPunCallbacks** ,
OnJoinedInstantiate , **ConnectionCallbacksContainer** ,
IConnectionCallbacks , **SupportLogger**
- OnRoomListUpdate() : **MonoBehaviourPunCallbacks** ,
OnJoinedInstantiate , **ILobbyCallbacks** , **SupportLogger**
- OnRoomPropertiesUpdate() : **MonoBehaviourPunCallbacks** ,
CountdownTimer , **PunTurnManager** , **INRoomCallbacks** ,
SupportLogger
- OnStatusChanged() : **LoadBalancingClient**
- OnStatusUpdate() : **IChatClientListener**
- OnSubscribed() : **IChatClientListener**
- OnTabChanged : **TabViewManager**
- OnTurnBegins() : **IPunTurnManagerCallbacks**
- OnTurnCompleted() : **IPunTurnManagerCallbacks**
- OnTurnTimeEnds() : **IPunTurnManagerCallbacks**
- OnUnsubscribed() : **IChatClientListener**
- OnUserSubscribed() : **IChatClientListener**
- OnUserUnsubscribed() : **IChatClientListener**
- OnWebRpcResponse() : **IWebRpcCallback**
- OpAuthenticate() : **LoadBalancingPeer**

- **OpAuthenticateOnce() : LoadBalancingPeer**
- **OpChangeGroups() : LoadBalancingClient , LoadBalancingPeer**
- **OpCleanActorRpcBuffer() : PhotonNetwork**
- **OpCleanRpcBuffer() : PhotonNetwork**
- **OpCreateRoom() : LoadBalancingClient , LoadBalancingPeer**
- **OperationNotAllowedInCurrentState : ErrorCode**
- **OpFindFriends() : LoadBalancingClient , LoadBalancingPeer**
- **OpGetGameList() : LoadBalancingClient , LoadBalancingPeer**
- **OpJoinLobby() : LoadBalancingClient , LoadBalancingPeer**
- **OpJoinOrCreateRoom() : LoadBalancingClient**
- **OpJoinRandomRoom() : LoadBalancingClient , LoadBalancingPeer**
- **OpJoinRoom() : LoadBalancingClient , LoadBalancingPeer**
- **OpLeaveLobby() : LoadBalancingClient , LoadBalancingPeer**
- **OpLeaveRoom() : LoadBalancingClient , LoadBalancingPeer**
- **OpRaiseEvent() : LoadBalancingClient , LoadBalancingPeer**
- **OpRejoinRoom() : LoadBalancingClient**
- **OpRemoveCompleteCacheOfPlayer() : PhotonNetwork**
- **OpResponseReceived : LoadBalancingClient**
- **OpSetCustomPropertiesOfActor() : LoadBalancingClient**
- **OpSetCustomPropertiesOfRoom() : LoadBalancingClient**
- **OpSetPropertiesOfRoom() : LoadBalancingClient**
- **OpSettings() : LoadBalancingPeer**
- **OpWebRpc() : LoadBalancingClient**
- **Owner : PhotonView**
- **OwnershipTransfer : PhotonView**
- **OwnershipWasTransferred : PhotonView**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes										
Class List					Class Index					Class Hierarchy					Class Members										
All			Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			

Here is a list of all documented class members with links to the class documentation for each member:

- p -

- PacketLossByCrcCheck : **PhotonNetwork**
- Parameters : **WebRpcResponse**
- Parent : **CellTreeNode**
- PeekNext() : **PhotonStream**
- Peer : **PhotonLagSimulationGui**
- PeerCount : **ParameterCode**
- PhotonServerSettings : **PhotonNetwork**
- PhotonStream() : **PhotonStream**
- PhotonStreamQueue() : **PhotonStreamQueue**
- photonView : **MonoBehaviourPun**
- PhotonViews : **PhotonNetwork**
- PlayerCount : **GamePropertyKey** , **Room** , **RoomInfo**
- PlayerList : **PhotonNetwork**
- PlayerListOthers : **PhotonNetwork**
- PlayerName : **ActorProperties**
- PlayerNumberingChanged() : **PlayerNumbering**
- PlayerProperties : **ParameterCode**
- Players : **Room**
- PlayersInRoomsCount : **LoadBalancingClient**
- PlayersOnMasterCount : **LoadBalancingClient**
- PlayersPerTeam : **PunTeams**
- PlayerTtl : **GamePropertyKey**
- PlayerTTL : **ParameterCode**
- PlayerTtl : **Room**
- playerTtl : **RoomInfo**

- PlayerTtl : **RoomOptions**
- Playing : **ChatUserStatus**
- PluginMismatch : **ErrorCode**
- PluginName : **ParameterCode**
- PluginReportedError : **ErrorCode**
- Plugins : **ParameterCode** , **RoomOptions**
- PluginVersion : **ParameterCode**
- Port : **AppSettings**
- Position : **ParameterCode**
- PrecisionForFloatSynchronization : **PhotonNetwork**
- PrecisionForQuaternionSynchronization : **PhotonNetwork**
- PrecisionForVectorSynchronization : **PhotonNetwork**
- PrefabPool : **PhotonNetwork**
- PrivateChannels : **ChatClient**
- PrivateMessage : **ChatEventCode**
- Properties : **ChatParameterCode** , **ParameterCode**
- PropertiesChanged : **EventCode**
- PropertiesListedInLobby : **Room**
- propertiesListedInLobby : **RoomInfo**
- PropsListedInLobby : **GamePropertyKey**
- Protocol : **AppSettings**
- PublicChannels : **ChatClient**
- Publish : **ChatOperationCode**
- PublishMessage() : **ChatClient**
- PublishSubscribers : **ChannelCreationOptions** , **ChatChannel**
- PublishUserId : **ParameterCode** , **RoomOptions**
- PunVersion : **PhotonNetwork**



Photon Unity Networking 2 2.12

Main Page				Related Pages					Modules				Classes										
Class List				Class Index			Class Hierarchy				Class Members												
All		Functions			Variables			Enumerations				Properties				Events							
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

Here is a list of all documented class members with links to the class documentation for each member:

- **q** -

- QueueState : **EventCode**
- QuickResends : **PhotonNetwork**



Photon Unity Networking 2 2.12

Main Page				Related Pages					Modules				Classes										
Class List				Class Index			Class Hierarchy				Class Members												
All		Functions			Variables			Enumerations				Properties				Events							
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

Here is a list of all documented class members with links to the class documentation for each member:

- r -

- RaiseEvent() : **PhotonNetwork** , **OperationCode**
- RealtimeFallbackThread() : **ConnectionHandler**
- ReceiveNext() : **PhotonStream** , **PhotonStreamQueue**
- ReceiverGroup : **ParameterCode**
- Receivers : **RaiseEventOptions**
- Reconnect() : **PhotonNetwork**
- ReconnectAndRejoin() : **PhotonNetwork** , **LoadBalancingClient**
- ReconnectToMaster() : **LoadBalancingClient**
- RefreshData() : **PlayerNumbering**
- RefreshRpcMonoBehaviourCache() : **PhotonView**
- Region : **ParameterCode**
- RegionHandler : **LoadBalancingClient**
- RejoinRoom() : **PhotonNetwork**
- RemainingSecondsInTurn : **PunTurnManager**
- Remove : **ParameterCode**
- RemoveCallbackTarget() : **PhotonNetwork** , **LoadBalancingClient**
- Removed : **GamePropertyKey**
- RemovedFromList : **RoomInfo**
- RemoveFriends() : **ChatClient** , **ChatOperationCode**
- RemovePlayerCustomProperties() : **PhotonNetwork**
- RemoveRPCs() : **PhotonNetwork**
- RemoveRPCsInGroup() : **PhotonNetwork**
- RequestOwnership() : **PhotonView**
- ResentReliableCommands : **PhotonNetwork**

- **Reset() : PhotonStreamQueue**
- **ResetBestRegionCodeInPreferences() : ServerSettings**
- **ResolveHost() : RegionPinger**
- **ResourceCache : DefaultPool**
- **ReturnCode : WebRpcResponse**
- **Room() : Room**
- **RoomName : ParameterCode**
- **RoomOptionFlags : ParameterCode**
- **RoomPlayerIndexedProp : PlayerNumbering**
- **RoomsCount : LoadBalancingClient**
- **RootNode : CellTree**
- **RPC() : PhotonView**
- **RpcSecure() : PhotonView**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes												
Class List					Class Index					Class Hierarchy					Class Members												
All		Functions					Variables					Enumerations					Properties					Events					
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w					

Here is a list of all documented class members with links to the class documentation for each member:

- S -

- Secret : **ChatParameterCode** , **ParameterCode**
- Secret1 : **EncryptionDataParameters**
- Secret2 : **EncryptionDataParameters**
- Selectable : **TextButtonTransition**
- SelectTab() : **TabViewManager**
- SendAcksOnly() : **ChatClient**
- SendAllOutgoingCommands() : **PhotonNetwork**
- SendAuthCookie : **WebFlags**
- Sender : **ChatParameterCode** , **PhotonMessageInfo**
- Senders : **ChatChannel** , **ChatParameterCode**
- SendMove() : **PunTurnManager**
- SendNext() : **PhotonStream** , **PhotonStreamQueue**
- SendPrivate : **ChatOperationCode**
- SendPrivateMessage() : **ChatClient**
- SendRate : **PhotonNetwork**
- SendState : **WebFlags**
- SendSync : **WebFlags**
- SequenceChannel : **RaiseEventOptions**
- SerializationRate : **PhotonNetwork**
- Serialize() : **PhotonStream** , **PhotonStreamQueue**
- Server : **PhotonNetwork** , **AppSettings** , **LoadBalancingClient**
- ServerAddress : **PhotonNetwork**
- ServerFull : **ErrorCode**
- ServerSettings : **OperationCode**
- ServerTimestamp : **PhotonNetwork**

- Service() : **ChatClient** , **LoadBalancingClient**
- SetAuthPostData() : **AuthenticationValues**
- SetCustomProperties() : **Player** , **Room**
- SetFinishedTurn() : **TurnExtensions**
- SetInterestGroups() : **PhotonNetwork**
- SetLayerSynchronized() : **PhotonAnimatorView**
- SetLevelPrefix() : **PhotonNetwork**
- SetMasterClient() : **PhotonNetwork** , **Room**
- SetOnlineStatus() : **ChatClient**
- SetParameterSynchronized() : **PhotonAnimatorView**
- SetPlayerCustomProperties() : **PhotonNetwork**
- SetPlayerNumber() : **PlayerNumberingExtensions**
- SetProperties : **EventCode** , **OperationCode**
- SetPropertiesListedInLobby() : **Room**
- SetSendingEnabled() : **PhotonNetwork**
- SetSynchronizedValues() : **PhotonTransformViewClassic** , **PhotonTransformViewPositionControl**
- SetTeam() : **TeamExtensions**
- SetTurn() : **TurnExtensions**
- SkipMessage : **ChatParameterCode**
- SlotError : **ErrorCode**
- SocketImplementationConfig : **ChatClient**
- StartPing() : **PingMono**
- State : **ChatClient** , **LoadBalancingClient**
- StateChanged : **LoadBalancingClient**
- statsOn : **PhotonStatsGui**
- statsRect : **PhotonStatsGui**
- statsWindowOn : **PhotonStatsGui**
- Status : **ChatParameterCode**
- StatusUpdate : **ChatEventCode**
- StopThread() : **ChatClient**
- StorePlayer() : **Room**
- StripKeysWithNullValues() : **Extensions**
- StripToStringKeys() : **Extensions**
- SUBDIVISION_FIRST_LEVEL_ORDER : **CullArea**
- SUBDIVISION_SECOND_LEVEL_ORDER : **CullArea**
- SUBDIVISION_THIRD_LEVEL_ORDER : **CullArea**
- Subscribe() : **ChatClient** , **ChatEventCode** , **ChatOperationCode**
- SubscribeResults : **ChatParameterCode**
- Subscribers : **ChatChannel**

- SummaryToCache : **RegionHandler**
- SuppressRoomEvents : **ParameterCode** , **RoomOptions**

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page				Related Pages					Modules				Classes										
Class List				Class Index			Class Hierarchy				Class Members												
All		Functions			Variables			Enumerations				Properties				Events							
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

Here is a list of all documented class members with links to the class documentation for each member:

- t -

- Tabs : **TabViewManager**
- TagObject : **Player**
- TargetActorNr : **ParameterCode**
- TargetActors : **RaiseEventOptions**
- Team : **PunTeams**
- TeamPlayerProp : **PunTeams**
- Time : **PhotonNetwork**
- ToArray() : **PhotonStream**
- toggle : **TextToggleIsOnTransition**
- ToggleGroup : **TabViewManager**
- Token : **AuthenticationValues**
- ToString() : **AuthenticationValues** , **ServerSettings** , **Player** , **Room** , **RoomInfo**
- ToStringFull() : **AppSettings** , **Extensions** , **Player** , **Room** , **RoomInfo** , **WebRpcResponse**
- ToStringFull< T >() : **Extensions**
- ToStringMessages() : **ChatChannel**
- trafficStatsOn : **PhotonStatsGui**
- TransferOwnership() : **PhotonView**
- TransportProtocol : **ChatClient**
- TruncateMessages() : **ChatChannel**
- TryGetChannel() : **ChatClient**
- Turn : **PunTurnManager**
- TurnDuration : **PunTurnManager**
- TurnManagerEventOffset : **PunTurnManager**

- TurnManagerListener : **PunTurnManager**
- TurnPropKey : **TurnExtensions**
- TurnStartPropKey : **TurnExtensions**
- Type : **TypedLobby**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes										
Class List					Class Index					Class Hierarchy					Class Members										
All			Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			

Here is a list of all documented class members with links to the class documentation for each member:

- u -

- **Unsubscribe()** : **ChatClient** , **ChatEventCode** , **ChatOperationCode**
- **Update()** : **PhotonStatsGui**
- **UpdatePosition()** : **PhotonTransformViewPositionControl**
- **UpdateStatus** : **ChatOperationCode**
- **UriPath** : **ParameterCode**
- **UseAlternativeUdpPorts** : **PhotonNetwork** , **LoadBalancingClient**
- **UseBackgroundWorkerForSending** : **ChatClient**
- **UseCloud()** : **ServerSettings**
- **UseNameServer** : **AppSettings**
- **UserBlocked** : **ErrorCode**
- **UserId** : **AuthenticationValues** , **ChatClient** , **ChatParameterCode** , **ParameterCode** , **ActorProperties** , **AuthenticationValues** , **LoadBalancingClient** , **ParameterCode** , **Player**
- **UseRpcMonoBehaviourCache** : **PhotonNetwork**
- **Users** : **ChatEventCode**
- **UserSubscribed** : **ChatEventCode**
- **UserUnsubscribed** : **ChatEventCode**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes											
Class List				Class Index				Class Hierarchy				Class Members											
All		Functions				Variables				Enumerations				Properties				Events					
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

Here is a list of all documented class members with links to the class documentation for each member:

- v -

- Version : **ConnectAndJoinRandom**
- ViewID : **PhotonView**
- Visible : **PhotonLagSimulationGui**



Photon Unity Networking 2 2.12

Main Page				Related Pages					Modules				Classes										
Class List				Class Index			Class Hierarchy				Class Members												
All		Functions			Variables			Enumerations				Properties				Events							
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

Here is a list of all documented class members with links to the class documentation for each member:

- W -

- WebFlags : **ChatParameterCode**
- WebRpc() : **PhotonNetwork** , **OperationCode**
- WebRpcParameters : **ParameterCode**
- WebRpcResponse() : **WebRpcResponse**
- WebRpcReturnCode : **ParameterCode**
- WebRpcReturnMessage : **ParameterCode**
- WindowId : **PhotonLagSimulationGui** , **PhotonStatsGui**
- WindowRect : **PhotonLagSimulationGui**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes									
Class List					Class Index					Class Hierarchy					Class Members									
All		Functions					Variables					Enumerations					Properties					Events		
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w					

- a -

- Add() : **ChatChannel**
- AddAuthParameter() : **AuthenticationValues**
- AddCallbackTarget() : **PhotonNetwork** , **LoadBalancingClient**
- AddChild() : **CellTreeNode**
- AddFriends() : **ChatClient**
- AddPlayer() : **Room**
- AllocateSceneViewID() : **PhotonNetwork**
- AllocateViewID() : **PhotonNetwork**
- AlmostEquals() : **PunExtensions**
- AuthenticateOnNameServer() : **ChatPeer**
- AuthenticationValues() : **AuthenticationValues**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes							
Class List				Class Index				Class Hierarchy				Class Members							
All		Functions			Variables			Enumerations				Properties				Events			
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w

- b -

- **BeginTurn()** : **PunTurnManager**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes									
Class List				Class Index				Class Hierarchy				Class Members									
All		Functions				Variables				Enumerations				Properties				Events			
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w		

