# Photon Unity Networking v1.18

Generated by Doxygen 1.8.2

Wed Jan 30 2013 16:19:09

## **Contents**

1	Main Page	1
2	General Documentation	3
3	Gui for Network Simulation	13
4	Gui for Network Statistics	15
5	What is the public PUN api	17
6	Module Index	19
	6.1 Modules	19
7	Namespace Index	21
	7.1 Packages	21
8	Hierarchical Index	23
	8.1 Class Hierarchy	23
9	Class Index	25
	9.1 Class List	25
10	) File Index	27
	10.1 File List	27
11	Module Documentation	29
	11.1 Optional Gui Elements	29
	11.1.1 Detailed Description	29
	11.2 Public API	30
	11.2.1 Detailed Description	31
	11.2.2 Enumeration Type Documentation	31
	11.2.2.1 DisconnectCause	31
	11.2.2.2 PeerState	31
	11.2.2.3 PhotonLogLevel	32
	11.2.2.4 PhotonNetworkingMessage	32

ii CONTENTS

			11.2.2.5	PhotonTargets	3			 	 	 	34
12	Nam	espace	Documer	tation							35
	12.1	Packag	je Photon					 	 	 	35
13	Class	s Docur	mentation								37
	13.1	ActorPi	roperties (	Class Reference	e			 	 	 	37
		13.1.1	Detailed	Description .				 	 	 	37
		13.1.2	Member	Data Documen	tation			 	 	 	37
			13.1.2.1	PlayerName				 	 	 	37
	13.2	ErrorCo	ode Class	Reference				 	 	 	37
		13.2.1	Detailed	Description .				 	 	 	38
		13.2.2	Member	Data Documen	tation			 	 	 	38
			13.2.2.1	AlreadyMatch	ed			 	 	 	38
			13.2.2.2	GameClosed				 	 	 	38
			13.2.2.3	GameDoesNo	otExist			 	 	 	38
			13.2.2.4	GameFull				 	 	 	39
			13.2.2.5	GameldAlread	dyExists .			 	 	 	39
			13.2.2.6	InternalServe	rError			 	 	 	39
			13.2.2.7	InvalidAuthen	tication			 	 	 	39
			13.2.2.8	InvalidOperati	onCode .			 	 	 	39
			13.2.2.9	InvalidRegion				 	 	 	39
			13.2.2.10	MaxCcuReac	h <mark>ed</mark>			 	 	 	39
			13.2.2.11	NoRandomMa	atchFound			 	 	 	39
			13.2.2.12	Ok				 	 	 	39
			13.2.2.13	OperationNot	AllowedInC	currentS	tate	 	 	 	40
			13.2.2.14	ServerFull .				 	 	 	40
			13.2.2.15	UserBlocked				 	 	 	40
	13.3	EventC	ode Class	Reference .				 	 	 	40
		13.3.1	Detailed	Description .				 	 	 	41
		13.3.2	Member	Data Documen	tation			 	 	 	41
			13.3.2.1	AppStats				 	 	 	41
			13.3.2.2	AzureNodeInf	0			 	 	 	41
			13.3.2.3	GameList				 	 	 	41
			13.3.2.4	GameListUpd	ate			 	 	 	41
			13.3.2.5	Join				 	 	 	41
			13.3.2.6	Leave				 	 	 	41
			13.3.2.7	Match				 	 	 	41
			13.3.2.8	PropertiesCha	anged			 	 	 	41
			13.3.2.9	QueueState				 	 	 	41
			13.3.2.10	SetProperties				 	 	 	42

CONTENTS

13.4	Extensi	ons Class	Reference	 	42
	13.4.1	Detailed I	Description	 	42
	13.4.2	Member I	Function Documentation	 	42
		13.4.2.1	AlmostEquals	 	42
		13.4.2.2	AlmostEquals	 	43
		13.4.2.3	AlmostEquals	 	43
		13.4.2.4	AlmostEquals	 	43
		13.4.2.5	Contains	 	43
		13.4.2.6	GetPhotonView	 	43
		13.4.2.7	GetPhotonViewsInChildren	 	43
		13.4.2.8	Merge	 	43
		13.4.2.9	MergeStringKeys	 	43
		13.4.2.10	O StripKeysWithNullValues	 	44
		13.4.2.11	1 StripToStringKeys	 	44
		13.4.2.12	2 ToStringFull	 	44
13.5	GameF	Properties	Class Reference	 	44
	13.5.1	Detailed I	Description	 	45
	13.5.2	Member I	Data Documentation	 	45
		13.5.2.1	CleanupCacheOnLeave	 	45
		13.5.2.2	IsOpen	 	45
		13.5.2.3	IsVisible	 	45
		13.5.2.4	MaxPlayers	 	45
		13.5.2.5	PlayerCount	 	45
		13.5.2.6	PropsListedInLobby	 	45
		13.5.2.7	Removed	 	45
13.6	Photon	.MonoBeh	naviour Class Reference	 	46
	13.6.1	Detailed I	Description	 	46
	13.6.2	Property	Documentation	 	46
		13.6.2.1	networkView	 	46
		13.6.2.2	photonView	 	46
13.7	Operati	onCode C	Class Reference	 	46
	13.7.1	Detailed I	Description	 	47
	13.7.2	Member I	Data Documentation	 	47
		13.7.2.1	Authenticate	 	47
		13.7.2.2	ChangeGroups	 	47
		13.7.2.3	CreateGame	 	47
		13.7.2.4	GetProperties	 	47
		13.7.2.5	JoinGame	 	47
		13.7.2.6	JoinLobby	 	47
		13.7.2.7	JoinRandomGame	 	47

iv CONTENTS

13	.7.2.8 Leave	. 47
13	.7.2.9 LeaveLobby	. 48
13	.7.2.10 RaiseEvent	. 48
13	.7.2.11 SetProperties	. 48
13.8 Parameter	Code Class Reference	. 48
13.8.1 De	stailed Description	. 49
13.8.2 M	ember Data Documentation	. 49
13	.8.2.1 ActorList	. 49
13	.8.2.2 ActorNr	. 50
13	.8.2.3 Add	. 50
13	.8.2.4 Address	. 50
13	.8.2.5 ApplicationId	. 50
13	.8.2.6 AppVersion	. 50
13	.8.2.7 AzureLocalNodeld	. 50
13	.8.2.8 AzureMasterNodeld	. 50
13	.8.2.9 AzureNodeInfo	. 50
13	.8.2.10 Broadcast	. 50
13	.8.2.11 Cache	. 50
13	.8.2.12 CleanupCacheOnLeave	. 50
13	.8.2.13 Code	. 51
13	.8.2.14 CustomEventContent	. 51
13	.8.2.15 Data	. 51
13	.8.2.16 GameCount	. 51
13	.8.2.17 GameList	. 51
13	.8.2.18 GameProperties	. 51
13	.8.2.19 Group	. 51
13	.8.2.20 MasterPeerCount	. 51
13	.8.2.21 MatchMakingType	. 51
13	.8.2.22 PeerCount	. 51
13	.8.2.23 PlayerProperties	. 51
13	.8.2.24 Position	. 52
13	.8.2.25 Properties	. 52
13	.8.2.26 ReceiverGroup	. 52
13	.8.2.27 Remove	. 52
13	.8.2.28 RoomName	. 52
13	.8.2.29 Secret	. 52
13	.8.2.30 TargetActorNr	. 52
13	.8.2.31 Userld	. 52
13.9 PhotonLag	SimulationGui Class Reference	. 52
13.9.1 De	stailed Description	. 53