- C -

- `CacheDiscreteTriggers()` : **PhotonAnimatorView**
- `CanChatInChannel()` : **ChatClient**
- `CellTree()` : **CellTree**
- `CellTreeNode()` : **CellTreeNode**
- `ChangeLocalID()` : **LoadBalancingClient**
- `ChatChannel()` : **ChatChannel**
- `ChatClient()` : **ChatClient**
- `ChatPeer()` : **ChatPeer**
- `ClearExpectedUsers()` : **Room**
- `ClearMessages()` : **ChatChannel**
- `CloseConnection()` : **PhotonNetwork**
- `Connect()` : **ChatClient** , **ChatPeer** , **LoadBalancingClient**
- `ConnectAndSetStatus()` : **ChatClient**
- `ConnectToBestCloudServer()` : **PhotonNetwork**
- `ConnectToMaster()` : **PhotonNetwork**
- `ConnectToNameServer()` : **LoadBalancingClient**
- `ConnectToRegion()` : **PhotonNetwork**
- `ConnectToRegionMaster()` : **LoadBalancingClient**
- `ConnectUsingSettings()` : **PhotonNetwork**
- `Contains()` : **Extensions**
- `CountdownTimerHasExpired()` : **CountdownTimer**
- `CreateRoom()` : **PhotonNetwork**



Photon Unity Networking 2 2.12

Main Page				Related Pages						Modules						Classes											
Class List				Class Index				Class Hierarchy						Class Members													
All		Functions				Variables				Enumerations						Properties						Events					
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w								

- d -

- DebugReturn() : **IChatClientListener** , **LoadBalancingClient**
- Deserialize() : **PhotonStreamQueue**
- Destroy() : **DefaultPool** , **IPunPrefabPool** , **PhotonNetwork**
- DestroyAll() : **PhotonNetwork**
- DestroyPlayerObjects() : **PhotonNetwork**
- Disconnect() : **ChatClient** , **PhotonNetwork** , **LoadBalancingClient**
- DoesLayerSynchronizeTypeExist() : **PhotonAnimatorView**
- DoesParameterSynchronizeTypeExist() : **PhotonAnimatorView**
- Draw() : **CellTreeNode**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes							
Class List				Class Index				Class Hierarchy				Class Members							
All		Functions			Variables			Enumerations				Properties				Events			
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w

- e -

- Equals() : **Player** , **RoomInfo**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes																
Class List					Class Index					Class Hierarchy					Class Members																
All		Functions					Variables					Enumerations					Properties					Events									
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w												

- f -

- `FetchServerTimestamp()` : **PhotonNetwork**
- `FindFriends()` : **PhotonNetwork**
- `FindGameObjectsWithComponent()` : **PhotonNetwork**



Photon Unity Networking 2 2.12

Main Page						Related Pages						Modules						Classes											
Class List						Class Index						Class Hierarchy						Class Members											
All			Functions				Variables				Enumerations						Properties						Events						
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z				

- g -

- **Get() : Player**
- **GetActiveCells() : CellTreeNode , CullArea**
- **GetCustomRoomList() : PhotonNetwork**
- **GetExtrapolatedPositionOffset() : PhotonTransformViewPositionControl**
- **GetFinishedTurn() : TurnExtensions**
- **GetHashCode() : Player , RoomInfo**
- **GetLayerSynchronizeType() : PhotonAnimatorView**
- **GetNetworkPosition() : PhotonTransformViewPositionControl**
- **GetNetworkRotation() : PhotonTransformViewRotationControl**
- **GetNetworkScale() : PhotonTransformViewScaleControl**
- **GetNext() : Player**
- **GetNextFor() : Player**
- **GetParameterSynchronizeType() : PhotonAnimatorView**
- **GetPing() : PhotonNetwork**
- **GetPlayer() : Room**
- **GetPlayerFinishedTurn() : PunTurnManager**
- **GetPlayerNumber() : PlayerNumberingExtensions**
- **GetPrivateChannelNameByUser() : ChatClient**
- **GetSynchronizedLayers() : PhotonAnimatorView**
- **GetSynchronizedParameters() : PhotonAnimatorView**
- **GetTeam() : TeamExtensions**
- **GetTurn() : TurnExtensions**
- **GetTurnStart() : TurnExtensions**

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes									
Class List				Class Index				Class Hierarchy				Class Members									
All		Functions				Variables				Enumerations				Properties				Events			
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w		

- h -

- HasQueuedObjects() : **PhotonStreamQueue**



Photon Unity Networking 2 2.12

Main Page				Related Pages					Modules					Classes									
Class List				Class Index				Class Hierarchy					Class Members										
All		Functions			Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w				

- i -

- Instantiate() : **DefaultPool** , **IPunPrefabPool**
- InternalCacheProperties() : **Player**
- IsAppld() : **ServerSettings**
- IsPointInsideCell() : **CellTreeNode**
- IsPointNearCell() : **CellTreeNode**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes							
Class List				Class Index				Class Hierarchy				Class Members							
All		Functions			Variables			Enumerations				Properties				Events			
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w

- j -

- `JoinLobby()` : **PhotonNetwork**
- `JoinOrCreateRoom()` : **PhotonNetwork**
- `JoinRandomRoom()` : **PhotonNetwork**
- `JoinRoom()` : **PhotonNetwork**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes							
Class List				Class Index				Class Hierarchy				Class Members							
All				Functions				Variables				Enumerations				Properties			
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w

- | -

- **LeaveLobby() : PhotonNetwork**
- **LeaveRoom() : PhotonNetwork**
- **LoadBalancingClient() : LoadBalancingClient**
- **LoadBalancingPeer() : LoadBalancingPeer**
- **LoadLevel() : PhotonNetwork**
- **LogStats() : SupportLogger**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes							
Class List				Class Index				Class Hierarchy				Class Members							
All		Functions			Variables			Enumerations				Properties				Events			
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w

- m -

- Merge() : **Extensions**
- MergeStringKeys() : **Extensions**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes							
Class List				Class Index				Class Hierarchy				Class Members							
All		Functions			Variables			Enumerations				Properties				Events			
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w

- n -

- NetworkStatisticsReset() : **PhotonNetwork**
- NetworkStatisticsToString() : **PhotonNetwork**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes											
Class List				Class Index				Class Hierarchy				Class Members											
All		Functions				Variables				Enumerations				Properties				Events					
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w				

- 0 -

- OnChatStateChange() : IChatClientListener
- OnConnected() : IChatClientListener , MonoBehaviourPunCallbacks , OnJoinedInstantiate , ConnectionCallbacksContainer , IConnectionCallbacks , SupportLogger
- OnConnectedToMaster() : MonoBehaviourPunCallbacks , ConnectAndJoinRandom , OnJoinedInstantiate , ConnectionCallbacksContainer , IConnectionCallbacks , SupportLogger
- OnCreatedRoom() : MonoBehaviourPunCallbacks , OnJoinedInstantiate , IMatchmakingCallbacks , MatchMakingCallbacksContainer , SupportLogger
- OnCreateRoomFailed() : MonoBehaviourPunCallbacks , OnJoinedInstantiate , IMatchmakingCallbacks , MatchMakingCallbacksContainer , SupportLogger
- OnCustomAuthenticationFailed() : MonoBehaviourPunCallbacks , OnJoinedInstantiate , ConnectionCallbacksContainer , IConnectionCallbacks , SupportLogger
- OnCustomAuthenticationResponse() : MonoBehaviourPunCallbacks , OnJoinedInstantiate , ConnectionCallbacksContainer , IConnectionCallbacks , SupportLogger
- OnDisconnected() : IChatClientListener , MonoBehaviourPunCallbacks , ConnectAndJoinRandom , OnJoinedInstantiate , ConnectionCallbacksContainer , IConnectionCallbacks , SupportLogger

- **OnDrawGizmos() : CullArea**
- **OnEvent() : PunTurnManager , IOnEventCallback , LoadBalancingClient**
- **OnFriendListUpdate() : MonoBehaviourPunCallbacks , OnJoinedInstantiate , IMatchmakingCallbacks , MatchMakingCallbacksContainer , SupportLogger**
- **OnGetMessages() : IChatClientListener**
- **OnJoinedLobby() : MonoBehaviourPunCallbacks , ConnectAndJoinRandom , OnJoinedInstantiate , ILobbyCallbacks , SupportLogger**
- **OnJoinedRoom() : MonoBehaviourPunCallbacks , ConnectAndJoinRandom , OnJoinedInstantiate , PlayerNumbering , PunTeams , IMatchmakingCallbacks , MatchMakingCallbacksContainer , SupportLogger**
- **OnJoinRandomFailed() : MonoBehaviourPunCallbacks , ConnectAndJoinRandom , OnJoinedInstantiate , IMatchmakingCallbacks , MatchMakingCallbacksContainer , SupportLogger**
- **OnJoinRoomFailed() : MonoBehaviourPunCallbacks , OnJoinedInstantiate , IMatchmakingCallbacks , MatchMakingCallbacksContainer , SupportLogger**
- **OnLeftLobby() : MonoBehaviourPunCallbacks , OnJoinedInstantiate , ILobbyCallbacks , SupportLogger**
- **OnLeftRoom() : MonoBehaviourPunCallbacks , OnJoinedInstantiate , PlayerNumbering , PunTeams , IMatchmakingCallbacks , MatchMakingCallbacksContainer , SupportLogger**
- **OnLobbyStatisticsUpdate() : MonoBehaviourPunCallbacks , OnJoinedInstantiate , ILobbyCallbacks , SupportLogger**
- **OnMasterClientSwitched() : MonoBehaviourPunCallbacks , IInRoomCallbacks , SupportLogger**
- **OnMessage() : LoadBalancingClient**
- **OnOperationResponse() : LoadBalancingClient**
- **OnOwnershipRequest() : IPunOwnershipCallbacks**
- **OnOwnershipTransferred() : IPunOwnershipCallbacks**
- **OnPhotonSerializeView() : IPunObservable , PhotonAnimatorView , PhotonRigidbody2DView , PhotonRigidbodyView , PhotonTransformView , PhotonTransformViewClassic , CullingHandler , SmoothSyncMovement**

- **OnPlayerEnteredRoom() : MonoBehaviourPunCallbacks , PlayerNumbering , PunTeams , IInRoomCallbacks , SupportLogger**
- **OnPlayerFinished() : IPunTurnManagerCallbacks**
- **OnPlayerLeftRoom() : MonoBehaviourPunCallbacks , PlayerNumbering , PunTeams , IInRoomCallbacks , SupportLogger**
- **OnPlayerMove() : IPunTurnManagerCallbacks**
- **OnPlayerPropertiesUpdate() : MonoBehaviourPunCallbacks , PlayerNumbering , PunTeams , IInRoomCallbacks , SupportLogger**
- **OnPrivateMessage() : IChatClientListener**
- **OnRegionListReceived() : MonoBehaviourPunCallbacks , OnJoinedInstantiate , ConnectionCallbacksContainer , IConnectionCallbacks , SupportLogger**
- **OnRoomListUpdate() : MonoBehaviourPunCallbacks , OnJoinedInstantiate , ILobbyCallbacks , SupportLogger**
- **OnRoomPropertiesUpdate() : MonoBehaviourPunCallbacks , CountdownTimer , PunTurnManager , IInRoomCallbacks , SupportLogger**
- **OnStatusChanged() : LoadBalancingClient**
- **OnStatusUpdate() : IChatClientListener**
- **OnSubscribed() : IChatClientListener**
- **OnTurnBegins() : IPunTurnManagerCallbacks**
- **OnTurnCompleted() : IPunTurnManagerCallbacks**
- **OnTurnTimeEnds() : IPunTurnManagerCallbacks**
- **OnUnsubscribed() : IChatClientListener**
- **OnUserSubscribed() : IChatClientListener**
- **OnUserUnsubscribed() : IChatClientListener**
- **OnWebRpcResponse() : IWebRpcCallback**
- **OpAuthenticate() : LoadBalancingPeer**
- **OpAuthenticateOnce() : LoadBalancingPeer**
- **OpChangeGroups() : LoadBalancingClient , LoadBalancingPeer**
- **OpCleanActorRpcBuffer() : PhotonNetwork**
- **OpCleanRpcBuffer() : PhotonNetwork**
- **OpCreateRoom() : LoadBalancingClient , LoadBalancingPeer**
- **OpFindFriends() : LoadBalancingClient , LoadBalancingPeer**
- **OpGetGameList() : LoadBalancingClient , LoadBalancingPeer**
- **OpJoinLobby() : LoadBalancingClient , LoadBalancingPeer**

- OpJoinOrCreateRoom() : **LoadBalancingClient**
- OpJoinRandomRoom() : **LoadBalancingClient** , **LoadBalancingPeer**
- OpJoinRoom() : **LoadBalancingClient** , **LoadBalancingPeer**
- OpLeaveLobby() : **LoadBalancingClient** , **LoadBalancingPeer**
- OpLeaveRoom() : **LoadBalancingClient** , **LoadBalancingPeer**
- OpRaiseEvent() : **LoadBalancingClient** , **LoadBalancingPeer**
- OpRejoinRoom() : **LoadBalancingClient**
- OpRemoveCompleteCacheOfPlayer() : **PhotonNetwork**
- OpSetCustomPropertiesOfActor() : **LoadBalancingClient**
- OpSetCustomPropertiesOfRoom() : **LoadBalancingClient**
- OpSetPropertiesOfRoom() : **LoadBalancingClient**
- OpSettings() : **LoadBalancingPeer**
- OpWebRpc() : **LoadBalancingClient**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes							
Class List				Class Index				Class Hierarchy				Class Members							
All		Functions			Variables			Enumerations			Properties			Events					
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w

- p -

- PeekNext() : **PhotonStream**
- PhotonStream() : **PhotonStream**
- PhotonStreamQueue() : **PhotonStreamQueue**
- PlayerNumberingChanged() : **PlayerNumbering**
- PublishMessage() : **ChatClient**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes										
Class List					Class Index					Class Hierarchy					Class Members										
All			Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w						

- r -

- RaiseEvent() : **PhotonNetwork**
- RealtimeFallbackThread() : **ConnectionHandler**
- ReceiveNext() : **PhotonStream** , **PhotonStreamQueue**
- Reconnect() : **PhotonNetwork**
- ReconnectAndRejoin() : **PhotonNetwork** , **LoadBalancingClient**
- ReconnectToMaster() : **LoadBalancingClient**
- RefreshData() : **PlayerNumbering**
- RefreshRpcMonoBehaviourCache() : **PhotonView**
- RejoinRoom() : **PhotonNetwork**
- RemoveCallbackTarget() : **PhotonNetwork** , **LoadBalancingClient**
- RemoveFriends() : **ChatClient**
- RemovePlayerCustomProperties() : **PhotonNetwork**
- RemoveRPCs() : **PhotonNetwork**
- RemoveRPCsInGroup() : **PhotonNetwork**
- RequestOwnership() : **PhotonView**
- Reset() : **PhotonStreamQueue**
- ResetBestRegionCodeInPreferences() : **ServerSettings**
- ResolveHost() : **RegionPinger**
- Room() : **Room**
- RPC() : **PhotonView**
- RpcSecure() : **PhotonView**



Photon Unity Networking 2 2.12

Main Page				Related Pages					Modules					Classes									
Class List				Class Index				Class Hierarchy					Class Members										
All		Functions			Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w				

- S -

- SelectTab() : **TabViewManager**
- SendAcksOnly() : **ChatClient**
- SendAllOutgoingCommands() : **PhotonNetwork**
- SendMove() : **PunTurnManager**
- SendNext() : **PhotonStream** , **PhotonStreamQueue**
- SendPrivateMessage() : **ChatClient**
- Serialize() : **PhotonStream** , **PhotonStreamQueue**
- Service() : **ChatClient** , **LoadBalancingClient**
- SetAuthPostData() : **AuthenticationValues**
- SetCustomProperties() : **Player** , **Room**
- SetFinishedTurn() : **TurnExtensions**
- SetInterestGroups() : **PhotonNetwork**
- SetLayerSynchronized() : **PhotonAnimatorView**
- SetLevelPrefix() : **PhotonNetwork**
- SetMasterClient() : **PhotonNetwork** , **Room**
- SetOnlineStatus() : **ChatClient**
- SetParameterSynchronized() : **PhotonAnimatorView**
- SetPlayerCustomProperties() : **PhotonNetwork**
- SetPlayerNumber() : **PlayerNumberingExtensions**
- SetPropertiesListedInLobby() : **Room**
- SetSendingEnabled() : **PhotonNetwork**
- SetSynchronizedValues() : **PhotonTransformViewClassic** , **PhotonTransformViewPositionControl**
- SetTeam() : **TeamExtensions**
- SetTurn() : **TurnExtensions**
- StartPing() : **PingMono**

- `StopThread()` : **ChatClient**
- `StorePlayer()` : **Room**
- `StripKeysWithNullValues()` : **Extensions**
- `StripToStringKeys()` : **Extensions**
- `Subscribe()` : **ChatClient**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes							
Class List				Class Index				Class Hierarchy				Class Members							
All		Functions			Variables			Enumerations			Properties			Events					
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w

- t -

- **ToArray() : PhotonStream**
- **ToString() : AuthenticationValues , ServerSettings , Player , Room , RoomInfo**
- **ToStringFull() : AppSettings , Extensions , Player , Room , RoomInfo , WebRpcResponse**
- **ToStringFull< T >() : Extensions**
- **ToStringMessages() : ChatChannel**
- **TransferOwnership() : PhotonView**
- **TruncateMessages() : ChatChannel**
- **TryGetChannel() : ChatClient**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes							
Class List				Class Index				Class Hierarchy				Class Members							
All		Functions			Variables			Enumerations			Properties			Events					
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w

- u -

- `Unsubscribe()` : **ChatClient**
- `Update()` : **PhotonStatsGui**
- `UpdatePosition()` : **PhotonTransformViewPositionControl**
- `UseCloud()` : **ServerSettings**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes							
Class List				Class Index				Class Hierarchy				Class Members							
All		Functions			Variables			Enumerations				Properties				Events			
a	b	c	d	e	f	g	h	i	j	l	m	n	o	p	r	s	t	u	w

- W -

- WebRpc() : **PhotonNetwork**
- WebRpcResponse() : **WebRpcResponse**

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes									
Class List					Class Index					Class Hierarchy					Class Members									
All		Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

- a -

- ActorList : **ParameterCode**
- ActorNr : **ParameterCode**
- Add : **ParameterCode**
- AddFriends : **ChatOperationCode**
- Address : **ParameterCode**
- AppIdChat : **AppSettings**
- AppIdRealtime : **AppSettings**
- AppIdVoice : **AppSettings**
- ApplicationId : **ParameterCode**
- AppStats : **EventCode**
- AppVersion : **ParameterCode** , **AppSettings** , **ParameterCode**
- Authenticate : **ChatOperationCode** , **OperationCode**
- AuthenticateOnce : **OperationCode**
- AuthenticationTicketExpired : **ErrorCode**
- AuthEvent : **EventCode**
- AuthMode : **LoadBalancingClient**
- autoCleanUp : **RoomInfo**
- AutoConnect : **ConnectAndJoinRandom**
- Away : **ChatUserStatus**
- AzureLocalNodeId : **ParameterCode**
- AzureMasterNodeId : **ParameterCode**
- AzureNodeInfo : **EventCode** , **ParameterCode**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes									
Class List					Class Index					Class Hierarchy					Class Members									
All		Functions			Variables			Enumerations					Properties					Events						
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

- b -

- Broadcast : **ParameterCode**
- buttonsOn : **PhotonStatsGui**



Photon Unity Networking 2 2.12

Main Page				Related Pages						Modules					Classes									
Class List				Class Index				Class Hierarchy					Class Members											
All		Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

- C -

- Cache : **ParameterCode**
- CacheSliceChanged : **EventCode**
- CacheSliceIndex : **ParameterCode**
- CachingOption : **RaiseEventOptions**
- Center : **CellTreeNode**
- ChangeGroups : **OperationCode**
- Channel : **ChatParameterCode**
- ChannelHistory : **ChatOperationCode**
- Channels : **ChatParameterCode**
- ChannelSubscribers : **ChatParameterCode**
- ChannelUserCount : **ChatParameterCode**
- ChatMessages : **ChatEventCode**
- chatPeer : **ChatClient**
- CheckUserOnJoin : **ParameterCode**
- Childs : **CellTreeNode**
- CleanupCacheOnLeave : **GamePropertyKey** , **ParameterCode**
- ClientAuthenticationData : **ParameterCode**
- ClientAuthenticationParams : **ParameterCode**
- ClientAuthenticationType : **ParameterCode**
- Code : **ParameterCode**
- ConnectionCallbackTargets : **LoadBalancingClient**
- ConnectMethod : **PhotonNetwork**
- CreateGame : **OperationCode**
- CustomAuthenticationFailed : **ErrorCode**
- CustomEventContent : **ParameterCode**
- CustomInitData : **ParameterCode**

- CustomRoomProperties : **RoomOptions**
- CustomRoomPropertiesForLobby : **RoomOptions**

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page				Related Pages						Modules					Classes									
Class List				Class Index				Class Hierarchy					Class Members											
All		Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

- d -

- Data : **ParameterCode**
- Default : **ChannelCreationOptions** , **RaiseEventOptions**
- DefaultMaxSubscribers : **ChatClient**
- DND : **ChatUserStatus**
- dontDestroyOnLoad : **PlayerNumbering**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes									
Class List					Class Index					Class Hierarchy					Class Members									
All		Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

- e -

- EmptyRoomTtl : **GamePropertyKey**
- EmptyRoomTTL : **ParameterCode**
- emptyRoomTtl : **RoomInfo**
- EmptyRoomTtl : **RoomOptions**
- EnableLobbyStatistics : **AppSettings** , **LoadBalancingClient**
- EncryptionData : **ParameterCode**
- EncryptionMode : **LoadBalancingClient** , **ParameterCode**
- ErrorInfo : **EventCode**
- EventForward : **ParameterCode**
- EvFinalMove : **PunTurnManager**
- EvMove : **PunTurnManager**
- ExpectedProtocol : **LoadBalancingClient** , **ParameterCode**
- ExpectedUsers : **GamePropertyKey**
- expectedUsers : **RoomInfo**
- ExpectedValues : **ParameterCode**
- ExternalHttpCallFailed : **ErrorCode**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes									
Class List					Class Index					Class Hierarchy					Class Members									
All		Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

- f -

- FindFriends : **OperationCode**
- FindFriendsRequestList : **ParameterCode**
- FindFriendsResponseOnlineList : **ParameterCode**
- FindFriendsResponseRoomIdList : **ParameterCode**
- FinishedTurnPropKey : **TurnExtensions**
- FIRST_GROUP_ID : **CullArea**
- FixedRegion : **AppSettings**
- Flags : **RaiseEventOptions**
- Friends : **ChatParameterCode**
- FriendsList : **ChatEventCode**



Photon Unity Networking 2 2.12

Main Page				Related Pages					Modules				Classes										
Class List				Class Index			Class Hierarchy				Class Members												
All		Functions			Variables			Enumerations				Properties				Events							
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

- g -

- GameClosed : **ErrorCode**
- GameCount : **ParameterCode**
- GameDoesNotExist : **ErrorCode**
- GameFull : **ErrorCode**
- GameldAlreadyExists : **ErrorCode**
- GameList : **EventCode** , **ParameterCode**
- GameListUpdate : **EventCode**
- GameProperties : **ParameterCode**
- GetGameList : **OperationCode**
- GetLobbyStats : **OperationCode**
- GetProperties : **OperationCode**
- GetRegions : **OperationCode**
- Group : **ParameterCode**



Photon Unity Networking 2 2.12

Main Page				Related Pages						Modules					Classes									
Class List				Class Index				Class Hierarchy					Class Members											
All		Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

- h -

- healthStatsVisible : **PhotonStatsGui**
- HistoryLength : **ChatParameterCode**
- HoverColor : **TextButtonTransition**
- HoverOffColor : **TextToggleIsOnTransition**
- HoverOnColor : **TextToggleIsOnTransition**
- HttpLimitReached : **ErrorCode**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes									
Class List					Class Index					Class Hierarchy					Class Members									
All		Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

- i -

- Id : **CellTreeNode**
- Info : **ParameterCode**
- instance : **PlayerNumbering**
- InterestGroup : **RaiseEventOptions**
- InternalServerError : **ErrorCode**
- InvalidAuthentication : **ErrorCode**
- InvalidEncryptionParameters : **ErrorCode**
- InvalidOperation : **ErrorCode**
- InvalidOperationCode : **ErrorCode**
- InvalidRegion : **ErrorCode**
- Invisible : **ChatUserStatus**
- IsComingBack : **ParameterCode**
- IsInactive : **ActorProperties** , **ParameterCode**
- IsLocal : **Player**
- IsOpen : **GamePropertyKey**
- isOpen : **RoomInfo**
- IsVisible : **GamePropertyKey**
- isVisible : **RoomInfo**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes											
Class List				Class Index				Class Hierarchy				Class Members											
All		Functions				Variables				Enumerations				Properties				Events					
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

- j -

- Join : **EventCode** , **OperationCode**
- JoinFailedFoundActiveJoiner : **ErrorCode**
- JoinFailedFoundExcludedUserId : **ErrorCode**
- JoinFailedFoundInactiveJoiner : **ErrorCode**
- JoinFailedPeerAlreadyJoined : **ErrorCode**
- JoinFailedWithRejoinerNotFound : **ErrorCode**
- JoinGame : **OperationCode**
- JoinLobby : **OperationCode**
- JoinMode : **ParameterCode**
- JoinRandomGame : **OperationCode**



Photon Unity Networking 2 2.12

Main Page				Related Pages				Modules				Classes											
Class List				Class Index				Class Hierarchy				Class Members											
All		Functions			Variables			Enumerations				Properties				Events							
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

- k -

- KeepAliveInBackground : **ConnectionHandler**



Photon Unity Networking 2 2.12

Main Page				Related Pages					Modules				Classes										
Class List				Class Index			Class Hierarchy					Class Members											
All		Functions			Variables			Enumerations				Properties				Events							
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

- | -

- Leave : **EventCode** , **OperationCode**
- LeaveLobby : **OperationCode**
- LFG : **ChatUserStatus**
- LobbyName : **ParameterCode**
- LobbyStats : **EventCode** , **ParameterCode**
- LobbyType : **ParameterCode**
- LogLevel : **PhotonNetwork**
- LogTrafficStats : **SupportLogger**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes										
Class List					Class Index					Class Hierarchy					Class Members										
All			Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			

- m -

- MasterClientId : **GamePropertyKey** , **ParameterCode**
- masterClientId : **RoomInfo**
- MasterPeerCount : **ParameterCode**
- Match : **EventCode**
- MatchMakingCallbackTargets : **LoadBalancingClient**
- MatchMakingType : **ParameterCode**
- MAX_VIEW_IDS : **PhotonNetwork**
- MaxCcuReached : **ErrorCode**
- MaxPlayers : **GamePropertyKey**
- maxPlayers : **RoomInfo**
- MaxPlayers : **RoomOptions**
- Message : **ChatParameterCode**
- MessageLimit : **ChatChannel** , **ChatClient**
- Messages : **ChatChannel** , **ChatParameterCode**
- Mode : **EncryptionDataParameters**
- MsgId : **ChatParameterCode**
- MsgIds : **ChatParameterCode**



Photon Unity Networking 2 2.12

Main Page				Related Pages					Modules				Classes										
Class List				Class Index			Class Hierarchy				Class Members												
All		Functions			Variables			Enumerations				Properties				Events							
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

- n -

- Name : **ChatChannel**
- name : **RoomInfo**
- Name : **TypedLobby**
- NameServerHost : **ChatPeer** , **LoadBalancingClient**
- NameServerHttp : **ChatPeer** , **LoadBalancingClient**
- NetworkingClient : **PhotonNetwork**
- NetworkLogging : **AppSettings**
- NickName : **ParameterCode**
- NodeType : **CellTreeNode**
- NoRandomMatchFound : **ErrorCode**
- NormalColor : **TextButtonTransition**
- NormalOffColor : **TextToggleIsOnTransition**
- NormalOnColor : **TextToggleIsOnTransition**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes										
Class List					Class Index					Class Hierarchy					Class Members										
All			Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			

- 0 -

- ObjectsInOneUpdate : **PhotonNetwork**
- Offline : **ChatUserStatus**
- Ok : **ErrorCode**
- Online : **ChatUserStatus**
- OnTabChanged : **TabViewManager**
- OperationNotAllowedInCurrentState : **ErrorCode**
- OwnershipTransfer : **PhotonView**
- OwnershipWasTransferred : **PhotonView**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes										
Class List					Class Index					Class Hierarchy					Class Members										
All			Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			

- p -

- Parent : **CellTreeNode**
- PeerCount : **ParameterCode**
- PhotonServerSettings : **PhotonNetwork**
- PlayerCount : **GamePropertyKey**
- PlayerName : **ActorProperties**
- PlayerProperties : **ParameterCode**
- PlayersPerTeam : **PunTeams**
- PlayerTtl : **GamePropertyKey**
- PlayerTTL : **ParameterCode**
- playerTtl : **RoomInfo**
- PlayerTtl : **RoomOptions**
- Playing : **ChatUserStatus**
- PluginMismatch : **ErrorCode**
- PluginName : **ParameterCode**
- PluginReportedError : **ErrorCode**
- Plugins : **ParameterCode** , **RoomOptions**
- PluginVersion : **ParameterCode**
- Port : **AppSettings**
- Position : **ParameterCode**
- PrecisionForFloatSynchronization : **PhotonNetwork**
- PrecisionForQuaternionSynchronization : **PhotonNetwork**
- PrecisionForVectorSynchronization : **PhotonNetwork**
- PrivateChannels : **ChatClient**
- PrivateMessage : **ChatEventCode**
- Properties : **ChatParameterCode** , **ParameterCode**
- PropertiesChanged : **EventCode**

- propertiesListedInLobby : **RoomInfo**
- PropsListedInLobby : **GamePropertyKey**
- Protocol : **AppSettings**
- PublicChannels : **ChatClient**
- Publish : **ChatOperationCode**
- PublishUserId : **ParameterCode**
- PunVersion : **PhotonNetwork**



Photon Unity Networking 2 2.12

Main Page				Related Pages					Modules					Classes									
Class List				Class Index			Class Hierarchy					Class Members											
All		Functions			Variables			Enumerations					Properties					Events					
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	

- q -

- QueueState : **EventCode**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes										
Class List					Class Index					Class Hierarchy					Class Members										
All			Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			

- r -

- RaiseEvent : **OperationCode**
- ReceiverGroup : **ParameterCode**
- Receivers : **RaiseEventOptions**
- Region : **ParameterCode**
- RegionHandler : **LoadBalancingClient**
- Remove : **ParameterCode**
- Removed : **GamePropertyKey**
- RemovedFromList : **RoomInfo**
- RemoveFriends : **ChatOperationCode**
- ResourceCache : **DefaultPool**
- RoomName : **ParameterCode**
- RoomOptionFlags : **ParameterCode**
- RoomPlayerIndexedProp : **PlayerNumbering**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes										
Class List					Class Index					Class Hierarchy					Class Members										
All			Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w			

- S -

- Secret : **ChatParameterCode** , **ParameterCode**
- Secret1 : **EncryptionDataParameters**
- Secret2 : **EncryptionDataParameters**
- Selectable : **TextButtonTransition**
- Sender : **ChatParameterCode** , **PhotonMessageInfo**
- Senders : **ChatChannel** , **ChatParameterCode**
- SendPrivate : **ChatOperationCode**
- SequenceChannel : **RaiseEventOptions**
- Server : **AppSettings**
- ServerFull : **ErrorCode**
- ServerSettings : **OperationCode**
- SetProperties : **EventCode** , **OperationCode**
- SkipMessage : **ChatParameterCode**
- SlotError : **ErrorCode**
- statsOn : **PhotonStatsGui**
- statsRect : **PhotonStatsGui**
- statsWindowOn : **PhotonStatsGui**
- Status : **ChatParameterCode**
- StatusUpdate : **ChatEventCode**
- SUBDIVISION_FIRST_LEVEL_ORDER : **CullArea**
- SUBDIVISION_SECOND_LEVEL_ORDER : **CullArea**
- SUBDIVISION_THIRD_LEVEL_ORDER : **CullArea**
- Subscribe : **ChatEventCode** , **ChatOperationCode**
- SubscribeResults : **ChatParameterCode**
- Subscribers : **ChatChannel**
- SuppressRoomEvents : **ParameterCode**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes									
Class List					Class Index					Class Hierarchy					Class Members									
All		Functions			Variables			Enumerations					Properties					Events						
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

- t -

- Tabs : **TabViewManager**
- TagObject : **Player**
- TargetActorNr : **ParameterCode**
- TargetActors : **RaiseEventOptions**
- TeamPlayerProp : **PunTeams**
- toggle : **TextToggleIsOnTransition**
- ToggleGroup : **TabViewManager**
- trafficStatsOn : **PhotonStatsGui**
- TurnDuration : **PunTurnManager**
- TurnManagerEventOffset : **PunTurnManager**
- TurnManagerListener : **PunTurnManager**
- TurnPropKey : **TurnExtensions**
- TurnStartPropKey : **TurnExtensions**
- Type : **TypedLobby**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes									
Class List					Class Index					Class Hierarchy					Class Members									
All		Functions			Variables			Enumerations					Properties					Events						
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

- u -

- Unsubscribe : **ChatEventCode** , **ChatOperationCode**
- UpdateStatus : **ChatOperationCode**
- UriPath : **ParameterCode**
- UseNameServer : **AppSettings**
- UserBlocked : **ErrorCode**
- UserId : **ChatParameterCode** , **ParameterCode** , **ActorProperties** , **ParameterCode**
- UseRpcMonoBehaviourCache : **PhotonNetwork**
- Users : **ChatEventCode**
- UserSubscribed : **ChatEventCode**
- UserUnsubscribed : **ChatEventCode**



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes									
Class List					Class Index					Class Hierarchy					Class Members									
All		Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

- v -

- Version : **ConnectAndJoinRandom**
- Visible : **PhotonLagSimulationGui**

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page					Related Pages					Modules					Classes									
Class List					Class Index					Class Hierarchy					Class Members									
All		Functions				Variables				Enumerations					Properties					Events				
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w		

- W -

- WebFlags : **ChatParameterCode**
- WebRpc : **OperationCode**
- WebRpcParameters : **ParameterCode**
- WebRpcReturnCode : **ParameterCode**
- WebRpcReturnMessage : **ParameterCode**
- WindowId : **PhotonLagSimulationGui** , **PhotonStatsGui**
- WindowRect : **PhotonLagSimulationGui**