CONTENTS

13.9.2 Member Function Documentation	53
13.9.2.1 OnGUI	53
13.9.2.2 Start	53
13.9.3 Member Data Documentation	53
13.9.3.1 Visible	53
13.9.3.2 Windowld	53
13.9.3.3 WindowRect	53
13.9.4 Property Documentation	53
13.9.4.1 Peer	53
13.10PhotonMessageInfo Class Reference	54
13.10.1 Detailed Description	54
13.10.2 Constructor & Destructor Documentation	54
13.10.2.1 PhotonMessageInfo	54
13.10.2.2 PhotonMessageInfo	54
13.10.3 Member Function Documentation	54
13.10.3.1 ToString	54
13.10.4 Member Data Documentation	54
13.10.4.1 photonView	54
13.10.4.2 sender	54
13.10.5 Property Documentation	54
13.10.5.1 timestamp	54
13.11PhotonNetwork Class Reference	55
13.11.1 Detailed Description	59
13.11.2 Member Function Documentation	59
13.11.2.1 AllocateViewID	59
13.11.2.2 CloseConnection	59
13.11.2.3 Connect	59
13.11.2.4 Connect	59
13.11.2.5 ConnectUsingSettings	60
13.11.2.6 ConnectUsingSettings	60
13.11.2.7 CreateRoom	60
13.11.2.8 CreateRoom	61
13.11.2.9 CreateRoom	61
13.11.2.10Destroy	61
13.11.2.11Destroy	61
13.11.2.12DestroyPlayerObjects	62
13.11.2.13Disconnect	62
13.11.2.14GetPing	62
13.11.2.15GetRoomList	62
13.11.2.16InitializeSecurity	62

vi CONTENTS

13.11.2.17Instantiate	 . 62
13.11.2.18Instantiate	 . 63
13.11.2.19nstantiateSceneObject	 . 63
13.11.2.20InternalCleanPhotonMonoFromScenelfStuck	 . 63
13.11.2.21JoinRandomRoom	 . 64
13.11.2.22JoinRandomRoom	 . 64
13.11.2.23JoinRandomRoom	 . 64
13.11.2.24JoinRoom	 . 64
13.11.2.25JoinRoom	 . 64
13.11.2.26LeaveRoom	 . 65
13.11.2.27LoadLevel	 . 65
13.11.2.28LoadLevel	 . 65
13.11.2.29NetworkStatisticsReset	 . 65
13.11.2.30NetworkStatisticsToString	 . 65
13.11.2.31RemoveAllBufferedMessages	 . 65
13.11.2.32RemoveAllBufferedMessages	 . 65
13.11.2.33RemoveAllInstantiatedObjects	 . 65
13.11.2.34RemoveAllInstantiatedObjects	 . 66
13.11.2.35RemoveRPCs	 . 66
13.11.2.36RemoveRPCs	 . 66
13.11.2.37RemoveRPCs	 . 66
13.11.2.38RemoveRPCsInGroup	 . 66
13.11.2.39SendOutgoingCommands	 . 66
13.11.2.40SetLevelPrefix	 . 66
13.11.2.41SetPlayerCustomProperties	 . 67
13.11.2.42SetReceivingEnabled	 . 67
13.11.2.43SetSendingEnabled	 . 67
13.11.2.44UnAllocateViewID	 . 67
13.11.3 Member Data Documentation	 . 67
13.11.3.1 logLevel	 . 67
13.11.3.2 MAX_VIEW_IDS	 . 68
13.11.3.3 precisionForFloatSynchronization	 . 68
13.11.3.4 precisionForQuaternionSynchronization	 . 68
13.11.3.5 precisionForVectorSynchronization	 . 68
13.11.3.6 PrefabCache	 . 68
13.11.3.7 serverSettingsAssetFile	 . 68
13.11.3.8 serverSettingsAssetPath	 . 68
13.11.3.9 UsePrefabCache	 . 68
13.11.3.10versionPUN	 . 68
13.11.4 Property Documentation	 . 68

CONTENTS vii

13.11.4.1 autoCleanUpPlayerObjects	69
13.11.4.2 autoJoinLobby	69
13.11.4.3 automaticallySyncScene	69
13.11.4.4 connected	69
13.11.4.5 connectionState	69
13.11.4.6 connectionStateDetailed	69
13.11.4.7 countOfPlayers	69
13.11.4.8 countOfPlayersInRooms	69
13.11.4.9 countOfPlayersOnMaster	70
13.11.4.10countOfRooms	70
13.11.4.11insideLobby	70
13.11.4.12sMasterClient	70
13.11.4.13sMessageQueueRunning	70
13.11.4.14isNonMasterClientInRoom	70
13.11.4.15masterClient	70
13.11.4.16maxConnections	70
13.11.4.17NetworkStatisticsEnabled	70
13.11.4.18offlineMode	70
13.11.4.19otherPlayers	71
13.11.4.20player	71
13.11.4.21playerList	71
13.11.4.22playerName	71
13.11.4.23room	71
13.11.4.24sendRate	71
13.11.4.25sendRateOnSerialize	71
13.11.4.26time	71
13.11.4.27unreliableCommandsLimit	71
13.12PhotonPlayer Class Reference	72
13.12.1 Detailed Description	72
13.12.2 Constructor & Destructor Documentation	73
13.12.2.1 PhotonPlayer	73
13.12.2.2 PhotonPlayer	73
13.12.3 Member Function Documentation	73
13.12.3.1 Equals	73
13.12.3.2 Find	73
13.12.3.3 GetHashCode	73
13.12.3.4 SetCustomProperties	73
13.12.3.5 ToString	74
13.12.4 Member Data Documentation	74
13.12.4.1 isLocal	74

viii CONTENTS

13.12.5 Property Documentation	. 74
13.12.5.1 allProperties	. 74
13.12.5.2 customProperties	. 74
13.12.5.3 ID	. 74
13.12.5.4 isMasterClient	. 74
13.12.5.5 name	. 74
13.13PhotonStatsGui Class Reference	. 74
13.13.1 Detailed Description	. 75
13.13.2 Member Function Documentation	. 75
13.13.2.1 OnGUI	. 75
13.13.2.2 Start	. 75
13.13.2.3 TrafficStatsWindow	. 75
13.13.2.4 Update	. 75
13.13.3 Member Data Documentation	. 75
13.13.3.1 buttonsOn	. 75
13.13.3.2 healthStatsVisible	. 76
13.13.3.3 statsOn	. 76
13.13.3.4 statsRect	. 76
13.13.3.5 statsWindowOn	. 76
13.13.3.6 trafficStatsOn	. 76
13.13.3.7 Windowld	. 76
13.14PhotonStream Class Reference	. 76
13.14.1 Detailed Description	. 77
13.14.2 Constructor & Destructor Documentation	. 77
13.14.2.1 PhotonStream	. 77
13.14.3 Member Function Documentation	. 77
13.14.3.1 ReceiveNext	. 77
13.14.3.2 SendNext	. 77
13.14.3.3 Serialize	. 77
13.14.3.4 Serialize	. 77
13.14.3.5 Serialize	. 77
13.14.3.6 Serialize	. 77
13.14.3.7 Serialize	. 77
13.14.3.8 Serialize	. 77
13.14.3.9 Serialize	. 77
13.14.3.10Serialize	. 77
13.14.3.11Serialize	. 77
13.14.3.12Serialize	. 77
13.14.3.13ToArray	
13.14.4 Property Documentation	. 77

CONTENTS

13.14.4.1 Count	. 77
13.14.4.2 isReading	. 77
13.14.4.3 isWriting	. 77
13.15PhotonView Class Reference	. 78
13.15.1 Detailed Description	. 79
13.15.2 Member Function Documentation	. 79
13.15.2.1 Awake	. 79
13.15.2.2 Find	. 79
13.15.2.3 Get	. 79
13.15.2.4 Get	
13.15.2.5 OnApplicationQuit	
13.15.2.6 OnDestroy	. 79
13.15.2.7 RPC	. 79
13.15.2.8 RPC	. 79
13.15.2.9 ToString	. 79
13.15.3 Member Data Documentation	. 79
13.15.3.1 group	. 79
13.15.3.2 instantiationId	. 79
13.15.3.3 observed	. 79
13.15.3.4 onSerializeRigidBodyOption	. 79
13.15.3.5 onSerializeTransformOption	. 79
13.15.3.6 ownerld	. 79
13.15.3.7 prefixBackup	. 79
13.15.3.8 subld	. 79
13.15.3.9 synchronization	. 79
13.15.4 Property Documentation	. 80
13.15.4.1 instantiationData	. 80
13.15.4.2 isMine	. 80
13.15.4.3 isSceneView	. 80
13.15.4.4 owner	. 80
13.15.4.5 OwnerActorNr	. 80
13.15.4.6 prefix	. 80
13.15.4.7 viewID	. 80
13.16Room Class Reference	. 80
13.16.1 Detailed Description	. 81
13.16.2 Member Function Documentation	. 81
13.16.2.1 SetCustomProperties	. 81
13.16.3 Property Documentation	
13.16.3.1 autoCleanUp	
13.16.3.2 maxPlayers	. 81

CONTENTS

13.16.3.3 name	82
13.16.3.4 open	82
13.16.3.5 playerCount	82
13.16.3.6 propertiesListedInLobby	82
13.16.3.7 visible	82
13.17RoomInfo Class Reference	82
13.17.1 Detailed Description	83
13.17.2 Member Function Documentation	83
13.17.2.1 Equals	83
13.17.2.2 GetHashCode	84
13.17.2.3 ToString	84
13.17.3 Member Data Documentation	84
13.17.3.1 autoCleanUpField	84
13.17.3.2 maxPlayersField	84
13.17.3.3 nameField	84
13.17.3.4 openField	84
13.17.3.5 visibleField	84
13.17.4 Property Documentation	84
13.17.4.1 customProperties	84
13.17.4.2 isLocalClientInside	84
13.17.4.3 maxPlayers	85
13.17.4.4 name	85
13.17.4.5 open	85
13.17.4.6 playerCount	85
13.17.4.7 removedFromList	85
13.17.4.8 visible	85
13.18ServerSettings Class Reference	85
13.18.1 Detailed Description	86
13.18.2 Member Enumeration Documentation	86
13.18.2.1 HostingOption	86
13.18.3 Member Function Documentation	87
13.18.3.1 FindRegionForServerAddress	87
13.18.3.2 FindServerAddressForRegion	87
13.18.3.3 ToString	87
13.18.3.4 UseCloud	87
13.18.3.5 UseMyServer	87
13.18.4 Member Data Documentation	87
13.18.4.1 AppID	87
13.18.4.2 CloudServerRegionNames	87
13.18.4.3 CloudServerRegionPrefixes	87

CONTENTS xi

	13.18.4.4 DefaultAppID	87
	13.18.4.5 DefaultCloudServerUrl	87
	13.18.4.6 DefaultMasterPort	87
	13.18.4.7 DefaultServerAddress	87
	13.18.4.8 DisableAutoOpenWizard	87
	13.18.4.9 HostType	87
	13.18.4.10ServerAddress	87
	13.18.4.11ServerPort	87
14	File Documentation	89
	14.1 _Doc/general.md File Reference	89
	14.2 _Doc/main.md File Reference	89
	14.3 _Doc/optionalGui.md File Reference	89
	14.4 _Doc/photonStatsGui.md File Reference	89
	14.5 _Doc/publicApi.md File Reference	89
	14.6 Photon Unity Networking/Plugins/PhotonNetwork/CustomTypes.cs File Reference	89
	14.7 Photon Unity Networking/Plugins/PhotonNetwork/Enums.cs File Reference	89
	14.7.1 Enumeration Type Documentation	90
	14.7.1.1 ConnectionState	90
	14.8 Photon Unity Networking/Plugins/PhotonNetwork/Extensions.cs File Reference	90
	14.9 Photon Unity Networking/Plugins/PhotonNetwork/LoadbalancingPeer.cs File Reference	90
	14.9.1 Enumeration Type Documentation	91
	14.9.1.1 MatchmakingMode	91
	14.10 Photon Unity Networking/Plugins/PhotonNetwork/NetworkingPeer.cs File Reference	91
	14.11 Photon Unity Networking/Plugins/PhotonNetwork/PhotonClasses.cs File Reference	91
	14.12Photon Unity Networking/Plugins/PhotonNetwork/PhotonHandler.cs File Reference	92
	14.13Photon Unity Networking/Plugins/PhotonNetwork/PhotonLagSimulationGui.cs File Reference	92
	14.14Photon Unity Networking/Plugins/PhotonNetwork/PhotonNetwork.cs File Reference	92
	14.14.1 Typedef Documentation	93
	14.14.1.1 Object	93
	14.15Photon Unity Networking/Plugins/PhotonNetwork/PhotonPlayer.cs File Reference	93
	14.16Photon Unity Networking/Plugins/PhotonNetwork/PhotonStatsGui.cs File Reference	93
	14.17 Photon Unity Networking/Plugins/PhotonNetwork/PhotonView.cs File Reference	93
	14.17.1 Enumeration Type Documentation	93
	14.17.1.1 OnSerializeRigidBody	93
	14.17.1.2 OnSerializeTransform	93
	14.17.1.3 ViewSynchronization	94
	14.18 Photon Unity Networking/Plugins/PhotonNetwork/Room.cs File Reference	94
	14.19 Photon Unity Networking/Plugins/PhotonNetwork/RoomInfo.cs File Reference	94
	14.20 Photon Unity Networking/Plugins/PhotonNetwork/ServerSettings.cs File Reference	94

••	001175170
ii	CONTENTS
11	CONTENIO

Index 94

## Main Page

#### Introduction

Photon Unity Network (PUN) is an alternative networking solution for Unity, which aims to fix and extend the built-in networking. It makes use of Photon which becomes easier to use than ever before. Full source code is available, so you can scale this plugin to support any type of multiplayer game you'd ever need.

The PhotonNetwork API looks very similar to that of Unity's networking solution and users who have experience with Unity Networking should feel at home immediately. Even better: An automatic converter will help you port your Unity networking project to the Photon equivalent. Users that have no previous networking experience should have no easier experience than starting with Photon: the powerful API abstracts all the complicated work.

By default, this plugin makes use of the hosted "Exit Games Cloud" service, which runs Photon for you. A setup window registers you (for free) in less than a minute.

#### Most notable features:

- · Dead-easy API
- · Server available as hosted service (currently free of charge!)
- Partially automatic conversion from Unity Networking to PhotonNetworking
- Offline mode: re-use your multiplayer code in singleplayer game modes
- Outstanding performance of the Photon Server
- Load balanced workflow scales across servers (with no extra effort)
- · No direct P2P and no NAT punch-through needed

#### **Next Steps**

If you know how to use Unity's networking, then you should feel at home with PUN, too. You might want to run the converter on one of your projects and dive into the code.

To read up on PUN, this documentation is split into a General Documentation and a Public API reference documentation

Aside from that, the source of Photon Unity Networking is available to you.

2 Main Page

### **General Documentation**

#### **Photon Server**

#### **Exit Games Cloud**

The Exit Games Cloud is a service that provides hosted and load balanced Photon instances for you, run by Exit Games. Free trials are available and subscription costs for commercial use are comparable with web hosting offers.

In the service, you can't implement your own server logic. Instead, make the clients authoritative. Applications are separated by "application id" and your client's a "game version". With that, your players won't clash with that of another developer or older versions.

When you imported the editor scripts from the Photon Unity Networking package, a setup window automatically open. Enter your email address and register for the cloud.

Subscriptions bought in Asset Store

Follow these steps, if you bought a package with Photon Cloud Subscription in the Asset Store:

- · Register a Photon Cloud Account: cloud.exitgames.com
- · Get your AppID from the Dashboard
- Send a Mail to: developer@exitgames.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 hosted Photon service, you can run your own server and develop on top of our "Load Balancing" game logic. This gives you full control of the server logic.

The Photon v3.0 SDK can be downloaded on:  $\verb|http://www.exitgames.com/Download/Photon||$ 

If you run your own Photon server, use the setup wizard, to switch your settings for it. Open it in the Menu: Window, Photon Unity Networking.

4 General Documentation

#### First steps

This plugin consists of quite a few files, however there's only one that truly matters: PhotonNetwork. This class contains all functions and variables that you need. If you ever have custom requirements, you can always modify the source files - this plugin is just an implementation of Photon after all. The imported package includes a setup wizard, which creates a configuration for either the cloud service or your own Photon server. Check: PhotonServer-Settings.

Using Unity Javascript? To be able to use the Photon classes you'll need to move the Plugins folder to the root of your project.

To show you how this API works, here are a few examples right away.

#### Connecting to games

PhotonNetwork always uses a master server and one or more game servers. The master server manages the list of game servers and currently running games on those servers. To pick a game (or get into a random one), players connect to the Master server. The Master forwards the clients to the game servers, where the actual gameplay is done. The servers are all run on dedicated machines - there is no such thing as player-hosted 'servers'. You don't have to bother remembering about the server organization though, as the API all hides this for you.

```
PhotonNetwork.ConnectUsingSettings("v1.0");
```

The code above is required to make use of any PhotonNetwork features. It sets your client's game version and uses the setup-wizard's config (stored in: PhotonServerSettings). The wizard can also be used when you host Photon yourself. Alternatively, use Connect() and you can ignore the PhotonServerSettings file.

#### Versioning

The loadbalancing logic for Photon uses your appID to separate your players from anyone else's. The same is done by game version, which separates players with a new client from those with older clients. As we can't guarantee that different Photon Unity Networking versions are compatible with each other, we add the PUN version to your game's version before sending it (since PUN v1.7).

#### **Creating and Joining Games**

Next, you'll want to join or create a room. The following code showcases some required functions:

```
//Join a room
PhotonNetwork.JoinRoom(roomName);

//Create this room.
PhotonNetwork.CreateRoom(roomName);

// Fails if it already exists and calls: OnPhotonCreateGameFailed

//Tries to join any random game:
PhotonNetwork.JoinRandomRoom();

//Fails if there are no matching games: OnPhotonRandomJoinFailed
```

A list of currently running games is provided by the master server's lobby. It can be joined like other rooms but only provides and updates the list of rooms. The PhotonNetwork plugin will automatically join the lobby after connecting. When you're joining a room, the list will no longer update.

To display the list of rooms (in a lobby):

Alternatively, the game can use random matchmaking: It will try to join any room and fail if none has room for another player. In that case: Create a room without name and wait until other players join it randomly.

#### Advanced Matchmaking & Room Properties

Fully random matchmaking is not always something players enjoy. Sometimes you just want to play a certain map or just two versus two.

In Photon Cloud and Loadbalancing, you can set arbitrary room properties and filter for those in JoinRandom.