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes		
Class List		Class Index		Class Hierarchy		Class Members	
All	Functions	Variables		Enumerations		Properties	Events

- Team : **PunTeams**



Photon Unity Networking 2 2.12

Main Page				Related Pages					Modules					Classes							
Class List				Class Index			Class Hierarchy					Class Members									
All		Functions			Variables			Enumerations					Properties					Events			
a	b	c	d	e	f	g	h	i	k	l	m	n	o	p	q	r	s	t	u	v	

- a -

- ActorNumber : **Player**
- AppId : **ChatClient** , **LoadBalancingClient**
- AppVersion : **ChatClient** , **PhotonNetwork** , **LoadBalancingClient**
- AuthGetParameters : **AuthenticationValues**
- AuthPostData : **AuthenticationValues**
- AuthType : **AuthenticationValues**
- AuthValues : **ChatClient** , **PhotonNetwork** , **LoadBalancingClient**
- AutoCleanUp : **Room**
- AutomaticallySyncScene : **PhotonNetwork**

- b -

- BestRegion : **RegionHandler**
- BestRegionSummaryInPreferences : **PhotonNetwork** , **ServerSettings**
- BroadcastPropsChangeToAll : **RoomOptions**

- c -

- CanChat : **ChatClient**
- ChatRegion : **ChatClient**
- CleanupCacheOnLeave : **RoomOptions**

- Client : **ConnectionHandler** , **SupportLogger**
- CloudRegion : **PhotonNetwork** , **LoadBalancingClient**
- Cluster : **Region**
- Count : **PhotonStream**
- CountOfPlayers : **PhotonNetwork**
- CountOfPlayersInRooms : **PhotonNetwork**
- CountOfPlayersOnMaster : **PhotonNetwork**
- CountOfRooms : **PhotonNetwork**
- CountSendAcksOnly : **ConnectionHandler**
- CrcCheckEnabled : **PhotonNetwork**
- CurrentLobby : **PhotonNetwork** , **LoadBalancingClient**
- CurrentRoom : **PhotonNetwork** , **LoadBalancingClient**
- CurrentServerAddress : **LoadBalancingClient**
- CustomProperties : **Player** , **RoomInfo**

- d -

- DebugMessage : **WebRpcResponse**
- DebugOut : **ChatClient**
- DeleteNullProperties : **RoomOptions**
- DisconnectedCause : **ChatClient** , **LoadBalancingClient**

- e -

- ElapsedTimeInTurn : **PunTurnManager**
- EmptyRoomTtl : **Room**
- EnabledRegions : **RegionHandler**
- EnableLobbyStatistics : **PhotonNetwork**
- ExpectedUsers : **Room**

- f -

- FrontendAddress : **ChatClient**

- g -

- GameServerAddress : **LoadBalancingClient**

- GameVersion : **PhotonNetwork**

- h -

- HttpForward : **WebFlags**

- i -

- InLobby : **PhotonNetwork , LoadBalancingClient**
- InRoom : **PhotonNetwork , LoadBalancingClient**
- InstantiationData : **PhotonView**
- IsBestRegion : **AppSettings**
- IsCompletedByAll : **PunTurnManager**
- IsConnected : **PhotonNetwork , LoadBalancingClient**
- IsConnectedAndReady : **PhotonNetwork , LoadBalancingClient**
- IsDefaultNameServer : **AppSettings**
- IsDefaultPort : **AppSettings**
- IsFetchingFriendList : **LoadBalancingClient**
- IsFinishedByMe : **PunTurnManager**
- IsInactive : **Player**
- IsMasterClient : **PhotonNetwork , Player**
- IsMasterServerAddress : **AppSettings**
- IsMessageQueueRunning : **PhotonNetwork**
- IsMine : **PhotonView**
- IsOpen : **Room , RoomInfo , RoomOptions**
- IsOver : **PunTurnManager**
- IsPrivate : **ChatChannel**
- IsReading : **PhotonStream**
- IsSceneView : **PhotonView**
- IsUsingNameServer : **LoadBalancingClient**
- IsVisible : **Room , RoomInfo , RoomOptions**
- IsWriting : **PhotonStream**

- k -

- KeepAliveInBackground : **PhotonNetwork**

- l -

- LastMsgId : **ChatChannel**
- LevelLoadingProgress : **PhotonNetwork**
- LoadBalancingClient : **Room**
- LoadBalancingPeer : **LoadBalancingClient**
- LocalPlayer : **PhotonNetwork** , **LoadBalancingClient**

- m -

- MasterClient : **PhotonNetwork**
- MasterClientId : **Room**
- MasterServerAddress : **LoadBalancingClient**
- MaxPlayers : **Room** , **RoomInfo**
- MaxResendsBeforeDisconnect : **PhotonNetwork**
- MaxSubscribers : **ChannelCreationOptions** , **ChatChannel**
- MessageCount : **ChatChannel**

- n -

- Name : **Room** , **RoomInfo** , **WebRpcResponse**
- NameServerAddress : **ChatClient** , **ChatPeer** , **LoadBalancingClient**
- NetworkClientState : **PhotonNetwork**
- NetworkStatisticsEnabled : **PhotonNetwork**
- NickName : **PhotonNetwork** , **LoadBalancingClient** , **Player**

- o -

- OfflineMode : **PhotonNetwork**
- Owner : **PhotonView**

- p -

- PacketLossByCrcCheck : **PhotonNetwork**
- Parameters : **WebRpcResponse**
- Peer : **PhotonLagSimulationGui**

- photonView : **MonoBehaviourPun**
- PhotonViews : **PhotonNetwork**
- PlayerCount : **Room , RoomInfo**
- PlayerList : **PhotonNetwork**
- PlayerListOthers : **PhotonNetwork**
- Players : **Room**
- PlayersInRoomsCount : **LoadBalancingClient**
- PlayersOnMasterCount : **LoadBalancingClient**
- PlayerTtl : **Room**
- PrefabPool : **PhotonNetwork**
- PropertiesListedInLobby : **Room**
- PublishSubscribers : **ChannelCreationOptions , ChatChannel**
- PublishUserId : **RoomOptions**

- q -

- QuickResends : **PhotonNetwork**

- r -

- RemainingSecondsInTurn : **PunTurnManager**
- ResentReliableCommands : **PhotonNetwork**
- ReturnCode : **WebRpcResponse**
- RoomsCount : **LoadBalancingClient**
- RootNode : **CellTree**

- s -

- SendAuthCookie : **WebFlags**
- SendRate : **PhotonNetwork**
- SendState : **WebFlags**
- SendSync : **WebFlags**
- SerializationRate : **PhotonNetwork**
- Server : **PhotonNetwork , LoadBalancingClient**
- ServerAddress : **PhotonNetwork**
- ServerTimestamp : **PhotonNetwork**
- SocketImplementationConfig : **ChatClient**
- State : **ChatClient , LoadBalancingClient**

- SummaryToCache : **RegionHandler**
- SuppressRoomEvents : **RoomOptions**

- t -

- Time : **PhotonNetwork**
- Token : **AuthenticationValues**
- TransportProtocol : **ChatClient**
- Turn : **PunTurnManager**

- u -

- UseAlternativeUdpPorts : **PhotonNetwork** , **LoadBalancingClient**
- UseBackgroundWorkerForSending : **ChatClient**
- UserId : **AuthenticationValues** , **ChatClient** , **AuthenticationValues** , **LoadBalancingClient** , **Player**

- v -

- ViewID : **PhotonView**



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes		
Class List		Class Index		Class Hierarchy		Class Members	
All	Functions	Variables	Enumerations		Properties		Events

- EventReceived : **LoadBalancingClient**
- OnCountdownTimerHasExpired : **CountdownTimer**
- OnPlayerNumberingChanged : **PlayerNumbering**
- OpResponseReceived : **LoadBalancingClient**
- StateChanged : **LoadBalancingClient**



Photon Unity Networking 2 2.12

[Main Page](#)[Related Pages](#)[Modules](#)[Classes](#)

Related Pages

Here is a list of all related documentation pages:

General Documentation	Brief overview of Photon, subscriptions, hosting options and how to start
Network Simulation GUI	Simple GUI element to control the built-in network condition simulation
Network Statistics GUI	The PhotonStatsGui is a simple GUI component to track and show network-metrics at runtime
Public API Module	The Public API module rounds up the most commonly used classes of PUN

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonNetwork		

PhotonNetwork Member List

This is the complete list of members for **PhotonNetwork**, including all inherited members.

AddCallbackTarget(object target)

AllocateSceneViewID(PhotonView view)

AllocateViewID(PhotonView view)

AppVersion

AuthValues

AutomaticallySyncScene

BackgroundTimeout (defined in **PhotonNetwork**)

BestRegionSummaryInPreferences

CleanRpcBufferIfMine(PhotonView view) (defined in **PhotonNetwork**)

CloseConnection(Player kickPlayer)

CloudRegion

ConnectMethod

ConnectToBestCloudServer()

ConnectToMaster(string masterServerAddress, int port, string appId)

ConnectToRegion(string region)

ConnectUsingSettings()

CountOfPlayers

CountOfPlayersInRooms

CountOfPlayersOnMaster

CountOfRooms

CrcCheckEnabled

CreateRoom(string roomName, RoomOptions roomOptions=null, Type

CurrentLobby

CurrentRoom

Destroy(PhotonView targetView)

Destroy(GameObject targetGo)

DestroyAll()

DestroyAll(bool localOnly) (defined in **PhotonNetwork**)

DestroyPlayerObjects(Player targetPlayer)

DestroyPlayerObjects(int targetPlayerId)

DestroyPlayerObjects(int playerId, bool localOnly)

Disconnect()

EnableLobbyStatistics

FetchServerTimestamp()

FindFriends(string[] friendsToFind)

FindGameObjectsWithComponent(Type type)

GameVersion

GetCustomRoomList(TypedLobby typedLobby, string sqlLobbyFilter)

GetPhotonView(int viewID) (defined in **PhotonNetwork**)

GetPing()

InLobby

InRoom

Instantiate(string prefabName, Vector3 position, Quaternion rotation, by

InstantiateSceneObject(string prefabName, Vector3 position, Quaternion

IsConnected

IsConnectedAndReady

IsMasterClient

IsMessageQueueRunning

JoinLobby()

JoinLobby(TypedLobby typedLobby)

JoinOrCreateRoom(string roomName, RoomOptions roomOptions, Typ

JoinRandomRoom()

JoinRandomRoom(Hashtable expectedCustomRoomProperties, byte e

JoinRandomRoom(Hashtable expectedCustomRoomProperties, byte e

JoinRoom(string roomName, string[] expectedUsers=null)

KeepAliveInBackground

LeaveLobby()

LeaveRoom(bool becomeInactive=true)

LevelLoadingProgress

LoadLevel(int levelNumber)

LoadLevel(string levelName)

LocalCleanPhotonView(PhotonView view) (defined in **PhotonNetwork**

LocalPlayer

LogLevel

MasterClient

MAX_VIEW_IDS

MaxResendsBeforeDisconnect

NetworkClientState

NetworkingClient

NetworkStatisticsEnabled

NetworkStatisticsReset()

NetworkStatisticsToString()

NickName

ObjectsInOneUpdate

OfflineMode

OpCleanActorRpcBuffer(int actorNumber)

OpCleanRpcBuffer(PhotonView view)

OpRemoveCompleteCache() (defined in **PhotonNetwork**)

OpRemoveCompleteCacheOfPlayer(int actorNumber)

PacketLossByCrcCheck

PhotonServerSettings

PhotonViews

PlayerList

PlayerListOthers

PrecisionForFloatSynchronization

PrecisionForQuaternionSynchronization

PrecisionForVectorSynchronization

PrefabPool

PunVersion

QuickResends

RaiseEvent(byte eventCode, object eventContent, RaiseEventOptions r
Reconnect()

ReconnectAndRejoin()

RegisterPhotonView(PhotonView netView) (defined in **PhotonNetwork**)

RejoinRoom(string roomName)

RemoveCallbackTarget(object target)

RemovePlayerCustomProperties(string[] customPropertiesToDelete)

RemoveRPCs(Player targetPlayer)

RemoveRPCs(PhotonView targetPhotonView)

RemoveRPCsInGroup(int group)

ResentReliableCommands

SendAllOutgoingCommands()

SendRate

SerializationRate

Server

ServerAddress

ServerTimestamp

SetInterestGroups(byte group, bool enabled)

SetInterestGroups(byte[] disableGroups, byte[] enableGroups)

SetLevelPrefix(byte prefix)

SetMasterClient(Player masterClientPlayer)

SetPlayerCustomProperties(Hashtable customProperties)

SetSendingEnabled(byte group, bool enabled)

SetSendingEnabled(byte[] disableGroups, byte[] enableGroups)

SyncCompressed (defined in **PhotonNetwork**)

SyncFirstValue (defined in **PhotonNetwork**)

SyncNullValues (defined in **PhotonNetwork**)

SyncViewId (defined in **PhotonNetwork**)

Time

UseAlternativeUdpPorts

UseRpcMonoBehaviourCache

WebRpc(string name, object parameters, bool sendAuthCookie=false)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonView		

PhotonView Member List

This is the complete list of members for **PhotonView**, including all inherited members.

Controller (defined in **PhotonView**)

ControllerActorNr (defined in **PhotonView**)

CreatorActorNr (defined in **PhotonView**)

DeserializeView(PhotonStream stream, PhotonMessageInfo info) (defir

Find(int viewID) (defined in **PhotonView**)

Get(Component component) (defined in **PhotonView**)

Get(GameObject gameObj) (defined in **PhotonView**)

Group (defined in **PhotonView**)

InstantiationData

InstantiationId (defined in **PhotonView**)

IsMine

IsOwnerActive (defined in **PhotonView**)

IsSceneView

ObservedComponents (defined in **PhotonView**)

Owner

OwnerActorNr (defined in **PhotonView**)

OwnershipTransfer

OwnershipWasTransferred

Prefix (defined in **PhotonView**)

prefixField (defined in **PhotonView**)

RefreshRpcMonoBehaviourCache()

RequestOwnership()

RPC(string methodName, RpcTarget target, params object[] parameters)

RPC(string methodName, Player targetPlayer, params object[] parameters)

RpcSecure(string methodName, RpcTarget target, bool encrypt, params object[] parameters)

RpcSecure(string methodName, Player targetPlayer, bool encrypt, params object[] parameters)

SerializeView(PhotonStream stream, PhotonMessageInfo info) (defined in **PhotonView**)

Synchronization (defined in **PhotonView**)

ToString() (defined in **PhotonView**)

TransferOwnership(Player newOwner)

TransferOwnership(int newOwnerId)

ViewID



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonMessageInfo		

PhotonMessageInfo Member List

This is the complete list of members for **PhotonMessageInfo**, including all inherited members.

PhotonMessageInfo(Player player, int timestamp, PhotonView view) (d

photonView (defined in **PhotonMessageInfo**)

Sender

SentServerTime (defined in **PhotonMessageInfo**)

SentServerTimestamp (defined in **PhotonMessageInfo**)

timestamp (defined in **PhotonMessageInfo**)

ToString() (defined in **PhotonMessageInfo**)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonStream		

PhotonStream Member List

This is the complete list of members for **PhotonStream**, including all inherited members.

CopyToListAndClear(List< object > target) (defined in **PhotonStream**)

Count

IsReading

IsWriting

PeekNext()

PhotonStream(bool write, object[] incomingData)

ReceiveNext()

SendNext(object obj)

Serialize(ref bool myBool)

Serialize(ref int myInt)

Serialize(ref string value)

Serialize(ref char value)

Serialize(ref short value)

Serialize(ref float obj)

Serialize(ref Player obj)

Serialize(ref Vector3 obj)

Serialize(ref Vector2 obj)

Serialize(ref Quaternion obj)

SetReadStream(object[] incomingData, byte pos=0) (defined in **PhotonStream**)
ToArray()



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	PhotonLagSimulationGui	

PhotonLagSimulationGui Member List

This is the complete list of members for **PhotonLagSimulationGui**, including all inherited members.

OnGUI() (defined in PhotonLagSimulationGui)	PhotonLagSimulation
Peer	PhotonLagSimulation
Start() (defined in PhotonLagSimulationGui)	PhotonLagSimulation
Visible	PhotonLagSimulation
WindowId	PhotonLagSimulation
WindowRect	PhotonLagSimulation



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	PhotonStatsGui	

PhotonStatsGui Member List

This is the complete list of members for **PhotonStatsGui**, including all inherited members.

buttonsOn	Photo
healthStatsVisible	Photo
OnGUI() (defined in PhotonStatsGui)	Photo
Start() (defined in PhotonStatsGui)	Photo
statsOn	Photo
statsRect	Photo
statsWindowOn	Photo
trafficStatsOn	Photo
TrafficStatsWindow (int windowID) (defined in PhotonStatsGui)	Photo
Update()	Photo
WindowId	Photo



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	IConnectionCallbacks		

IConnectionCallbacks Member List

This is the complete list of members for **IConnectionCallbacks**, including all inherited members.

OnConnected()

OnConnectedToMaster()

OnCustomAuthenticationFailed(string debugMessage)

OnCustomAuthenticationResponse(Dictionary< string, object > data)

OnDisconnected(DisconnectCause cause)

OnRegionListReceived(RegionHandler regionHandler)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	ILobbyCallbacks		

ILobbyCallbacks Member List

This is the complete list of members for **ILobbyCallbacks**, including all inherited members.

OnJoinedLobby()	ILo
OnLeftLobby()	ILo
OnLobbyStatisticsUpdate(List< TypedLobbyInfo > lobbyStatistics)	ILo
OnRoomListUpdate(List< RoomInfo > roomList)	ILo



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	IMatchmakingCallbacks		

IMatchmakingCallbacks Member List

This is the complete list of members for **IMatchmakingCallbacks**, including all inherited members.

OnCreatedRoom()	IMatchmakingCallbacks
OnCreateRoomFailed (short returnCode, string message)	IMatchmakingCallbacks
OnFriendListUpdate (List< FriendInfo > friendList)	IMatchmakingCallbacks
OnJoinedRoom()	IMatchmakingCallbacks
OnJoinRandomFailed (short returnCode, string message)	IMatchmakingCallbacks
OnJoinRoomFailed (short returnCode, string message)	IMatchmakingCallbacks
OnLeftRoom()	IMatchmakingCallbacks



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	IInRoomCallbacks		

IInRoomCallbacks Member List

This is the complete list of members for **IInRoomCallbacks**, including all inherited members.

OnMasterClientSwitched(Player newMasterClient)

OnPlayerEnteredRoom(Player newPlayer)

OnPlayerLeftRoom(Player otherPlayer)

OnPlayerPropertiesUpdate(Player targetPlayer, Hashtable changedPr

OnRoomPropertiesUpdate(Hashtable propertiesThatChanged)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	IOnEventCallback		

IOnEventCallback Member List

This is the complete list of members for **IOnEventCallback**, including all inherited members.

OnEvent(EventData photonEvent) **IOnEventCallback**



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	IWebRpcCallback		

IWebRpcCallback Member List

This is the complete list of members for **IWebRpcCallback**, including all inherited members.

OnWebRpcResponse(OperationResponse response) **IWebRpcCallba**



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	IPunObservable		

IPunObservable Member List

This is the complete list of members for **IPunObservable**, including all inherited members.

OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	IPunOwnershipCallbacks		

IPunOwnershipCallbacks Member List

This is the complete list of members for **IPunOwnershipCallbacks**, including all inherited members.

OnOwnershipRequest(PhotonView targetView, Player requestingPlaye

OnOwnershipTransferred(PhotonView targetView, Player previousOwn



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	IPunInstantiateMagicCallback		

IPunInstantiateMagicCallback Member List

This is the complete list of members for **IPunInstantiateMagicCallback**, including all inherited members.

OnPhotonInstantiate(PhotonMessageInfo info) (defined in **IPunInstant**



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	MonoBehaviourPunCallbacks		

MonoBehaviourPunCallbacks Member List

This is the complete list of members for **MonoBehaviourPunCallbacks**, including all inherited members.

- OnConnected()**
- OnConnectedToMaster()**
- OnCreatedRoom()**
- OnCreateRoomFailed**(short returnCode, string message)
- OnCustomAuthenticationFailed**(string debugMessage)
- OnCustomAuthenticationResponse**(Dictionary< string, object > data)
- OnDisable()** (defined in **MonoBehaviourPunCallbacks**)
- OnDisconnected**(DisconnectCause cause)
- OnEnable()** (defined in **MonoBehaviourPunCallbacks**)
- OnFriendListUpdate**(List< FriendInfo > friendList)
- OnJoinedLobby()**
- OnJoinedRoom()**
- OnJoinRandomFailed**(short returnCode, string message)
- OnJoinRoomFailed**(short returnCode, string message)
- OnLeftLobby()**
- OnLeftRoom()**
- OnLobbyStatisticsUpdate**(List< TypedLobbyInfo > lobbyStatistics)
- OnMasterClientSwitched**(Player newMasterClient)
- OnPlayerEnteredRoom**(Player newPlayer)
- OnPlayerLeftRoom**(Player otherPlayer)

OnPlayerPropertiesUpdate(Player target, Hashtable changedProps)

OnRegionListReceived(RegionHandler regionHandler)

OnRoomListUpdate(List< RoomInfo > roomList)

OnRoomPropertiesUpdate(Hashtable propertiesThatChanged)

OnWebRpcResponse(OperationResponse response) (defined in **Monoc
photonView**)



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	AuthenticationValues		

AuthenticationValues Member List

This is the complete list of members for **AuthenticationValues**, including all inherited members.

AddAuthParameter (string key, string value)	AuthenticationValues	virt
AuthenticationValues ()	AuthenticationValues	
AuthenticationValues (string userId)	AuthenticationValues	
AuthGetParameters	AuthenticationValues	
AuthPostData	AuthenticationValues	
AuthType	AuthenticationValues	
SetAuthPostData (string stringData)	AuthenticationValues	virt
SetAuthPostData (byte[] byteData)	AuthenticationValues	virt
Token	AuthenticationValues	
ToString ()	AuthenticationValues	
UserId	AuthenticationValues	



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChannelCreationOptions		

ChannelCreationOptions Member List

This is the complete list of members for **ChannelCreationOptions**, including all inherited members.

Default	ChannelCreationOptions	static
MaxSubscribers	ChannelCreationOptions	
PublishSubscribers	ChannelCreationOptions	



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChannelWellKnownProperties		

ChannelWellKnownProperties Member List

This is the complete list of members for **ChannelWellKnownProperties**, including all inherited members.

MaxSubscribers (defined in ChannelWellKnownProperties)	Char
PublishSubscribers (defined in ChannelWellKnownProperties)	Char



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChatChannel		

ChatChannel Member List

This is the complete list of members for **ChatChannel**, including all inherited members.

Add (string sender, object message, int msgId)	ChatChannel
Add (string[] senders, object[] messages, int lastMsgId)	ChatChannel
ChatChannel (string name)	ChatChannel
ClearMessages ()	ChatChannel
IsPrivate	ChatChannel
LastMsgId	ChatChannel
MaxSubscribers	ChatChannel
MessageCount	ChatChannel
MessageLimit	ChatChannel
Messages	ChatChannel
Name	ChatChannel
PublishSubscribers	ChatChannel
Senders	ChatChannel
Subscribers	ChatChannel
ToStringMessages ()	ChatChannel
TruncateMessages ()	ChatChannel



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChatClient		

ChatClient Member List

This is the complete list of members for **ChatClient**, including all inherited members.

AddFriends(string[] friends)

AppId

AppVersion

AuthValues

CanChat

CanChatInChannel(string channelName)

ChatClient(IChatClientListener listener, ConnectionProtocol protocol=C
chatPeer

ChatRegion

Connect(string appId, string appVersion, AuthenticationValues authValu

ConnectAndSetStatus(string appId, string appVersion, AuthenticationV

DebugOut

DefaultMaxSubscribers

Disconnect()

DisconnectedCause

FrontendAddress

GetPrivateChannelNameByUser(string userName)

MessageLimit

NameServerAddress

PrivateChannels

PublicChannels

PublishMessage(string channelName, object message, bool forwardAs

RemoveFriends(string[] friends)

SendAcksOnly()

SendPrivateMessage(string target, object message, bool forwardAsWe

SendPrivateMessage(string target, object message, bool encrypt, bool
Service())

SetOnlineStatus(int status)

SetOnlineStatus(int status, object message)

SocketImplementationConfig

State

StopThread()

Subscribe(string[] channels)

Subscribe(string[] channels, int[] lastMsgIds)

Subscribe(string[] channels, int messagesFromHistory)

Subscribe(string channel, int lastMsgId=0, int messagesFromHistory=-1

TransportProtocol

TryGetChannel(string channelName, bool isPrivate, out ChatChannel c

TryGetChannel(string channelName, out ChatChannel channel)

Unsubscribe(string[] channels)

UseBackgroundWorkerForSending

UserId



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChatEventCode		

ChatEventCode Member List

This is the complete list of members for **ChatEventCode**, including all inherited members.

ChatMessages	ChatEventCode
FriendsList	ChatEventCode
PrivateMessage	ChatEventCode
StatusUpdate	ChatEventCode
Subscribe	ChatEventCode
Unsubscribe	ChatEventCode
Users	ChatEventCode
UserSubscribed	ChatEventCode
UserUnsubscribed	ChatEventCode



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChatOperationCode		

ChatOperationCode Member List

This is the complete list of members for **ChatOperationCode**, including all inherited members.

AddFriends	ChatOperationCode
Authenticate	ChatOperationCode
ChannelHistory	ChatOperationCode
Publish	ChatOperationCode
RemoveFriends	ChatOperationCode
SendPrivate	ChatOperationCode
Subscribe	ChatOperationCode
Unsubscribe	ChatOperationCode
UpdateStatus	ChatOperationCode



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChatParameterCode		

ChatParameterCode Member List

This is the complete list of members for **ChatParameterCode**, including all inherited members.

Channel	ChatParameterCode
Channels	ChatParameterCode
ChannelSubscribers	ChatParameterCode
ChannelUserCount	ChatParameterCode
Friends	ChatParameterCode
HistoryLength	ChatParameterCode
Message	ChatParameterCode
Messages	ChatParameterCode
MsgId	ChatParameterCode
MsgIds	ChatParameterCode
Properties	ChatParameterCode
Secret	ChatParameterCode
Sender	ChatParameterCode
Senders	ChatParameterCode
SkipMessage	ChatParameterCode
Status	ChatParameterCode
SubscribeResults	ChatParameterCode
UserId	ChatParameterCode
WebFlags	ChatParameterCode



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChatPeer		

ChatPeer Member List

This is the complete list of members for **ChatPeer**, including all inherited members.

```
AuthenticateOnNameServer(string appld, string appVersion, string reg  
ChatPeer(IPhotonPeerListener listener, ConnectionProtocol protocol)  
Connect()  
NameServerAddress  
NameServerHost  
NameServerHttp
```



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ChatUserStatus		

ChatUserStatus Member List

This is the complete list of members for **ChatUserStatus**, including all inherited members.

Away	ChatUserStatus
DND	ChatUserStatus
Invisible	ChatUserStatus
LFG	ChatUserStatus
Offline	ChatUserStatus
Online	ChatUserStatus
Playing	ChatUserStatus



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ErrorCode		

ErrorCode Member List

This is the complete list of members for **ErrorCode**, including all inherited members.

CustomAuthenticationFailed	ErrorCode
GameClosed	ErrorCode
GameDoesNotExist	ErrorCode
GameFull	ErrorCode
GameIdAlreadyExists	ErrorCode
InternalServerError	ErrorCode
InvalidAuthentication	ErrorCode
InvalidOperationCode	ErrorCode
InvalidRegion	ErrorCode
MaxCcuReached	ErrorCode
NoRandomMatchFound	ErrorCode
Ok	ErrorCode
OperationNotAllowedInCurrentState	ErrorCode
ServerFull	ErrorCode
UserBlocked	ErrorCode



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	IChatClientListener		

IChatClientListener Member List

This is the complete list of members for **IChatClientListener**, including all inherited members.

DebugReturn(DebugLevel level, string message)

OnChatStateChange(ChatState state)

OnConnected()

OnDisconnected()

OnGetMessages(string channelName, string[] senders, object[] message)

OnPrivateMessage(string sender, object message, string channelName)

OnStatusUpdate(string user, int status, bool gotMessage, object message)

OnSubscribed(string[] channels, bool[] results)

OnUnsubscribed(string[] channels)

OnUserSubscribed(string channel, string user)

OnUserUnsubscribed(string channel, string user)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Chat	ParameterCode		

ParameterCode Member List

This is the complete list of members for **ParameterCode**, including all inherited members.

Address	ParameterCode
ApplicationId	ParameterCode
AppVersion	ParameterCode
ClientAuthenticationData	ParameterCode
ClientAuthenticationParams	ParameterCode
ClientAuthenticationType	ParameterCode
Region	ParameterCode
Secret	ParameterCode
UserId	ParameterCode



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	ButtonInsideScrollList	

ButtonInsideScrollList Member List

This is the complete list of members for **ButtonInsideScrollList**, including all inherited members.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	CellTree	

CellTree Member List

This is the complete list of members for **CellTree**, including all inherited members.

CellTree()	CellTree
CellTree(CellTreeNode root)	CellTree
RootNode	CellTree



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	CellTreeNode	

CellTreeNode Member List

This is the complete list of members for **CellTreeNode**, including all inherited members.

AddChild(CellTreeNode child)

CellTreeNode()

CellTreeNode(byte id, ENodeType nodeType, CellTreeNode parent)

Center

Childs

Draw()

ENodeType enum name (defined in **CellTreeNode**)

GetActiveCells(List< byte > activeCells, bool yIsUpAxis, Vector3 position)

Id

IsPointInsideCell(bool yIsUpAxis, Vector3 point)

IsPointNearCell(bool yIsUpAxis, Vector3 point)

NodeType

Parent



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	ConnectAndJoinRandom	

ConnectAndJoinRandom Member List

This is the complete list of members for **ConnectAndJoinRandom**, including all inherited members.

AutoConnect

ConnectNow() (defined in **ConnectAndJoinRandom**)

OnConnected()

OnConnectedToMaster()

OnCreatedRoom()

OnCreateRoomFailed(short returnCode, string message)

OnCustomAuthenticationFailed(string debugMessage)

OnCustomAuthenticationResponse(Dictionary< string, object > data)

OnDisable() (defined in **MonoBehaviourPunCallbacks**)

OnDisconnected(DisconnectCause cause)

OnEnable() (defined in **MonoBehaviourPunCallbacks**)

OnFriendListUpdate(List< FriendInfo > friendList)

OnJoinedLobby()

OnJoinedRoom()

OnJoinRandomFailed(short returnCode, string message)

OnJoinRoomFailed(short returnCode, string message)

OnLeftLobby()

OnLeftRoom()

OnLobbyStatisticsUpdate(List< TypedLobbyInfo > lobbyStatistics)

OnMasterClientSwitched(Player newMasterClient)

OnPlayerEnteredRoom(Player newPlayer)

OnPlayerLeftRoom(Player otherPlayer)

OnPlayerPropertiesUpdate(Player target, Hashtable changedProps)

OnRegionListReceived(RegionHandler regionHandler)

OnRoomListUpdate(List< RoomInfo > roomList)

OnRoomPropertiesUpdate(Hashtable propertiesThatChanged)

OnWebRpcResponse(OperationResponse response) (defined in **Monocloner** photonView)

Start() (defined in **ConnectAndJoinRandom**)

Version



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	CountdownTimer	

CountdownTimer Member List

This is the complete list of members for **CountdownTimer**, including all inherited members.

Countdown (defined in **CountdownTimer**)

CountdownStartTime (defined in **CountdownTimer**)

CountdownTimerHasExpired()

OnConnected()

OnConnectedToMaster()

OnCountdownTimerHasExpired

OnCreatedRoom()

OnCreateRoomFailed(short returnCode, string message)

OnCustomAuthenticationFailed(string debugMessage)

OnCustomAuthenticationResponse(Dictionary< string, object > data)

OnDisable() (defined in **MonoBehaviourPunCallbacks**)

OnDisconnected(DisconnectCause cause)

OnEnable() (defined in **MonoBehaviourPunCallbacks**)

OnFriendListUpdate(List< FriendInfo > friendList)

OnJoinedLobby()

OnJoinedRoom()

OnJoinRandomFailed(short returnCode, string message)

OnJoinRoomFailed(short returnCode, string message)

OnLeftLobby()

OnLeftRoom()

OnLobbyStatisticsUpdate(List< TypedLobbyInfo > lobbyStatistics)

OnMasterClientSwitched(Player newMasterClient)

OnPlayerEnteredRoom(Player newPlayer)

OnPlayerLeftRoom(Player otherPlayer)

OnPlayerPropertiesUpdate(Player target, Hashtable changedProps)

OnRegionListReceived(RegionHandler regionHandler)

OnRoomListUpdate(List< RoomInfo > roomList)

OnRoomPropertiesUpdate(Hashtable propertiesThatChanged)

OnWebRpcResponse(OperationResponse response) (defined in **MonoclonView**)

Start() (defined in **CountdownTimer**)

Text (defined in **CountdownTimer**)

Update() (defined in **CountdownTimer**)



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	CullArea	

CullArea Member List

This is the complete list of members for **CullArea**, including all inherited members.

CellCount (defined in CullArea)	CullArea
CellTree (defined in CullArea)	CullArea
Center (defined in CullArea)	CullArea
FIRST_GROUP_ID	CullArea
GetActiveCells (Vector3 position)	CullArea
Map (defined in CullArea)	CullArea
MAX_NUMBER_OF_SUBDIVISIONS (defined in CullArea)	CullArea
NumberOfSubdivisions (defined in CullArea)	CullArea
OnDrawGizmos ()	CullArea
RecreateCellHierarchy (defined in CullArea)	CullArea
Size (defined in CullArea)	CullArea
SUBDIVISION_FIRST_LEVEL_ORDER	CullArea
SUBDIVISION_SECOND_LEVEL_ORDER	CullArea
SUBDIVISION_THIRD_LEVEL_ORDER	CullArea
Subdivisions (defined in CullArea)	CullArea
YIsUpAxis (defined in CullArea)	CullArea



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	CullingHandler	

CullingHandler Member List

This is the complete list of members for **CullingHandler**, including all inherited members.

OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	EventSystemSpawner	

EventSystemSpawner Member List

This is the complete list of members for **EventSystemSpawner**, including all inherited members.



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	GraphicToggleIsOnTransition	

GraphicToggleIsOnTransition Member List

This is the complete list of members for **GraphicToggleIsOnTransition**, including all inherited members.

HoverOffColor (defined in **GraphicToggleIsOnTransition**)
HoverOnColor (defined in **GraphicToggleIsOnTransition**)
NormalOffColor (defined in **GraphicToggleIsOnTransition**)
NormalOnColor (defined in **GraphicToggleIsOnTransition**)
OnDisable() (defined in **GraphicToggleIsOnTransition**)
OnEnable() (defined in **GraphicToggleIsOnTransition**)
OnPointerEnter(PointerEventData eventData) (defined in **GraphicToggleIsOnTransition**)
OnPointerExit(PointerEventData eventData) (defined in **GraphicToggleIsOnTransition**)
OnValueChanged(bool isOn) (defined in **GraphicToggleIsOnTransition**)
toggle (defined in **GraphicToggleIsOnTransition**)



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	IPunTurnManagerCallbacks			

IPunTurnManagerCallbacks Member List

This is the complete list of members for **IPunTurnManagerCallbacks**, including all inherited members.

OnPlayerFinished (Player player, int turn, object move)	IPunTurnMana
OnPlayerMove (Player player, int turn, object move)	IPunTurnMana
OnTurnBegins (int turn)	IPunTurnMana
OnTurnCompleted (int turn)	IPunTurnMana
OnTurnTimeEnds (int turn)	IPunTurnMana



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	MoveByKeys	

MoveByKeys Member List

This is the complete list of members for **MoveByKeys**, including all inherited members.

FixedUpdate() (defined in MoveByKeys)	MoveByKeys	
JumpForce (defined in MoveByKeys)	MoveByKeys	
JumpTimeout (defined in MoveByKeys)	MoveByKeys	
photonView	MonoBehaviourPun	
Speed (defined in MoveByKeys)	MoveByKeys	
Start() (defined in MoveByKeys)	MoveByKeys	



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	OnClickDestroy	

OnClickDestroy Member List

This is the complete list of members for **OnClickDestroy**, including all inherited members.

Button (defined in OnClickDestroy)	OnClickDestroy	
DestroyByRpc (defined in OnClickDestroy)	OnClickDestroy	
DestroyRpc() (defined in OnClickDestroy)	OnClickDestroy	
ModifierKey (defined in OnClickDestroy)	OnClickDestroy	
photonView	MonoBehaviourPun	



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	OnClickInstantiate	

OnClickInstantiate Member List

This is the complete list of members for **OnClickInstantiate**, including all inherited members.

Button (defined in OnClickInstantiate)	OnClick
InstantiateOption enum name (defined in OnClickInstantiate)	OnClick
ModifierKey (defined in OnClickInstantiate)	OnClick
Prefab (defined in OnClickInstantiate)	OnClick



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	OnClickRpc	

OnClickRpc Member List

This is the complete list of members for **OnClickRpc**, including all inherited members.

Button (defined in OnClickRpc)	OnClickRpc	
ClickFlash() (defined in OnClickRpc)	OnClickRpc	
ClickRpc() (defined in OnClickRpc)	OnClickRpc	
ModifierKey (defined in OnClickRpc)	OnClickRpc	
photonView	MonoBehaviourPun	
Target (defined in OnClickRpc)	OnClickRpc	



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	OnEscapeQuit	

OnEscapeQuit Member List

This is the complete list of members for **OnEscapeQuit**, including all inherited members.

Update() (defined in **OnEscapeQuit**) **OnEscapeQuit**



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	OnJoinedInstantiate	

OnJoinedInstantiate Member List

This is the complete list of members for **OnJoinedInstantiate**, including all inherited members.

OnConnected()

OnConnectedToMaster()

OnCreatedRoom()

OnCreateRoomFailed(short returnCode, string message)

OnCustomAuthenticationFailed(string debugMessage)

OnCustomAuthenticationResponse(Dictionary< string, object > data)

OnDisable() (defined in **OnJoinedInstantiate**)

OnDisconnected(DisconnectCause cause)

OnEnable() (defined in **OnJoinedInstantiate**)

OnFriendListUpdate(List< FriendInfo > friendList)

OnJoinedLobby()

OnJoinedRoom()

OnJoinRandomFailed(short returnCode, string message)

OnJoinRoomFailed(short returnCode, string message)

OnLeftLobby()

OnLeftRoom()

OnLobbyStatisticsUpdate(List< TypedLobbyInfo > lobbyStatistics)

OnRegionListReceived(RegionHandler regionHandler)

OnRoomListUpdate(List< RoomInfo > roomList)

PositionOffset (defined in **OnJoinedInstantiate**)

PrefabsToInstantiate (defined in **OnJoinedInstantiate**)

SpawnPosition (defined in **OnJoinedInstantiate**)

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	OnPointerOverTooltip	

OnPointerOverTooltip Member List

This is the complete list of members for **OnPointerOverTooltip**, including all inherited members.



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	OnStartDelete			

OnStartDelete Member List

This is the complete list of members for **OnStartDelete**, including all inherited members.



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy		Class Members
Photon	Pun	UtilityScripts	PlayerNumbering			

PlayerNumbering Member List

This is the complete list of members for **PlayerNumbering**, including all inherited members.

Awake() (defined in **PlayerNumbering**)

dontDestroyOnLoad

instance

OnConnected()

OnConnectedToMaster()

OnCreatedRoom()

OnCreateRoomFailed(short returnCode, string message)

OnCustomAuthenticationFailed(string debugMessage)

OnCustomAuthenticationResponse(Dictionary< string, object > data)

OnDisable() (defined in **MonoBehaviourPunCallbacks**)

OnDisconnected(DisconnectCause cause)

OnEnable() (defined in **MonoBehaviourPunCallbacks**)

OnFriendListUpdate(List< FriendInfo > friendList)

OnJoinedLobby()

OnJoinedRoom()

OnJoinRandomFailed(short returnCode, string message)

OnJoinRoomFailed(short returnCode, string message)

OnLeftLobby()

OnLeftRoom()

OnLobbyStatisticsUpdate(List< TypedLobbyInfo > lobbyStatistics)

OnMasterClientSwitched(Player newMasterClient)

OnPlayerEnteredRoom(Player newPlayer)

OnPlayerLeftRoom(Player otherPlayer)

OnPlayerNumberingChanged

OnPlayerPropertiesUpdate(Player targetPlayer, Hashtable changedProp)

OnRegionListReceived(RegionHandler regionHandler)

OnRoomListUpdate(List< RoomInfo > roomList)

OnRoomPropertiesUpdate(Hashtable propertiesThatChanged)

OnWebRpcResponse(OperationResponse response) (defined in **Monocloner**)
photonView

PlayerNumberingChanged()

RefreshData()

RoomPlayerIndexedProp

SortedPlayers (defined in **PlayerNumbering**)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	PlayerNumberingExtensions	

PlayerNumberingExtensions Member List

This is the complete list of members for **PlayerNumberingExtensions**, including all inherited members.

GetPlayerNumber(this Player player) **PlayerNumber**

SetPlayerNumber(this Player player, int playerNumber) **PlayerNumber**



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	PointedAtGameObjectInfo	

PointedAtGameObjectInfo Member List

This is the complete list of members for **PointedAtGameObjectInfo**, including all inherited members.

Instance (defined in **PointedAtGameObjectInfo**)

RemoveFocus(PhotonView pv) (defined in **PointedAtGameObjectInfo**)

SetFocus(PhotonView pv) (defined in **PointedAtGameObjectInfo**)

text (defined in **PointedAtGameObjectInfo**)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	PunPlayerScores	

PunPlayerScores Member List

This is the complete list of members for **PunPlayerScores**, including all inherited members.

PlayerScoreProp (defined in **PunPlayerScores**) **PunPlayerScores**



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	PunTeams	

PunTeams Member List

This is the complete list of members for **PunTeams**, including all inherited members.

OnConnected()

OnConnectedToMaster()

OnCreatedRoom()

OnCreateRoomFailed(short returnCode, string message)

OnCustomAuthenticationFailed(string debugMessage)

OnCustomAuthenticationResponse(Dictionary< string, object > data)

OnDisable() (defined in **PunTeams**)

OnDisconnected(DisconnectCause cause)

OnEnable() (defined in **MonoBehaviourPunCallbacks**)

OnFriendListUpdate(List< FriendInfo > friendList)

OnJoinedLobby()

OnJoinedRoom()

OnJoinRandomFailed(short returnCode, string message)

OnJoinRoomFailed(short returnCode, string message)

OnLeftLobby()

OnLeftRoom()

OnLobbyStatisticsUpdate(List< TypedLobbyInfo > lobbyStatistics)

OnMasterClientSwitched(Player newMasterClient)

OnPlayerEnteredRoom(Player newPlayer)

OnPlayerLeftRoom(Player otherPlayer)

OnPlayerPropertiesUpdate(Player targetPlayer, Hashtable changedPr
OnRegionListReceived(RegionHandler regionHandler)
OnRoomListUpdate(List< RoomInfo > roomList)
OnRoomPropertiesUpdate(Hashtable propertiesThatChanged)
OnWebRpcResponse(OperationResponse response) (defined in **Monoc
photonView**
PlayersPerTeam
Start() (defined in **PunTeams**)
Team enum name
TeamPlayerProp
UpdateTeams() (defined in **PunTeams**)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	PunTurnManager	

PunTurnManager Member List

This is the complete list of members for **PunTurnManager**, including all inherited members.

BeginTurn()

ElapsedTimeInTurn

EvFinalMove

EvMove

GetPlayerFinishedTurn(Player player)

IsCompletedByAll

IsFinishedByMe

IsOver

OnConnected()

OnConnectedToMaster()

OnCreatedRoom()

OnCreateRoomFailed(short returnCode, string message)

OnCustomAuthenticationFailed(string debugMessage)

OnCustomAuthenticationResponse(Dictionary< string, object > data)

OnDisable() (defined in **MonoBehaviourPunCallbacks**)

OnDisconnected(DisconnectCause cause)

OnEnable() (defined in **MonoBehaviourPunCallbacks**)

OnEvent(EventData photonEvent)

OnFriendListUpdate(List< FriendInfo > friendList)

OnJoinedLobby()

OnJoinedRoom()

OnJoinRandomFailed(short returnCode, string message)

OnJoinRoomFailed(short returnCode, string message)

OnLeftLobby()

OnLeftRoom()

OnLobbyStatisticsUpdate(List< TypedLobbyInfo > lobbyStatistics)

OnMasterClientSwitched(Player newMasterClient)

OnPlayerEnteredRoom(Player newPlayer)

OnPlayerLeftRoom(Player otherPlayer)

OnPlayerPropertiesUpdate(Player target, Hashtable changedProps)

OnRegionListReceived(RegionHandler regionHandler)

OnRoomListUpdate(List< RoomInfo > roomList)

OnRoomPropertiesUpdate(Hashtable propertiesThatChanged)

OnWebRpcResponse(OperationResponse response) (defined in **MonoclonView**)

RemainingSecondsInTurn

SendMove(object move, bool finished)

Turn

TurnDuration

TurnManagerEventOffset

TurnManagerListener



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List	Class Index	Class Hierarchy		Class Members		
Photon	Pun	UtilityScripts	ScoreExtensions			

ScoreExtensions Member List

This is the complete list of members for **ScoreExtensions**, including all inherited members.

AddScore(this Player player, int scoreToAddToCurrent) (defined in **ScoreExtensions**)

GetScore(this Player player) (defined in **ScoreExtensions**)

SetScore(this Player player, int newScore) (defined in **ScoreExtensions**)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	SmoothSyncMovement	

SmoothSyncMovement Member List

This is the complete list of members for **SmoothSyncMovement**, including all inherited members.

Awake() (defined in **SmoothSyncMovement**)

OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info, PhotonView photonView)

SmoothingDelay (defined in **SmoothSyncMovement**)

Update() (defined in **SmoothSyncMovement**)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	StatesGui	

StatesGui Member List

This is the complete list of members for **StatesGui**, including all inherited members.

AppVersion (defined in StatesGui)	StatesGui
Buttons (defined in StatesGui)	StatesGui
DetailedConnection (defined in StatesGui)	StatesGui
DontDestroy (defined in StatesGui)	StatesGui
ExpectedUsers (defined in StatesGui)	StatesGui
GuiOffset (defined in StatesGui)	StatesGui
LocalPlayer (defined in StatesGui)	StatesGui
Others (defined in StatesGui)	StatesGui
PlayerProps (defined in StatesGui)	StatesGui
Room (defined in StatesGui)	StatesGui
RoomProps (defined in StatesGui)	StatesGui
Server (defined in StatesGui)	StatesGui
ServerTimestamp (defined in StatesGui)	StatesGui
UserId (defined in StatesGui)	StatesGui



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	TabViewManager			

TabViewManager Member List

This is the complete list of members for **TabViewManager**, including all inherited members.

CurrentTab (defined in TabViewManager)	TabViewManager	protected
OnTabChanged	TabViewManager	
SelectTab (string id)	TabViewManager	
Tabs	TabViewManager	
ToggleGroup	TabViewManager	



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	TabViewManager	Tab		

TabViewManager.Tab Member List

This is the complete list of members for **TabViewManager.Tab**, including all inherited members.

ID (defined in TabViewManager.Tab)	TabViewManager.Tab
Toggle (defined in TabViewManager.Tab)	TabViewManager.Tab
View (defined in TabViewManager.Tab)	TabViewManager.Tab



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	TeamExtensions	

TeamExtensions Member List

This is the complete list of members for **TeamExtensions**, including all inherited members.

GetTeam(this Player player) **TeamExtensions**

SetTeam(this Player player, PunTeams.Team team) **TeamExtensions**



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	TextButtonTransition	

TextButtonTransition Member List

This is the complete list of members for **TextButtonTransition**, including all inherited members.

Awake() (defined in **TextButtonTransition**)

HoverColor

NormalColor

OnDisable() (defined in **TextButtonTransition**)

OnEnable() (defined in **TextButtonTransition**)

OnPointerEnter(PointerEventData eventData) (defined in **TextButtonTransition**)

OnPointerExit(PointerEventData eventData) (defined in **TextButtonTransition**)

Selectable



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	TextToggleIsOnTransition	

TextToggleIsOnTransition Member List

This is the complete list of members for **TextToggleIsOnTransition**, including all inherited members.

HoverOffColor

HoverOnColor

NormalOffColor

NormalOnColor

OnDisable() (defined in **TextToggleIsOnTransition**)

OnEnable() (defined in **TextToggleIsOnTransition**)

OnPointerEnter(PointerEventData eventData) (defined in **TextToggleIsOnTransition**)

OnPointerExit(PointerEventData eventData) (defined in **TextToggleIsOnTransition**)

OnValueChanged(bool isOn) (defined in **TextToggleIsOnTransition**)

toggle



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	UtilityScripts	TurnExtensions	

TurnExtensions Member List

This is the complete list of members for **TurnExtensions**, including all inherited members.

FinishedTurnPropKey	TurnExtens
GetFinishedTurn (this Player player)	TurnExtens
GetTurn (this RoomInfo room)	TurnExtens
GetTurnStart (this RoomInfo room)	TurnExtens
SetFinishedTurn (this Player player, int turn)	TurnExtens
SetTurn (this Room room, int turn, bool setStartTime=false)	TurnExtens
TurnPropKey	TurnExtens
TurnStartPropKey	TurnExtens



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	DefaultPool		

DefaultPool Member List

This is the complete list of members for **DefaultPool**, including all inherited members.

Destroy (GameObject gameObject)	DefaultPool
Instantiate (string prefabId, Vector3 position, Quaternion rotation)	DefaultPool
ResourceCache	DefaultPool



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	InstantiateParameters		

InstantiateParameters Member List

This is the complete list of members for **InstantiateParameters**, including all inherited members.

creator (defined in **InstantiateParameters**)

data (defined in **InstantiateParameters**)

group (defined in **InstantiateParameters**)

InstantiateParameters(string prefabName, Vector3 position, Quaternion

objLevelPrefix (defined in **InstantiateParameters**)

position (defined in **InstantiateParameters**)

prefabName (defined in **InstantiateParameters**)

rotation (defined in **InstantiateParameters**)

timestamp (defined in **InstantiateParameters**)

viewIDs (defined in **InstantiateParameters**)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	IPunPrefabPool		

IPunPrefabPool Member List

This is the complete list of members for **IPunPrefabPool**, including all inherited members.

Destroy(GameObject gameObject) **IPunF**

Instantiate(string prefabId, Vector3 position, Quaternion rotation) **IPunF**



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	MonoBehaviourPun		

MonoBehaviourPun Member List

This is the complete list of members for **MonoBehaviourPun**, including all inherited members.

[photonView](#) [MonoBehaviourPun](#)



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonAnimatorView		

PhotonAnimatorView Member List

This is the complete list of members for **PhotonAnimatorView**, including all inherited members.