#### **Room** Properties and the Lobby

Room properties are synced to all players in the room and can be useful to keep track of the current map, round, starttime, etc. They are handled as Hashtable with string keys. Preferably short keys.

You can forward selected properties to the lobby, too. This makes them available for listing them and for random matchmaking, too. Not all room properties are interesting in the lobby, so you define the set of properties for the lobby on room creation.

```
string[] roomPropsInLobby = { "map", "ai" };
Hashtable customRoomProperties = new Hashtable() { { "map", 1 } };
CreateRoom(roomName, true, true, 4, customRoomProperties, roomPropsInLobby);
```

Note that "ai" has no value yet. It won't show up in the lobby until it's set in the game via Room.SetCustom-Properties(). When you change the values for "map" or "ai", they will be updated in the lobby with a short delay, too.

Keep the list short to make sure your clients performance doesn't suffer from loading the list.

#### Filtering Room Properties in Join Random

In JoinRandom, you could pass a Hashtable with expected room properties and max player value. These work as filters when the server selects a "fitting" room for you.

```
\label{thm:hashtable} Hashtable \ expected Custom Room Properties = new \ Hashtable \mbox{() } \{ \ \mbox{"map", 1 } \}; \\ \mbox{JoinRandom Room (expected Custom Room Properties, 4);}
```

If you pass more filter properties, chances are lower that a room matches them. Better limit the options.

Make sure you never filter for properties that are not known to the lobby (see above).

#### **MonoBehaviour Callbacks**

PhotonNetwork implements several callbacks to let your game know about state changes, like "connected" or "joined a game". Each of the methods used as callback is part of the PhotonNetworkingMessage enum. Per enum item, the use is explained (check the tooltip when you type in e.g. PhotonNetworkingMessage.OnConnectedToPhoton). You can add these methods on any number of MonoBehaviours, they will be called in the respective situation. The complete list of callbacks is also in the Plugin reference.

This covers the basics of setting up game rooms. Next up is actual communication in games.

#### Sending messages in game rooms

Inside a room you are able to send network messages to other connected players. Furthermore you are able to send buffered messages that will also be sent to players that connect in the future (for spawning your player for instance).

6 General Documentation

Sending messages can be done using two methods. Either RPCs or by using the PhotonView property OnSerialize-PhotonView. There is more network interaction though. You can listen for callbacks for certain network events (e.g. OnPhotonInstantiate, OnPhotonPlayerConnected) and you can trigger some of these events (PhotonNetwork-Instantiate). Don't worry if you're confused by the last paragraph, next up we'll explain for each of these subjects.

#### **Using Groups in PUN**

Groups are not synchronized when they are changed on any PhotonView. It's up to the developer to keep photonviews in the same groups on all clients, if that's needed. Using different group numbers for the same photonview on several clients will cause some inconsistent behaviour. Some network messages are checked for their receiver group at the receiver side only, namely: RPCS that are targetted to a single player (or MasterClient) RPCS that are buffered (AllBuffered/OthersBuffered). This includes PhotonNetwork.Instantiate (as it is buffered).

Technical reason for this: the photon server only supports interestgroups for messages that are not cached and are not targetted at sepcific actor(s). This might change in the future.

#### **PhotonView**

PhotonView is a script component that is used to send messages (RPCs and OnSerializePhotonView). You need to attach the PhotonView to your games gameobjects. Note that the PhotonView is very similar to Unity's NetworkView.

At all times, you need at least one PhotonView in your game in order to send messages and optionally instantiate/allocate other PhotonViews.

To add a PhotonView to a gameobject, simply select a gameobject and use: "Components/Miscellaneous/Photon View".

#### **Observe Transform**

If you attach a Transform to a PhotonView's Observe property, you can choose to sync Position, Rotation and Scale or a combination of those across the players. This can be a great help for prototyping or smaller games. Note: A change to any observed value will send out all observed values - not just the single value that's changed. Also, updates are not smoothed or interpolated.

#### Observe MonoBehaviour

A PhotonView can be set to observe a MonoBehaviour. In this case, the script's OnPhotonSerializeView method will be called. This method is called for writing an object's state and for reading it, depending on whether the script is controlled by the local player.

The simple code below shows how to add character state synchronization with just a few lines of code more:

```
void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo
    info)
{
    if (stream.isWriting)
    {
        //We own this player: send the others our data
            stream.SendNext((int)controllerScript._characterState);
            stream.SendNext(transform.position);
            stream.SendNext(transform.rotation);
    }
    else
    {
        //Network player, receive data
            controllerScript._characterState = (CharacterState) (int)stream.
            ReceiveNext();
            correctPlayerPos = (Vector3)stream.ReceiveNext();
            correctPlayerRot = (Quaternion)stream.ReceiveNext();
    }
}
```

If you send something "ReliableDeltaCompressed", make sure to always write data to the stream in the same order. If you write no data to the PhotonStream, the update is not sent. This can be useful in pauses. Now on, to yet another way to communicate: RPCs.

#### **Remote Procedure Calls**

Remote Procedure Calls (RPCs) are exactly what the name implies: methods that can be called on remote clients in the same room. To enable remote calls for a method of a MonoBehaviour, you must apply the attribute: [RPC]. A PhotonView instance is needed on the same GameObject, to call the marked functions.

```
[RPC]
void ChatMessage(string a, string b)
{
    Debug.Log("ChatMessage " + a + " " + b);
}
```

To call the method from any script, you need access to a PhotonView object. If your script derives from Photon.-MonoBehaviour, it has a photonView field. Any regular MonoBehaviour or GameObject can use: PhotonView.-Get(this) to get access to its PhotonView component and then call RPCs on it.

So, instead of directly calling the target method, you call RPC() on a PhotonView. Provide the name of the method to call, which players should call the method and then provide a list of parameters.

Careful: The parameters list used in RPC() has to match the number of expected parameters! If the receiving client can't find a matching method, it will log an error. There is one exception to this rule: The last parameter of a RPC method can be of type PhotonMessageInfo, which will provide some context for each call.

#### Timing for RPCs and Loading Levels

RPCs are called on specific PhotonViews and always target the matching one on the remote client. If the remote client does not know the fitting PhotonView, the RPC is lost.

A typical cause for lost RPCs is when clients load and set up levels. One client is faster or in the room for a longer time and sends important RPCs for objects that are not yet loaded on the other clients. The same happens when RPCs are buffered.

The solution is to pause the message queue, while during scene setup:

```
private IEnumerator MoveToGameScene()
{
    // Temporary disable processing of futher network messages
    PhotonNetwork.isMessageQueueRunning =
        false;
    Application.LoadLevel(levelName);
}
```

Disabling the message queue will delay incoming and outgoing messages until the queue is unlocked. Obviously, it's very important to unlock the queue when you're ready to go on.

RPCs that belonged to the previously loaded scene but still arrived will now be discarded. But you should be able to define a break between both scenes by RPC.

8 General Documentation

#### Various topics

#### **Differences to Unity Networking**

#### 1. Host model

- Unity networking is server-client based (NOT P2P!). Servers are run via a Unity client (so via one of the players)
- Photon is server-client based as well, but has a dedicated server; No more dropped connections due to hosts leaving.

#### 2. Connectivity

- Unity networking works with NAT punchthrough to try to improve connectivity: since players host the
  network servers, the connection often fails due to firewalls/routers etc. Connectivity can never be guaranteed, there is a low success rate.
- Photon has a dedicated server, there is no need for NAT punchthrough or other concepts. Connectivity is a guaranteed 100%. If, in the rare case, a connection fails it must be due to a very strict client side network (a business VPN for example).

#### 3. Performance

 Photon beats Unity networking performance wise. We do not have the figures to prove this yet but the library has been optimized for years now. Furthermore, since the Unity servers are player hosted latency/ping is usually worse; you rely on the connection of the player acting as server. These connections are never any better then the connection of your dedicated Photon server.

#### 4. Price

Like the Unity Networking solution, the Photon Unity Networking plugin is free as well. You can subscribe
to use Photon Cloud hosting service for your game. Alternatively, you can rent your own servers and run
Photon on them. The free license enables up to 100 concurrent players. Other licenses cost a one-time
fee (as you do the hosting) and lift the concurrent user limits.

#### 5. Features & maintenance

• Unity does not seem to give much priority to their Networking implementation. There are rarely feature improvements and bugfixes are as seldom. The Photon solution is actively maintained and parts of it are available with source code. Furthermore, Photon already offers more features than Unity, such as the built-in load balancing and offline mode.

#### 6. Master Server

• The Master Server for Photon is a bit different from the Master Server for plain Unity Networking: In our case, it's a Photon Server that lists room-names of currently played games in so called "lobbies". Like Unity's Master, it will forward clients to the Game Server(s), where the actual gameplay is done.

#### Instantiating objects over the network

In about every game you need to instantiate one or more player objects for every player. There are various options to do so which are listed below.

#### PhotonNetwork.Instantiate

PUN can automatically take care of spawning an object by passing a starting position, rotation and a prefab name to the PhotonNetwork.Instantiate method. Requirement: The prefab should be available directly under a Resources/ folder so that the prefab can be loaded at run time. Watch out with webplayers: Everything in the resources folder will be streamed at the very first scene per default. Under the webplayer settings you can specify the first level that uses assets from the Resources folder by using the "First streamed level". If you set this to your first game scene, your preloader and mainmenu will not be slowed down if they don't use the Resources folder assets.

Gain more control: Manually instantiate

If don't want to rely on the Resources folders to instantiate objects over the network you'll have to manually Instantiate objects as shown in the example at the end of this section.

The main reason for wanting to instantiate manually is gaining control over what is downloaded when for streaming webplayers. The details about streaming and the Resources folder in Unity can be found here.

If you spawn manually, you will have to assign a PhotonViewID yourself, these viewID's are the key to routing network messages to the correct gameobject/scripts. The player who wants to own and spawn a new object should allocate a new viewID using PhotonNetwork.AllocateViewID();. This PhotonViewID should then be send to all other players using a PhotonView that has already been set up (for example an existing scene PhotonView). You will have to keep in mind that this RPC needs to be buffered so that any clients that connect later will also receive the spawn instructions. Then the RPC message that is used to spawn the object will need a reference to your desired prefab and instantiate this using Unity's GameObject.Instantiate. Finally you will need to set setup the PhotonViews attached to this prefab by assigning all PhotonViews a PhotonViewID.

If you want to use asset bundles to load your network objects from, all you have to do is add your own assetbundle loading code and replace the "playerPrefab" from the example with the prefab from your asset bundle.

#### Offline mode

Offline mode is a feature to be able to re-use your multiplayer code in singleplayer game modes as well.

Mike Hergaarden: At M2H we had to rebuild our games several times as game portals usually require you to remove multiplayer functionality completely. Furthermore, being able to use the same code for single and multiplayer saves a lot of work on itself.

The most common features that you'll want to be able to use in singleplayer are sending RPCs and using Photon-Network.Instantiate. The main goal of offline mode is to disable nullreferences and other errors when using Photon-Network functionality while not connected. You would still need to keep track of the fact that you're running a singleplayer game, to set up the game etc. However, while running the game, all code should be reusable.

You need to manually enable offline mode, as PhotonNetwork needs to be able to distinguish erroneous from intended behaviour. Enabling this feature is very easy:

```
PhotonNetwork.offlineMode = true;
```

10 General Documentation

You can now reuse certain multiplayer methods without generating any connections and errors. Furthermore there is no noticeable overhead. Below follows a list of PhotonNetwork functions and variables and their results during offline mode:

PhotonNetwork.player The player ID is always -1 PhotonNetwork.playerName Works as expected. PhotonNetwork.playerList Contains only the local player PhotonNetwork.otherPlayers Always empty PhotonNetwork.time returns Time.time; PhotonNetwork.isMasterClient Always true PhotonNetwork.AllocateViewID() Works as expected. PhotonNetwork.Instantiate Works as expected PhotonNetwork.Destroy Works as expected. PhotonNetwork.RemoveRPCs/RemoveRPCsInGroup/SetReceivingEnabled/SetSendingEnabled/SetLevelPrefix While these make no sense in Singleplayer, they will not hurt either. PhotonView.RPC Works as expected.

Note that using other methods than the ones above can yield unexpected results and some will simply do nothing. E.g. PhotonNetwork.room will, obviously, return null. If you intend on starting a game in singleplayer, but move it to multiplayer at a later stage, you might want to consider hosting a 1 player game instead; this will preserve buffered RPCs and Instantiation calls, whereas offline mode Instantiations will not automatically carry over after Connecting.

Either set PhotonNetwork.offlineMode = false; or Simply call Connect() to stop offline mode.

#### Limitations

#### Views and players

For performance reasons, the PhotonNetwork API supports up to 1000 PhotonViews per player and a maximum of 2,147,483 players (note that this is WAY higher than your hardware can support!). You can easily allow for more PhotonViews per player, at the cost of maximum players. This works as follows: PhotonViews send out a viewID for every network message. This viewID is an integer and it is composed of the player ID and the player's view ID. The maximum size of an int is 2,147,483,647, divided by our MAX\_VIEW\_IDS(1000) that allows for over 2 million players, each having 1000 view IDs. As you can see, you can easily increase the player count by reducing the MAX\_VIEW\_IDS. The other way around, you can give all players more VIEW\_IDS at the cost of less maximum players. It is important to note that most games will never need more than a few view ID's per player (one or two for the character..and that's usually it). If you need much more then you might be doing something wrong! It is extremely inefficient to assign a PhotonView and ID for every bullet that your weapon fires, instead keep track of your fire bullets via the player or weapon's PhotonView.

There is room for improving your bandwidth performance by reducing the int to a short( 32,768, 32,768). By setting MAX\_VIEW\_IDS to 32 you can then still support 1023 players Search for "//LIMITS NETWORKVIEWS&PLAYERS" for all occurrences of the int viewID. Furthermore, currently the API is not using uint/ushort but only the positive range of the numbers. This is done for simplicity and the usage of viewIDs is not a crucial performance issue for most situations.

#### **Groups and Scoping**

The PhotonNetwork plugin does not support real network groups and no scoping yet. While Unity's "scope" feature is not implemented, the network groups are currently implemented purely client side: Any RPC that should be ignored due to grouping, will be discarded after it's received. This way, groups are working but won't save bandwidth.

#### **Feedback**

We are interested in your feedback, as this solution is an ongoing project for us. Let us know if something was too hidden, missing or not working. To let us know, post in our Forum: forum.exitgames.com

#### F.A.Q.

Can I use multiple PhotonViews per GameObject? Why?

Yes this is perfectly fine. You will need multiple PhotonViews if you need to observe 2 or more targets; You can only observe one per PhotonView. For your RPC's you'll only ever need one PhotonView and this can be the same

PhotonView that is already observing something. RPC's never clash with an observed target.

Can I use it from Javascript?

To be able to use the Photon classes you'll need to move the Plugins folder to the root of your project.

#### Converting your Unity networking project to Photon

Converting your Unity networking project to Photon can be done in one day. Just to be sure, make a backup of your project, as our automated converter will change your scripts. After this is done, run the converter from the Photon editor window (Window -> Photon Unity Networking -> Converter -> Start). The automatic conversion takes between 30 seconds to 10 minutes, depending on the size of your project and your computers performance. This automatic conversion takes care of the following:

- All NetworkViews are replaced with PhotonViews and the exact same settings. This is applied for all scenes and all prefabs.
- · All scripts (JS/BOO/C#) are scanned for Network API calls, and they are replaced with PhotonNetwork calls.

There are some minor differences, therefore you will need to manually fix a few script conversion bugs. After conversion, you will most likely see some compile errors. You'll have to fix these first. Most common compile errors:

PhotonNetwork.RemoveRPCs(player); PhotonNetwork.DestroyPlayerObjects(player); These do not exist, and can be safely removed. Photon automatically cleans up players when they leave (even though you can disable this and take care of cleanup yourself if you want to) ..CloseConnection takes '2' arguments... Remove the second, boolean, argument from this call. PhotonNetwork.GetPing(player); GetPing does not take any arguments, you can only request the ping to the photon server, not ping to other players. myPlayerClass.transform.photonView.XXX error You will need to convert code like this to: myPlayerClass.transform.GetComponent<PhotonView>().XXX Inside of scripts, you can use photonView to get the attached PhotonView component. However, you cannot call this on an external transform directly. RegisterServer There's no more need to register your games to a masterserver, Photon does this automatically.

You should be able to fix all compile errors in 5-30 minutes. Most errors will originate from main menu/GUI code, related to IPs/Ports/Lobby GUI.

This is where Photon differs most from Unity's solution:

There is only one Photon server and you connect using the room names. Therefore all references to IPs/ports can be removed from your code (usually GUI code). PhotonNetwork.JoinRoom(string room) only takes a room argument, you'll need to remove your old IP/port/NAT arguments. If you have been using the "Ultimate Unity networking project" by M2H, you should remove the MultiplayerFunctions class.

Lastly, all old MasterServer calls can be removed. You never need to register servers, and fetching the room list is as easy as calling PhotonNetwork.GetRoomList(). This list is always up to date (no need to fetch/poll etc). Rewriting the room listing can be most work, if your GUI needs to be redone, it might be simpler to write the GUI from scratch.

12 **General Documentation** 

### **Gui for Network Simulation**

As tool for developers, the Photon client library can simulate network conditions for lag (message delay) and loss.

The PUN package contains a small GUI component, to set the relevant values at runtime of a game.