CacheDiscreteTriggers()

DoesLayerSynchronizeTypeExist(int layerIndex)

DoesParameterSynchronizeTypeExist(string name)

GetLayerSynchronizeType(int layerIndex)

GetParameterSynchronizeType(string name)

GetSynchronizedLayers()

GetSynchronizedParameters()

OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)

ParameterType enum name (defined in **PhotonAnimatorView**)

SetLayerSynchronized(int layerIndex, SynchronizeType synchronizeType)

SetParameterSynchronized(string name, ParameterType type, SynchronizeType synchronizeType)

SynchronizeType enum name (defined in **PhotonAnimatorView**)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonAnimatorView	SynchronizedLayer	

PhotonAnimatorView.SynchronizedLayer Member List

This is the complete list of members for **PhotonAnimatorView.SynchronizedLayer**, including all inherited members.

LayerIndex (defined in **PhotonAnimatorView.SynchronizedLayer**)

SynchronizeType (defined in **PhotonAnimatorView.SynchronizedLay**)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonAnimatorView	SynchronizedParameter	

PhotonAnimatorView.SynchronizedParameter Member List

This is the complete list of members for **PhotonAnimatorView.SynchronizedParameter**, including all inherited members.

Name (defined in **PhotonAnimatorView.SynchronizedParameter**)

SynchronizeType (defined in **PhotonAnimatorView.SynchronizedParameter**)

Type (defined in **PhotonAnimatorView.SynchronizedParameter**)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonRigidbody2DView		

PhotonRigidbody2DView Member List

This is the complete list of members for **PhotonRigidbody2DView**, including all inherited members.

Awake() (defined in **PhotonRigidbody2DView**)

FixedUpdate() (defined in **PhotonRigidbody2DView**)

m_SynchronizeAngularVelocity (defined in **PhotonRigidbody2DView**)

m_SynchronizeVelocity (defined in **PhotonRigidbody2DView**)

m_TeleportEnabled (defined in **PhotonRigidbody2DView**)

m_TeleportIfDistanceGreaterThan (defined in **PhotonRigidbody2DView**)

OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonRigidbodyView		

PhotonRigidbodyView Member List

This is the complete list of members for **PhotonRigidbodyView**, including all inherited members.

Awake() (defined in **PhotonRigidbodyView**)

FixedUpdate() (defined in **PhotonRigidbodyView**)

m_SynchronizeAngularVelocity (defined in **PhotonRigidbodyView**)

m_SynchronizeVelocity (defined in **PhotonRigidbodyView**)

m_TeleportEnabled (defined in **PhotonRigidbodyView**)

m_TeleportIfDistanceGreaterThan (defined in **PhotonRigidbodyView**)

OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonStreamQueue		

PhotonStreamQueue Member List

This is the complete list of members for **PhotonStreamQueue**, including all inherited members.

Deserialize (PhotonStream stream)	PhotonStreamQueue
HasQueuedObjects ()	PhotonStreamQueue
PhotonStreamQueue (int sampleRate)	PhotonStreamQueue
ReceiveNext ()	PhotonStreamQueue
Reset ()	PhotonStreamQueue
SendNext (object obj)	PhotonStreamQueue
Serialize (PhotonStream stream)	PhotonStreamQueue



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonTransformView		

PhotonTransformView Member List

This is the complete list of members for **PhotonTransformView**, including all inherited members.

Awake() (defined in **PhotonTransformView**)

m_SynchronizePosition (defined in **PhotonTransformView**)

m_SynchronizeRotation (defined in **PhotonTransformView**)

m_SynchronizeScale (defined in **PhotonTransformView**)

OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)

Update() (defined in **PhotonTransformView**)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonTransformViewClassic		

PhotonTransformViewClassic Member List

This is the complete list of members for **PhotonTransformViewClassic**, including all inherited members.

m_PositionModel (defined in **PhotonTransformViewClassic**)

m_RotationModel (defined in **PhotonTransformViewClassic**)

m_ScaleModel (defined in **PhotonTransformViewClassic**)

OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)

SetSynchronizedValues(Vector3 speed, float turnSpeed)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonTransformViewPositionControl		

PhotonTransformViewPositionControl Member List

This is the complete list of members for **PhotonTransformViewPositionControl**, including all inherited members.

GetExtrapolatedPositionOffset()

GetNetworkPosition()

OnPhotonSerializeView(Vector3 currentPosition, PhotonStream stream)

PhotonTransformViewPositionControl(PhotonTransformViewPositionControl)

SetSynchronizedValues(Vector3 speed, float turnSpeed)

UpdatePosition(Vector3 currentPosition)



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonTransformViewPositionModel		

PhotonTransformViewPositionModel Member List

This is the complete list of members for **PhotonTransformViewPositionModel**, including all inherited members.

ExtrapolateIncludingRoundTripTime (defined in **PhotonTransformViewPositionModel**)
ExtrapolateNumberOfStoredPositions (defined in **PhotonTransformViewPositionModel**)
ExtrapolateOption (defined in **PhotonTransformViewPositionModel**)
ExtrapolateOptions enum name (defined in **PhotonTransformViewPositionModel**)
ExtrapolateSpeed (defined in **PhotonTransformViewPositionModel**)
InterpolateLerpSpeed (defined in **PhotonTransformViewPositionModel**)
InterpolateMoveTowardsSpeed (defined in **PhotonTransformViewPositionModel**)
InterpolateOption (defined in **PhotonTransformViewPositionModel**)
InterpolateOptions enum name (defined in **PhotonTransformViewPositionModel**)
SynchronizeEnabled (defined in **PhotonTransformViewPositionModel**)
TeleportEnabled (defined in **PhotonTransformViewPositionModel**)
TeleportIfDistanceGreaterThan (defined in **PhotonTransformViewPositionModel**)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonTransformViewRotationControl		

PhotonTransformViewRotationControl Member List

This is the complete list of members for **PhotonTransformViewRotationControl**, including all inherited members.

GetNetworkRotation()

GetRotation(Quaternion currentRotation) (defined in **PhotonTransform**

OnPhotonSerializeView(Quaternion currentRotation, PhotonStream str

PhotonTransformViewRotationControl(PhotonTransformViewRotatior



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonTransformViewRotationModel		

PhotonTransformViewRotationModel Member List

This is the complete list of members for **PhotonTransformViewRotationModel**, including all inherited members.

InterpolateLerpSpeed (defined in **PhotonTransformViewRotationModel**)
InterpolateOption (defined in **PhotonTransformViewRotationModel**)
InterpolateOptions enum name (defined in **PhotonTransformViewRotationModel**)
InterpolateRotateTowardsSpeed (defined in **PhotonTransformViewRotationModel**)
SynchronizeEnabled (defined in **PhotonTransformViewRotationModel**)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PhotonTransformViewScaleControl		

PhotonTransformViewScaleControl Member List

This is the complete list of members for **PhotonTransformViewScaleControl**, including all inherited members.

GetNetworkScale()

GetScale(Vector3 currentScale) (defined in **PhotonTransformViewSca**

OnPhotonSerializeView(Vector3 currentScale, PhotonStream stream, I

PhotonTransformViewScaleControl(PhotonTransformViewScaleMode



Photon Unity Networking 2 2.12

Main Page		Related Pages		Modules	Classes	
Class List		Class Index		Class Hierarchy		Class Members
Photon	Pun	PhotonTransformViewScaleModel				

PhotonTransformViewScaleModel Member List

This is the complete list of members for **PhotonTransformViewScaleModel**, including all inherited members.

InterpolateLerpSpeed (defined in **PhotonTransformViewScaleModel**)

InterpolateMoveTowardsSpeed (defined in **PhotonTransformViewScaleModel**)

InterpolateOption (defined in **PhotonTransformViewScaleModel**)

InterpolateOptions enum name (defined in **PhotonTransformViewScaleModel**)

SynchronizeEnabled (defined in **PhotonTransformViewScaleModel**)



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	PunExtensions		

PunExtensions Member List

This is the complete list of members for **PunExtensions**, including all inherited members.

AlmostEquals(this Vector3 target, Vector3 second, float sqrMagnitudeP

AlmostEquals(this Vector2 target, Vector2 second, float sqrMagnitudeP

AlmostEquals(this Quaternion target, Quaternion second, float maxAng

AlmostEquals(this float target, float second, float floatDiff)

GetCachedParemers(this MethodInfo mo) (defined in **PunExtension**

GetPhotonView(this UnityEngine.GameObject go) (defined in **PunExte**

GetPhotonViewsInChildren(this UnityEngine.GameObject go) (defined

ParametersOfMethods (defined in **PunExtensions**)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	SceneManagerHelper		

SceneManagerHelper Member List

This is the complete list of members for **SceneManagerHelper**, including all inherited members.

ActiveSceneBuildIndex (defined in SceneManagerHelper)	SceneMar
ActiveSceneName (defined in SceneManagerHelper)	SceneMar



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Pun	ServerSettings		

ServerSettings Member List

This is the complete list of members for **ServerSettings**, including all inherited members.

AppSettings (defined in ServerSettings)	ServerSettings	
BestRegionSummaryInPreferences	ServerSettings	st
EnableSupportLogger (defined in ServerSettings)	ServerSettings	
IsAppId (string val)	ServerSettings	st
PunLogging (defined in ServerSettings)	ServerSettings	
ResetBestRegionCodeInPreferences ()	ServerSettings	st
RpcList (defined in ServerSettings)	ServerSettings	
RunInBackground (defined in ServerSettings)	ServerSettings	
StartInOfflineMode (defined in ServerSettings)	ServerSettings	
ToString ()	ServerSettings	
UseCloud (string cloudAppid, string code="")	ServerSettings	



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	ActorProperties		

ActorProperties Member List

This is the complete list of members for **ActorProperties**, including all inherited members.

IsInactive **ActorProperties**

PlayerName **ActorProperties**

UserId **ActorProperties**



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	AppSettings		

AppSettings Member List

This is the complete list of members for **AppSettings**, including all inherited members.

AppIdChat	AppSettings
AppIdRealtime	AppSettings
AppIdVoice	AppSettings
AppVersion	AppSettings
EnableLobbyStatistics	AppSettings
FixedRegion	AppSettings
IsBestRegion	AppSettings
IsDefaultNameServer	AppSettings
IsDefaultPort	AppSettings
IsMasterServerAddress	AppSettings
NetworkLogging	AppSettings
Port	AppSettings
Protocol	AppSettings
Server	AppSettings
ToStringFull()	AppSettings
UseNameServer	AppSettings



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	AuthenticationValues		

AuthenticationValues Member List

This is the complete list of members for **AuthenticationValues**, including all inherited members.

AddAuthParameter (string key, string value)	Authentication
AuthenticationValues ()	Authentication
AuthenticationValues (string userId)	Authentication
AuthGetParameters	Authentication
AuthPostData	Authentication
AuthType	Authentication
SetAuthPostData (string stringData)	Authentication
SetAuthPostData (byte[] byteData)	Authentication
SetAuthPostData (Dictionary< string, object > dictData)	Authentication
Token	Authentication
ToString () (defined in AuthenticationValues)	Authentication
UserId	Authentication



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	ConnectionCallbacksContainer		

ConnectionCallbacksContainer Member List

This is the complete list of members for **ConnectionCallbacksContainer**, including all inherited members.

AddCallbackTarget(IConnectionCallbacks target) (defined in **ConnectionCallbacksContainer**)

ConnectionCallbacksContainer() (defined in **ConnectionCallbacksContainer**)

OnConnected()

OnConnectedToMaster()

OnCustomAuthenticationFailed(string debugMessage)

OnCustomAuthenticationResponse(Dictionary< string, object > data)

OnDisconnected(DisconnectCause cause)

OnRegionListReceived(RegionHandler regionHandler)

RemoveCallbackTarget(IConnectionCallbacks target) (defined in **ConnectionCallbacksContainer**)



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	ConnectionHandler		

ConnectionHandler Member List

This is the complete list of members for **ConnectionHandler**, including all inherited members.

Client	Con
CountSendAcksOnly	Con
FallbackThreadRunning (defined in ConnectionHandler)	Con
KeepAliveInBackground	Con
RealtimeFallbackThread()	Con
StartFallbackSendAckThread() (defined in ConnectionHandler)	Con
StopFallbackSendAckThread() (defined in ConnectionHandler)	Con



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	EncryptionDataParameters		

EncryptionDataParameters Member List

This is the complete list of members for **EncryptionDataParameters**, including all inherited members.

Mode **EncryptionDataParameters**

Secret1 **EncryptionDataParameters**

Secret2 **EncryptionDataParameters**



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	EnterRoomParams		

EnterRoomParams Member List

This is the complete list of members for **EnterRoomParams**, including all inherited members.

CreateIfNotExists (defined in EnterRoomParams)	EnterRoomParam
ExpectedUsers (defined in EnterRoomParams)	EnterRoomParam
Lobby (defined in EnterRoomParams)	EnterRoomParam
PlayerProperties (defined in EnterRoomParams)	EnterRoomParam
RejoinOnly (defined in EnterRoomParams)	EnterRoomParam
RoomName (defined in EnterRoomParams)	EnterRoomParam
RoomOptions (defined in EnterRoomParams)	EnterRoomParam



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	ErrorCode		

ErrorCode Member List

This is the complete list of members for **ErrorCode**, including all inherited members.

AlreadyMatched (defined in ErrorCode)	ErrorCode
AuthenticationTicketExpired	ErrorCode
CustomAuthenticationFailed	ErrorCode
ExternalHttpCallFailed	ErrorCode
GameClosed	ErrorCode
GameDoesNotExist	ErrorCode
GameFull	ErrorCode
GameIdAlreadyExists	ErrorCode
HttpLimitReached	ErrorCode
InternalServerError	ErrorCode
InvalidAuthentication	ErrorCode
InvalidEncryptionParameters	ErrorCode
InvalidOperation	ErrorCode
InvalidOperationCode	ErrorCode
InvalidRegion	ErrorCode
JoinFailedFoundActiveJoiner	ErrorCode
JoinFailedFoundExcludedUserId	ErrorCode
JoinFailedFoundInactiveJoiner	ErrorCode
JoinFailedPeerAlreadyJoined	ErrorCode
JoinFailedWithRejoinerNotFound	ErrorCode

MaxCcuReached	ErrorCode
NoRandomMatchFound	ErrorCode
Ok	ErrorCode
OperationNotAllowedInCurrentState	ErrorCode
PluginMismatch	ErrorCode
PluginReportedError	ErrorCode
ServerFull	ErrorCode
SlotError	ErrorCode
UserBlocked	ErrorCode



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	EventCode		

EventCode Member List

This is the complete list of members for **EventCode**, including all inherited members.

AppStats	EventCode
AuthEvent	EventCode
AzureNodeInfo	EventCode
CacheSliceChanged	EventCode
ErrorInfo	EventCode
GameList	EventCode
GameListUpdate	EventCode
Join	EventCode
Leave	EventCode
LobbyStats	EventCode
Match	EventCode
PropertiesChanged	EventCode
QueueState	EventCode
SetProperties	EventCode



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	EventExt		

EventExt Member List

This is the complete list of members for **EventExt**, including all inherited members.

CustomData (this EventData ev) (defined in EventExt)	EventExt	static
Sender (this EventData ev) (defined in EventExt)	EventExt	static



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	Extensions		

Extensions Member List

This is the complete list of members for **Extensions**, including all inherited members.

Contains (this int[] target, int nr)	Extensi
Merge (this IDictionary target, IDictionary addHash)	Extensi
MergeStringKeys (this IDictionary target, IDictionary addHash)	Extensi
StripKeysWithNullValues (this IDictionary original)	Extensi
StripToStringKeys (this IDictionary original)	Extensi
ToStringFull (this IDictionary origin)	Extensi
ToStringFull (this object[] data)	Extensi
ToStringFull < T >(this List< T > data)	Extensi



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	FriendInfo		

FriendInfo Member List

This is the complete list of members for **FriendInfo**, including all inherited members.

IsInRoom (defined in FriendInfo)	FriendInfo	
IsOnline (defined in FriendInfo)	FriendInfo	
Name (defined in FriendInfo)	FriendInfo	
Room (defined in FriendInfo)	FriendInfo	
ToString() (defined in FriendInfo)	FriendInfo	
UserId (defined in FriendInfo)	FriendInfo	



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	GamePropertyKey		

GamePropertyKey Member List

This is the complete list of members for **GamePropertyKey**, including all inherited members.

CleanupCacheOnLeave	GamePropertyKey
EmptyRoomTtl	GamePropertyKey
ExpectedUsers	GamePropertyKey
IsOpen	GamePropertyKey
IsVisible	GamePropertyKey
MasterClientId	GamePropertyKey
MaxPlayers	GamePropertyKey
PlayerCount	GamePropertyKey
PlayerTtl	GamePropertyKey
PropsListedInLobby	GamePropertyKey
Removed	GamePropertyKey



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	LoadBalancingClient		

LoadBalancingClient Member List

This is the complete list of members for **LoadBalancingClient**, including all inherited members.