To use it, add the component PhotonNetSimSettingsGui 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.

•	•			
(2111 ·	tor.	NATWA	APP CIN	nulation
Gui	ıvı	IACTAAC	71 K JIII	HUIALIVI

### **Gui for Network Statistics**

The PhotonStatsGui is a simple GUI component to shows tracked network metrics easily at runtime.

#### Usage

Just add the 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.

## What is the public PUN api

The public api of PUN consists of any code that is considered useful for you as developer.

These classes are grouped into a "module" in this reference, to make it easier to learn about the important stuff of PUN.

What i	is the	public	PUN	ap
--------	--------	--------	-----	----

## **Module Index**

6.1	M	OC	uk	les

Here is a list of all modules:																			
Optional Gui Elements																			29
Public API							 												3

20 **Module Index** 

# Namespace Index

7.1	Packages	
Here a	are the packages with brief descriptions (if available):	
D.		_

22 Namespace Index

# **Chapter 8**

# **Hierarchical Index**

## 8.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

ActorProperties	7
ErrorCode	7
EventCode	)
Extensions	2
GameProperties	4
MonoBehaviour	
Photon.MonoBehaviour	3
PhotonView	3
PhotonLagSimulationGui	2
MonoBehaviour	
PhotonStatsGui	4
OperationCode	3
ParameterCode	3
PhotonMessageInfo	4
PhotonNetwork	5
PhotonPlayer	2
PhotonStream	3
RoomInfo	2
Room	)
ScriptableObject	
ServerSettings 8	=

24 **Hierarchical Index** 

# **Chapter 9**

# **Class Index**

### 9.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

ActorProperties	
Class for constants. These (byte) values define "well known" properties for an Actor / Player.	
Pun uses these constants internally.	37
ErrorCode	
Class for constants. These (int) values represent error codes, as defined and sent by the Photon	
LoadBalancing logic. Pun uses these constants internally.	37
EventCode	
Class for constants. These values are for events defined by Photon Loadbalancing. Pun uses	
these constants internally.	40
Extensions	
This static class defines some useful extension methods for several existing classes (e.g. Vec-	
tor3, float and others).	42
GameProperties	
Class for constants. These (byte) values are for "well known" room/game properties used in	
Photon Loadbalancing. Pun uses these constants internally.	44
Photon.MonoBehaviour	
This class adds the property photonView, while logging a warning when your game still uses the	
networkView	46
OperationCode	
Class for constants. Contains operation codes. Pun uses these constants internally	46
ParameterCode	
Class for constants. Codes for parameters of Operations and Events. Pun uses these constants	
internally	48
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.	52
PhotonMessageInfo	
Container class for info about a particular message, RPC or update.	54
PhotonNetwork	
The main class to use the PhotonNetwork plugin. This class is static	55
PhotonPlayer	
Summarizes a "player" within a room, identified (in that room) by actorID	72
PhotonStatsGui	
Basic GUI to show traffic and health statistics of the connection to Photon, toggled by shift+tab.	74
PhotonStream	
This "container" class is used to carry your data as written by OnPhotonSerializeView	76
PhotonView	
PUN's NetworkView replacement class for networking. Use it like a NetworkView	78

26 Class Index

Room	
This class resembles a room that PUN joins (or joined). The properties are settable as opposed to those of a RoomInfo and you can close or hide "your" room.	80
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 (open, maxPlayers, etc).	82
ServerSettings	
Collection of connection-relevant settings, used internally by PhotonNetwork.ConnectUsing-	
Settings	85

# Chapter 10

# File Index

### 10.1 File List

Here is a list of all files with brief descriptions:

Photon Unity Networking/Plugins/PhotonNetwork/CustomTypes.cs	89
Photon Unity Networking/Plugins/PhotonNetwork/Enums.cs	89
Photon Unity Networking/Plugins/PhotonNetwork/Extensions.cs	90
Photon Unity Networking/Plugins/PhotonNetwork/LoadbalancingPeer.cs	90
Photon Unity Networking/Plugins/PhotonNetwork/NetworkingPeer.cs	91
Photon Unity Networking/Plugins/PhotonNetwork/PhotonClasses.cs	91
Photon Unity Networking/Plugins/PhotonNetwork/PhotonHandler.cs	92
Photon Unity Networking/Plugins/PhotonNetwork/PhotonLagSimulationGui.cs	92
Photon Unity Networking/Plugins/PhotonNetwork/PhotonNetwork.cs	92
Photon Unity Networking/Plugins/PhotonNetwork/PhotonPlayer.cs	93
Photon Unity Networking/Plugins/PhotonNetwork/PhotonStatsGui.cs	93
Photon Unity Networking/Plugins/PhotonNetwork/PhotonView.cs	93
Photon Unity Networking/Plugins/PhotonNetwork/Room.cs	94
Photon Unity Networking/Plugins/PhotonNetwork/RoomInfo.cs	94
Photon Unity Networking/Plugins/PhotonNetwork/ServerSettings.cs	94

28 File Index

## **Chapter 11**

## **Module Documentation**

### 11.1 Optional Gui Elements

#### **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.

• class PhotonStatsGui

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

#### 11.1.1 Detailed Description

While the PUN package does not provide in-game Gui components, there are some that try to make your life as developer easier.

30 Module Documentation

#### 11.2 Public API

#### Classes

· class PhotonMessageInfo

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

class PhotonStream

This "container" class is used to carry your data as written by OnPhotonSerializeView.

class PhotonNetwork

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

class PhotonPlayer

Summarizes a "player" within a room, identified (in that room) by actorID.

class PhotonView

PUN's NetworkView replacement class for networking. Use it like a NetworkView.

class Room

This class resembles a room that PUN joins (or joined). The properties are settable as opposed to those of a Room-Info and you can close or hide "your" room.

· 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 (open, maxPlayers, etc).

#### **Enumerations**

enum PeerState {

Uninitialized, PeerCreated, Connecting, Connected,

Queued, Authenticated, JoinedLobby, DisconnectingFromMasterserver,

ConnectingToGameserver, ConnectedToGameserver, Joining, Joined,

Leaving, DisconnectingFromGameserver, ConnectingToMasterserver, ConnectedComingFromGameserver, QueuedComingFromGameserver, Disconnecting, Disconnected, ConnectedToMaster }

Detailed connection / networking peer state. PUN implements a loadbalancing and authentication workflow "behind the scenes", so some states will automatically advance to some follow up state. Those states are commented with "(will-change)".

enum PhotonNetworkingMessage {

OnConnectedToPhoton, OnLeftRoom, OnMasterClientSwitched, OnPhotonCreateRoomFailed,

OnPhotonJoinRoomFailed, OnCreatedRoom, OnJoinedLobby, OnLeftLobby,

OnDisconnectedFromPhoton, OnConnectionFail, OnFailedToConnectToPhoton, OnReceivedRoomList-Update.

OnJoinedRoom, OnPhotonPlayerConnected, OnPhotonPlayerDisconnected, OnPhotonRandomJoinFailed, OnConnectedToMaster, OnPhotonSerializeView, OnPhotonInstantiate, OnPhotonMaxCccuReached }

This enum makes up the set of MonoMessages sent by Photon Unity Networking. Implement any of these constant names as method and it will be called in the respective situation.

enum DisconnectCause {

ExceptionOnConnect = StatusCode.ExceptionOnConnect, TimeoutDisconnect = StatusCode.TimeoutDisconnect, InternalReceiveException = StatusCode.InternalReceiveException, DisconnectByServer = StatusCode.DisconnectByServer,

DisconnectByServerLogic = StatusCode.DisconnectByServerLogic, DisconnectByServerUserLimit = StatusCode.DisconnectByServerUserLimit, Exception = StatusCode.Exception, InvalidRegion = ErrorCode.Invalid-Region,

MaxCcuReached = ErrorCode.MaxCcuReached }

Summarizes the cause for a disconnect. Used in: OnConnectionFail and OnFailedToConnectToPhoton.

enum PhotonTargets {

All, Others, MasterClient, AllBuffered,

OthersBuffered }

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

11.2 Public API 31

enum PhotonLogLevel { ErrorsOnly, Informational, Full }

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

#### 11.2.1 Detailed Description

#### The public api of PUN consists of any code that is considered useful for you as developer.

For documentation, we concentrate on the public api.

Opposed to that, there are several classes that are for internal use by the PUN framework. Even some of the internally used classes are public. This is for ease of use and in parts a result of how Unity works.

#### 11.2.2 Enumeration Type Documentation

#### 11.2.2.1 enum DisconnectCause

Summarizes the cause for a disconnect. Used in: OnConnectionFail and OnFailedToConnectToPhoton.

Extracted from the status codes from ExitGames.Client.Photon.StatusCode.