AddCallbackTarget(object target)

AppId

AppVersion

AuthMode

AuthValues

ChangeLocalID(int newID)

CloudRegion

Connect()

ConnectionCallbackTargets

ConnectToNameServer()

ConnectToRegionMaster(string region)

CurrentLobby

CurrentRoom

CurrentServerAddress

DebugReturn(DebugLevel level, string message)

Disconnect()

DisconnectedCause

EnableLobbyStatistics

EncryptionMode

EventReceived

ExpectedProtocol

GameServerAddress

InLobby

InRoom

IsConnected

IsConnectedAndReady

IsFetchingFriendList

IsUsingNameServer

LoadBalancingClient(ConnectionProtocol protocol=ConnectionProtoco

LoadBalancingClient(string masterAddress, string appld, string gameV

LoadBalancingPeer

LocalPlayer

MasterServerAddress

MatchMakingCallbackTargets

NameServerAddress

NameServerHost

NameServerHttp

NickName

OnEvent(EventData photonEvent)

OnMessage(object message)

OnOperationResponse(OperationResponse operationResponse)

OnStatusChanged(StatusCode statusCode)

OpChangeGroups(byte[] groupsToRemove, byte[] groupsToAdd)

OpCreateRoom(EnterRoomParams enterRoomParams)

OpFindFriends(string[] friendsToFind)

OpGetGameList(TypedLobby typedLobby, string sqlLobbyFilter)

OpJoinLobby(TypedLobby lobby)

OpJoinOrCreateRoom(EnterRoomParams enterRoomParams)

OpJoinRandomRoom(OpJoinRandomRoomParams opJoinRandomRo

OpJoinRoom(EnterRoomParams enterRoomParams)

OpLeaveLobby()

OpLeaveRoom(bool becomeInactive, bool sendAuthCookie=false)

OpRaiseEvent(byte eventCode, object customEventContent, RaiseEvent)

OpRejoinRoom(string roomName)

OpResponseReceived

OpSetCustomPropertiesOfActor(int actorNr, Hashtable propertiesToSet)

OpSetCustomPropertiesOfRoom(Hashtable propertiesToSet, Hashtable)

OpSetPropertiesOfRoom(Hashtable gameProperties, Hashtable expected)

OpWebRpc(string uriPath, object parameters, bool sendAuthCookie=false)

PlayersInRoomsCount

PlayersOnMasterCount

ReconnectAndRejoin()

ReconnectToMaster()

RegionHandler

RemoveCallbackTarget(object target)

RoomsCount

Server

Service()

State

StateChanged

UseAlternativeUdpPorts

UserId



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	LoadBalancingPeer		

LoadBalancingPeer Member List

This is the complete list of members for **LoadBalancingPeer**, including all inherited members.

LoadBalancingPeer(ConnectionProtocol protocolType)

LoadBalancingPeer(IPhotonPeerListener listener, ConnectionProtocol

OpAuthenticate(string appld, string appVersion, AuthenticationValues a

OpAuthenticateOnce(string appld, string appVersion, AuthenticationVa

OpChangeGroups(byte[] groupsToRemove, byte[] groupsToAdd)

OpCreateRoom(EnterRoomParams opParams)

OpFindFriends(string[] friendsToFind)

OpGetGameList(TypedLobby lobby, string queryData)

OpGetRegions(string appld) (defined in **LoadBalancingPeer**)

OpJoinLobby(TypedLobby lobby=null)

OpJoinRandomRoom(OpJoinRandomRoomParams opJoinRandomRo

OpJoinRoom(EnterRoomParams opParams)

OpLeaveLobby()

OpLeaveRoom(bool becomeInactive, bool sendAuthCookie=false)

OpRaiseEvent(byte eventCode, object customEventContent, RaiseEve

OpSetCustomPropertiesOfActor(int actorNr, Hashtable actorPropertie

OpSetCustomPropertiesOfRoom(Hashtable gameProperties) (defined

OpSetPropertyOfRoom(byte propCode, object value) (defined in **Load**

OpSettings(bool receiveLobbyStats)

[Online Documentation](#) - [Dashboard](#) - [Support Forum](#)

Exit Games GmbH



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	MatchMakingCallbacksContainer		

MatchMakingCallbacksContainer Member List

This is the complete list of members for **MatchMakingCallbacksContainer**, including all inherited members.

AddCallbackTarget(IMatchmakingCallbacks target) (defined in **MatchMakingCallbacksContainer**)

MatchMakingCallbacksContainer() (defined in **MatchMakingCallbacksContainer**)

OnCreatedRoom()

OnCreateRoomFailed(short returnCode, string message)

OnFriendListUpdate(List< FriendInfo > friendList)

OnJoinedRoom()

OnJoinRandomFailed(short returnCode, string message)

OnJoinRoomFailed(short returnCode, string message)

OnLeftRoom()

RemoveCallbackTarget(IMatchmakingCallbacks target) (defined in **MatchMakingCallbacksContainer**)



Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	OperationCode		

OperationCode Member List

This is the complete list of members for **OperationCode**, including all inherited members.

Authenticate	Operation
AuthenticateOnce	Operation
ChangeGroups	Operation
CreateGame	Operation
ExchangeKeysForEncryption (defined in OperationCode)	Operation
FindFriends	Operation
GetGameList	Operation
GetLobbyStats	Operation
GetProperties	Operation
GetRegions	Operation
Join	Operation
JoinGame	Operation
JoinLobby	Operation
JoinRandomGame	Operation
Leave	Operation
LeaveLobby	Operation
RaiseEvent	Operation
ServerSettings	Operation
SetProperties	Operation
WebRpc	Operation



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	OpJoinRandomRoomParams		

OpJoinRandomRoomParams Member List

This is the complete list of members for **OpJoinRandomRoomParams**, including all inherited members.

ExpectedCustomRoomProperties (defined in **OpJoinRandomRoomParams**)

ExpectedMaxPlayers (defined in **OpJoinRandomRoomParams**)

ExpectedUsers (defined in **OpJoinRandomRoomParams**)

MatchingType (defined in **OpJoinRandomRoomParams**)

SqlLobbyFilter (defined in **OpJoinRandomRoomParams**)

TypedLobby (defined in **OpJoinRandomRoomParams**)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	ParameterCode		

ParameterCode Member List

This is the complete list of members for **ParameterCode**, including all inherited members.

ActorList	ParameterCode
ActorNr	ParameterCode
Add	ParameterCode
Address	ParameterCode
ApplicationId	ParameterCode
AppVersion	ParameterCode
AzureLocalNodeId	ParameterCode
AzureMasterNodeId	ParameterCode
AzureNodeInfo	ParameterCode
Broadcast	ParameterCode
Cache	ParameterCode
CacheSliceIndex	ParameterCode
CheckUserOnJoin	ParameterCode
CleanupCacheOnLeave	ParameterCode
ClientAuthenticationData	ParameterCode
ClientAuthenticationParams	ParameterCode
ClientAuthenticationType	ParameterCode
Code	ParameterCode
CustomEventContent	ParameterCode
CustomInitData	ParameterCode

Data	ParameterCode
EmptyRoomTTL	ParameterCode
EncryptionData	ParameterCode
EncryptionMode	ParameterCode
EventForward	ParameterCode
ExpectedProtocol	ParameterCode
ExpectedValues	ParameterCode
FindFriendsRequestList	ParameterCode
FindFriendsResponseOnlineList	ParameterCode
FindFriendsResponseRoomIdList	ParameterCode
GameCount	ParameterCode
GameList	ParameterCode
GameProperties	ParameterCode
Group	ParameterCode
Info	ParameterCode
IsComingBack	ParameterCode
IsInactive	ParameterCode
JoinMode	ParameterCode
LobbyName	ParameterCode
LobbyStats	ParameterCode
LobbyType	ParameterCode
MasterClientId	ParameterCode
MasterPeerCount	ParameterCode
MatchMakingType	ParameterCode
NickName	ParameterCode
PeerCount	ParameterCode
PlayerProperties	ParameterCode
PlayerTTL	ParameterCode
PluginName	ParameterCode
Plugins	ParameterCode
PluginVersion	ParameterCode

Position	ParameterCode
Properties	ParameterCode
PublishUserId	ParameterCode
ReceiverGroup	ParameterCode
Region	ParameterCode
Remove	ParameterCode
RoomName	ParameterCode
RoomOptionFlags	ParameterCode
Secret	ParameterCode
SuppressRoomEvents	ParameterCode
TargetActorNr	ParameterCode
UriPath	ParameterCode
UserId	ParameterCode
WebRpcParameters	ParameterCode
WebRpcReturnCode	ParameterCode
WebRpcReturnMessage	ParameterCode



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	PhotonPing		

PhotonPing Member List

This is the complete list of members for **PhotonPing**, including all inherited members.

DebugString (defined in PhotonPing)	PhotonPing	
Dispose() (defined in PhotonPing)	PhotonPing	virtual
Done() (defined in PhotonPing)	PhotonPing	virtual
StartPing (string ip) (defined in PhotonPing)	PhotonPing	virtual
Successful (defined in PhotonPing)	PhotonPing	



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	PingMono		

PingMono Member List

This is the complete list of members for **PingMono**, including all inherited members.

DebugString (defined in PhotonPing)	PhotonPing	
Dispose() (defined in PingMono)	PingMono	virtual
Done() (defined in PingMono)	PingMono	virtual
StartPing (string ip)	PingMono	virtual
Successful (defined in PhotonPing)	PhotonPing	



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	Player		

Player Member List

This is the complete list of members for **Player**, including all inherited members.

ActorNumber

CustomProperties

Equals(object p)

Get(int id)

GetHashCode()

GetNext()

GetNextFor(Player currentPlayer)

GetNextFor(int currentPlayerId)

InternalCacheProperties(Hashtable properties)

IsInactive

IsLocal

IsMasterClient

NickName

SetCustomProperties(Hashtable propertiesToSet, Hashtable expected')

TagObject

ToString()

ToStringFull()

UserId



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	RaiseEventOptions		

RaiseEventOptions Member List

This is the complete list of members for **RaiseEventOptions**, including all inherited members.

CachingOption	RaiseEventOptions	
Default	RaiseEventOptions	static
Flags	RaiseEventOptions	
InterestGroup	RaiseEventOptions	
Receivers	RaiseEventOptions	
SequenceChannel	RaiseEventOptions	
TargetActors	RaiseEventOptions	



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	Region		

Region Member List

This is the complete list of members for **Region**, including all inherited members.

Cluster	Region
Code (defined in Region)	Region
HostAndPort (defined in Region)	Region
Ping (defined in Region)	Region
Region (string code, string address) (defined in Region)	Region
Region (string code, int ping) (defined in Region)	Region
ToString () (defined in Region)	Region
ToString (bool compact=false) (defined in Region)	Region
WasPinged (defined in Region)	Region



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	RegionHandler		

RegionHandler Member List

This is the complete list of members for **RegionHandler**, including all inherited members.

BestRegion

EnabledRegions

IsPinging (defined in **RegionHandler**)

PingMinimumOfRegions(Action< RegionHandler > onCompleteCallback)

SetRegions(OperationResponse opGetRegions) (defined in **RegionHandler**)

SummaryToCache



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	RegionPinger		

RegionPinger Member List

This is the complete list of members for **RegionPinger**, including all inherited members.

Attempts (defined in **RegionPinger**)

CurrentAttempt (defined in **RegionPinger**)

Done (defined in **RegionPinger**)

IgnoreInitialAttempt (defined in **RegionPinger**)

MaxMilliseconsPerPing (defined in **RegionPinger**)

PingWhenFailed (defined in **RegionPinger**)

RegionPinger(Region region, Action< Region > onDoneCallback) (defir

ResolveHost(string hostName)

Start() (defined in **RegionPinger**)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	Room		

Room Member List

This is the complete list of members for **Room**, including all inherited members.

AddPlayer(Player player)

AutoCleanUp

autoCleanUp

ClearExpectedUsers()

CustomProperties

emptyRoomTtl

EmptyRoomTtl

Equals(object other)

expectedUsers

ExpectedUsers

GetHashCode()

GetPlayer(int id)

IsOffline (defined in **Room**)

isOpen

IsOpen

isVisible

IsVisible

LoadBalancingClient

masterClientId

MasterClientId

MaxPlayers

maxPlayers

Name

name

PlayerCount

Players

PlayerTtl

playerTtl

propertiesListedInLobby

PropertiesListedInLobby

RemovedFromList

Room(string roomName, RoomOptions options, bool isOffline=false)

SetCustomProperties(Hashtable propertiesToSet, Hashtable expected)

SetMasterClient(Player masterClientPlayer)

SetPropertiesListedInLobby(string[] propertiesListedInLobby)

StorePlayer(Player player)

ToString()

ToStringFull()



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	RoomInfo		

RoomInfo Member List

This is the complete list of members for **RoomInfo**, including all inherited members.

autoCleanUp	RoomInfo	protected
CustomProperties	RoomInfo	
emptyRoomTtl	RoomInfo	protected
Equals(object other)	RoomInfo	
expectedUsers	RoomInfo	protected
GetHashCode()	RoomInfo	
IsOpen	RoomInfo	
isOpen	RoomInfo	protected
isVisible	RoomInfo	protected
IsVisible	RoomInfo	
masterClientId	RoomInfo	
MaxPlayers	RoomInfo	
maxPlayers	RoomInfo	protected
name	RoomInfo	protected
Name	RoomInfo	
PlayerCount	RoomInfo	
playerTtl	RoomInfo	protected
propertiesListedInLobby	RoomInfo	protected
RemovedFromList	RoomInfo	
ToString()	RoomInfo	



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	RoomOptions		

RoomOptions Member List

This is the complete list of members for **RoomOptions**, including all inherited members.

BroadcastPropsChangeToAll	RoomOptions
CleanupCacheOnLeave	RoomOptions
CustomRoomProperties	RoomOptions
CustomRoomPropertiesForLobby	RoomOptions
DeleteNullProperties	RoomOptions
EmptyRoomTtl	RoomOptions
IsOpen	RoomOptions
IsVisible	RoomOptions
MaxPlayers	RoomOptions
PlayerTtl	RoomOptions
Plugins	RoomOptions
PublishUserId	RoomOptions
SuppressRoomEvents	RoomOptions



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	SupportLogger		

SupportLogger Member List

This is the complete list of members for **SupportLogger**, including all inherited members.

Client

LogStats()

LogTrafficStats

OnConnected()

OnConnectedToMaster()

OnCreatedRoom()

OnCreateRoomFailed(short returnCode, string message)

OnCustomAuthenticationFailed(string debugMessage)

OnCustomAuthenticationResponse(Dictionary< string, object > data)

OnDisconnected(DisconnectCause cause)

OnFriendListUpdate(List< FriendInfo > friendList)

OnJoinedLobby()

OnJoinedRoom()

OnJoinRandomFailed(short returnCode, string message)

OnJoinRoomFailed(short returnCode, string message)

OnLeftLobby()

OnLeftRoom()

OnLobbyStatisticsUpdate(List< TypedLobbyInfo > lobbyStatistics)

OnMasterClientSwitched(Player newMasterClient)

OnPlayerEnteredRoom(Player newPlayer)

OnPlayerLeftRoom(Player otherPlayer)

OnPlayerPropertiesUpdate(Player targetPlayer, Hashtable changedProperties)

OnRegionListReceived(RegionHandler regionHandler)

OnRoomListUpdate(List< RoomInfo > roomList)

OnRoomPropertiesUpdate(Hashtable propertiesThatChanged)



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	TypedLobby		

TypedLobby Member List

This is the complete list of members for **TypedLobby**, including all inherited members.

Default (defined in TypedLobby)	T
IsDefault (defined in TypedLobby)	T
Name	T
ToString() (defined in TypedLobby)	T
Type	T
TypedLobby() (defined in TypedLobby)	T
TypedLobby (string name, LobbyType type) (defined in TypedLobby)	T



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	TypedLobbyInfo		

TypedLobbyInfo Member List

This is the complete list of members for **TypedLobbyInfo**, including all inherited members.

Default (defined in TypedLobby)	T
IsDefault (defined in TypedLobby)	T
Name	T
PlayerCount (defined in TypedLobbyInfo)	T
RoomCount (defined in TypedLobbyInfo)	T
ToString() (defined in TypedLobbyInfo)	T
Type	T
TypedLobby() (defined in TypedLobby)	T
TypedLobby (string name, LobbyType type) (defined in TypedLobby)	T



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	WebFlags		

WebFlags Member List

This is the complete list of members for **WebFlags**, including all inherited members.

Default (defined in WebFlags)	WebFlags	static
HttpForward	WebFlags	
HttpForwardConst (defined in WebFlags)	WebFlags	
SendAuthCookie	WebFlags	
SendAuthCookieConst (defined in WebFlags)	WebFlags	
SendState	WebFlags	
SendStateConst (defined in WebFlags)	WebFlags	
SendSync	WebFlags	
SendSyncConst (defined in WebFlags)	WebFlags	
WebFlags (byte webhookFlags) (defined in WebFlags)	WebFlags	
WebhookFlags (defined in WebFlags)	WebFlags	



Photon Unity Networking 2 2.12

Main Page	Related Pages	Modules	Classes	
Class List	Class Index	Class Hierarchy	Class Members	
Photon	Realtime	WebRpcResponse		

WebRpcResponse Member List

This is the complete list of members for **WebRpcResponse**, including all inherited members.

DebugMessage	WebRpcResponse
Name	WebRpcResponse
Parameters	WebRpcResponse
ReturnCode	WebRpcResponse
ToStringFull()	WebRpcResponse
WebRpcResponse(OperationResponse response)	WebRpcResponse