#### See Also

#### PhotonNetworkingMessage

#### **Enumerator:**

**ExceptionOnConnect** Connection could not be established. Possible cause: Local server not running.

**TimeoutDisconnect** Connection timed out. Possible cause: Remote server not running or required ports blocked (due to router or firewall).

DisconnectByServer Server actively disconnected this client.

**DisconnectByServerLogic** Server actively disconnected this client. Possible cause: Server's send buffer full (too much data for client).

**DisconnectByServerUserLimit** Server actively disconnected this client. Possible cause: The server's user limit was hit and client was forced to disconnect (on connect).

**Exception** Some exception caused the connection to close.

*InvalidRegion* (32756) Authorization on the Photon Cloud failed because the app's subscription does not allow to use a particular region's server.

**MaxCcuReached** (32757) Authorization on the Photon Cloud failed because the concurrent users (CCU) limit of the app's subscription is reached.

#### 11.2.2.2 enum PeerState

Detailed connection / networking peer state. PUN implements a loadbalancing and authentication workflow "behind the scenes", so some states will automatically advance to some follow up state. Those states are commented with "(will-change)".

#### **Enumerator:**

**Uninitialized** Not running. Only set before initialization and first use.

PeerCreated Created and available to connect.

**Connecting** Working to establish the initial connection to the master server (until this process is finished, no operations can be sent).(will-change)

Connected Connection is setup, now PUN will exchange keys for encryption or authenticate.(will-change)

32 Module Documentation

Queued Not used at the moment.

**Authenticated** The application is authenticated. PUN usually joins the lobby now.(will-change) Unless Auto-JoinLobby is false.

**JoinedLobby** Client is in the lobby of the Master Server and gets room listings. Use Join, Create or Join-Random to get into a room to play.

**DisconnectingFromMasterserver** Disconnecting.(will-change)

Connecting To Gameserver Connecting to game server (to join/create a room and play).(will-change)

**ConnectedToGameserver** Similar to Connected state but on game server. Still in process to join/create room.(will-change)

**Joining** In process to join/create room (on game server).(will-change)

**Joined** Final state of a room join/create sequence. This client can now exchange events / call RPCs with other clients.

**Leaving** Leaving a room.(will-change)

**DisconnectingFromGameserver** Workflow is leaving the game server and will re-connect to the master server.(will-change)

**ConnectingToMasterserver** Workflow is connected to master server and will establish encryption and authenticate your app.(will-change)

ConnectedComingFromGameserver Same as Connected but coming from game server.(will-change)

QueuedComingFromGameserver Same Queued but coming from game server.(will-change)

**Disconnecting** PUN is disconnecting. This leads to Disconnected.(will-change)

Disconnected No connection is setup, ready to connect. Similar to PeerCreated.

ConnectedToMaster Final state for connecting to master without joining the lobby (AutoJoinLobby is false).

#### 11.2.2.3 enum PhotonLogLevel

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

#### **Enumerator:**

ErrorsOnly

Informational

Full

#### 11.2.2.4 enum PhotonNetworkingMessage

This enum makes up the set of MonoMessages sent by Photon Unity Networking. Implement any of these constant names as method and it will be called in the respective situation.

Implement: public void OnLeftRoom() { //some work }

#### **Enumerator:**

OnConnectedToPhoton Called when the server is available and before client authenticates. Wait for the call to OnJoinedLobby (or OnConnectedToMaster) before the client does anything! Example: void OnConnectedToPhoton(){ ... } This is not called for transitions from the masterserver to game servers, which is hidden for PUN users.

OnLeftRoom Called once the local user left a room. Example: void OnLeftRoom(){ ... }

OnMasterClientSwitched Called -after- switching to a new MasterClient because the previous MC left the room (not when getting into a room). The last MC will already be removed at this time. Example: void OnMasterClientSwitched(PhotonPlayer newMasterClient){ ... }

11.2 Public API 33

**OnPhotonCreateRoomFailed** Called if a CreateRoom() call failed. Most likely because the room name is already in use. Example: void OnPhotonCreateRoomFailed(){ ... }

- **OnPhotonJoinRoomFailed** Called if a JoinRoom() call failed. Most likely because the room does not exist or the room is full. Example: void OnPhotonJoinRoomFailed(){ ... }
- **OnCreatedRoom** Called when CreateRoom finishes creating the room. After this, OnJoinedRoom will be called, too (no matter if creating one or joining). Example: void OnCreatedRoom(){ ... } This implies the local client is the MasterClient.
- OnJoinedLobby Called on entering the Master Server's lobby. Client can create/join rooms but room list is not available until OnReceivedRoomListUpdate is called! Example: void OnJoinedLobby(){ ... } Note: When PhotonNetwork.autoJoinLobby is false, OnConnectedToMaster will be called instead and the room list won't be available. While in the lobby, the roomlist is automatically updated in fixed intervals (which you can't modify).
- OnLeftLobby Called after leaving the lobby. Example: void OnLeftLobby(){ ... }
- OnDisconnectedFromPhoton Called after disconnecting from the Photon server. In some cases, other events are sent before OnDisconnectedFromPhoton is called. Examples: OnConnectionFail and On-FailedToConnectToPhoton. Example: void OnDisconnectedFromPhoton(){ ... }
- OnConnectionFail Called when something causes the connection to fail (after it was established), followed by a call to OnDisconnectedFromPhoton. If the server could not be reached in the first place, OnFailed-ToConnectToPhoton is called instead. The reason for the error is provided as StatusCode. Example: void OnConnectionFail(DisconnectCause cause){ ... }
- OnFailedToConnectToPhoton Called if a connect call to the Photon server failed before the connection was established, followed by a call to OnDisconnectedFromPhoton. If the connection was established but then fails, OnConnectionFail is called. Example: void OnFailedToConnectToPhoton(DisconnectCause cause){ ... }
- **OnReceivedRoomListUpdate** Called for any update of the room listing (no matter if "new" list or "update for known" list). Only called in the Lobby state (on master server). Example: void OnReceivedRoomList-Update(){ ... }
- **OnJoinedRoom** Called when entering a room (by creating or joining it). Called on all clients (including the Master Client). Example: void OnJoinedRoom(){ ... }
- **OnPhotonPlayerConnected** Called after a remote player connected to the room. This PhotonPlayer is already added to the playerlist at this time. Example: void OnPhotonPlayerConnected(PhotonPlayer new-Player){ ... }
- **OnPhotonPlayerDisconnected** Called after a remote player disconnected from the room. This PhotonPlayer is already removed from the playerlist at this time. Example: void OnPhotonPlayerDisconnected(Photon-Player otherPlayer) { ... }
- **OnPhotonRandomJoinFailed** Called after a JoinRandom() call failed. Most likely all rooms are full or no rooms are available. Example: void OnPhotonRandomJoinFailed(){ ... }
- OnConnectedToMaster Called after the connection to the master is established and authenticated but only when PhotonNetwork.AutoJoinLobby is false. If AutoJoinLobby is false, the list of available rooms won't become available but you could join (random or by name) and create rooms anyways. Example: void OnConnectedToMaster(){ ... }
- OnPhotonSerializeView Called every network 'update' on MonoBehaviours that are being observed by a PhotonView. Example: void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info){ ... }
- **OnPhotonInstantiate** Called on all scripts on a GameObject(and it's children) that have been spawned using PhotonNetwork.Instantiate Example: void OnPhotonInstantiate(PhotonMessageInfo info){ ... }
- OnPhotonMaxCccuReached Because the concurrent user limit was (temporarily) reached, this client is rejected by the server and disconnecting. When this happens, the user might try again later. You can't create or join rooms in OnPhotonMaxCcuReached(), cause the client will be disconnecting. You can raise the CCU limits with a new license (when you host yourself) or extended subscription (when using the Photon Cloud). The Photon Cloud will mail you when the CCU limit was reached. This is also visible in the Dashboard (webpage). Example: void OnPhotonMaxCccuReached(){ ... }

34 Module Documentation

#### 11.2.2.5 enum PhotonTargets

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

#### Enumerator:

AII

Others

MasterClient

AllBuffered

OthersBuffered

# **Chapter 12**

# **Namespace Documentation**

### 12.1 Package Photon

#### **Classes**

• class MonoBehaviour

This class adds the property photonView, while logging a warning when your game still uses the networkView.

Namespace	Docume	ntation
Hairiespace	Docume	riitatioi

## **Chapter 13**

## **Class Documentation**

#### 13.1 ActorProperties Class Reference

Class for constants. These (byte) values define "well known" properties for an Actor / Player. Pun uses these constants internally.

#### **Public Attributes**

const byte PlayerName = 255
 (255) Name of a player/actor.

#### 13.1.1 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.

#### 13.1.2 Member Data Documentation

13.1.2.1 const byte ActorProperties.PlayerName = 255

(255) Name of a player/actor.

The documentation for this class was generated from the following file:

Photon Unity Networking/Plugins/PhotonNetwork/LoadbalancingPeer.cs

#### 13.2 ErrorCode Class Reference

Class for constants. These (int) values represent error codes, as defined and sent by the Photon LoadBalancing logic. Pun uses these constants internally.

#### **Public Attributes**

• 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 cant be used before getting into a room).
- 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 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: Appld is unknown to Photon (in cloud service).
- const int GameIdAlreadyExists = 0x7FFF 1
  - (32766) Gameld (name) already in use (can't create another). Change name.
- const int GameFull = 0x7FFF 2
  - (32765) Game is full. This can when players took over while you joined the game.
- const int GameClosed = 0x7FFF 3
  - (32764) Game is closed and can't be joined. Join another game.
- const int AlreadyMatched = 0x7FFF 4
- const int ServerFull = 0x7FFF 5
  - (32762) Not in use currently.
- const int UserBlocked = 0x7FFF 6
  - (32761) Not in use currently.
- const int NoRandomMatchFound = 0x7FFF 7
  - (32760) Random matchmaking only succeeds if a room exists thats neither closed nor full. Repeat in a few seconds or create a new room.
- 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 MaxCcuReached = 0x7FFF 10
  - (32757) Authorization on the Photon Cloud failed because the concurrent users (CCU) limit of the app's subscription is reached.
- 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.

#### 13.2.1 Detailed Description

Class for constants. These (int) values represent error codes, as defined and sent by the Photon LoadBalancing logic. Pun uses these constants internally.

<note>Codes from the Photon Core are negative. Default-app error codes go down from short.max.</note>

- 13.2.2 Member Data Documentation
- 13.2.2.1 const int ErrorCode.AlreadyMatched = 0x7FFF 4
- 13.2.2.2 const int ErrorCode.GameClosed = 0x7FFF 3
- (32764) Game is closed and can't be joined. Join another game.
- 13.2.2.3 const int ErrorCode.GameDoesNotExist = 0x7FFF 9
- (32758) Join can fail if the room (name) is not existing (anymore). This can happen when players leave while you join.

- 13.2.2.4 const int ErrorCode.GameFull = 0x7FFF 2
- (32765) Game is full. This can when players took over while you joined the game.
- 13.2.2.5 const int ErrorCode.GameIdAlreadyExists = 0x7FFF 1
- (32766) Gameld (name) already in use (can't create another). Change name.
- 13.2.2.6 const int ErrorCode.InternalServerError = -1
- (-1) Something went wrong in the server. Try to reproduce and contact Exit Games.
- 13.2.2.7 const int ErrorCode.InvalidAuthentication = 0x7FFF
- (32767) Authentication failed. Possible cause: Appld is unknown to Photon (in cloud service).
- 13.2.2.8 const int ErrorCode.InvalidOperationCode = -2
- (-2) The operation you called is not implemented on the server (application) you connect to. Make sure you run the fitting applications.
- 13.2.2.9 const int ErrorCode.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.-exitgames.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.

- 13.2.2.10 const int ErrorCode.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 an 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.

- 13.2.2.11 const int ErrorCode.NoRandomMatchFound = 0x7FFF 7
- (32760) Random matchmaking only succeeds if a room exists thats neither closed nor full. Repeat in a few seconds or create a new room.
- 13.2.2.12 const int ErrorCode.Ok = 0
- (0) is always "OK", anything else an error or specific situation.

13.2.2.13 const int ErrorCode.OperationNotAllowedInCurrentState = -3

(-3) Operation can't be executed yet (e.g. OpJoin can't be called before being authenticated, RaiseEvent cant 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 (with AutoJoinLobby = true) or ConnectedToMaster (AutoJoinLobby = false)

13.2.2.14 const int ErrorCode.ServerFull = 0x7FFF - 5

(32762) Not in use currently.

13.2.2.15 const int ErrorCode.UserBlocked = 0x7FFF - 6

(32761) Not in use currently.

The documentation for this class was generated from the following file:

Photon Unity Networking/Plugins/PhotonNetwork/LoadbalancingPeer.cs

#### 13.3 EventCode Class Reference

Class for constants. These values are for events defined by Photon Loadbalancing. Pun uses these constants internally.

#### **Public Attributes**

• 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 QueueState = 228

(228) Currently not used. State of queueing in case of server-full

• const byte Match = 227

(227) Currently not used. Event for matchmaking

• const byte AppStats = 226

(226) Event with stats about this application (players, rooms, etc)

• const byte AzureNodeInfo = 210

(210) Internally used in case of hosting by Azure

const byte Join = (byte)LiteEventCode.Join

(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)LiteEventCode.Leave

(254) Event Leave: The player who left the game can be identified by the actorNumber.

• const byte PropertiesChanged = (byte)LiteEventCode.PropertiesChanged

(253) When you call OpSetProperties with the broadcast option "on", this event is fired. It contains the properties being set.

const byte SetProperties = (byte)LiteEventCode.PropertiesChanged

(253) When you call OpSetProperties with the broadcast option "on", this event is fired. It contains the properties being set.

#### 13.3.1 Detailed Description

Class for constants. These values are for events defined by Photon Loadbalancing. Pun uses these constants internally.

They start at 255 and go DOWN. Your own in-game events can start at 0.

#### 13.3.2 Member Data Documentation

- 13.3.2.1 const byte EventCode.AppStats = 226
- (226) Event with stats about this application (players, rooms, etc)
- 13.3.2.2 const byte EventCode.AzureNodeInfo = 210
- (210) Internally used in case of hosting by Azure
- 13.3.2.3 const byte EventCode.GameList = 230
- (230) Initial list of RoomInfos (in lobby on Master)
- 13.3.2.4 const byte EventCode.GameListUpdate = 229
- (229) Update of RoomInfos to be merged into "initial" list (in lobby on Master)
- 13.3.2.5 const byte EventCode.Join = (byte)LiteEventCode.Join
- (255) Event Join: someone joined the game. The new actorNumber is provided as well as the properties of that actor (if set in OpJoin).
- 13.3.2.6 const byte EventCode.Leave = (byte)LiteEventCode.Leave
- (254) Event Leave: The player who left the game can be identified by the actorNumber.
- 13.3.2.7 const byte EventCode.Match = 227
- (227) Currently not used. Event for matchmaking
- 13.3.2.8 const byte EventCode.PropertiesChanged = (byte)LiteEventCode.PropertiesChanged
- (253) When you call OpSetProperties with the broadcast option "on", this event is fired. It contains the properties being set.
- 13.3.2.9 const byte EventCode.QueueState = 228
- (228) Currently not used. State of queueing in case of server-full

13.3.2.10 const byte EventCode.SetProperties = (byte)LiteEventCode.PropertiesChanged

(253) When you call OpSetProperties with the broadcast option "on", this event is fired. It contains the properties being set.

The documentation for this class was generated from the following file:

• Photon Unity Networking/Plugins/PhotonNetwork/LoadbalancingPeer.cs

#### 13.4 Extensions Class Reference

This static class defines some useful extension methods for several existing classes (e.g. Vector3, float and others).

#### **Static Public Member Functions**

- static PhotonView[] GetPhotonViewsInChildren (this UnityEngine.GameObject go)
- static PhotonView GetPhotonView (this UnityEngine.GameObject go)
- static bool AlmostEquals (this Vector3 target, Vector3 second, float sqrMagniturePrecision)

compares the square magniture of target - second to given float value

static bool AlmostEquals (this Vector2 target, Vector2 second, float sqrMagniturePrecision)

compares the square magniture of target - second to given float value

static bool AlmostEquals (this Quaternion target, Quaternion second, float maxAngle)

compares the angle between target and second to given float value

• static bool AlmostEquals (this float target, float second, float floatDiff)

compares two floats and returns true of their difference is less than floatDiff

• 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.

static void MergeStringKeys (this IDictionary target, IDictionary addHash)

Merges keys of type string to target Hashtable.

• static string ToStringFull (this IDictionary origin)

Returns a string-representation of the IDictionary's content, inlcuding type-information. Note: This might turn out a "heavy-duty" call if used frequently but it's usfuly to debug Dictionary or Hashtable content.

• static Hashtable StripToStringKeys (this IDictionary original)

This method copies all string-typed keys of the original into a new Hashtable.

static void StripKeysWithNullValues (this IDictionary original)

This removes all key-value pairs that have a null-reference as value. In Photon properties are removed by setting their value to null. Changes the original passed IDictionary!

static bool Contains (this int[] target, int nr)

Checks if a particular integer value is in an int-array.

#### 13.4.1 Detailed Description

This static class defines some useful extension methods for several existing classes (e.g. Vector3, float and others).

#### 13.4.2 Member Function Documentation

13.4.2.1 static bool Extensions.AlmostEquals ( this Vector3 target, Vector3 second, float sqrMagniturePrecision )
[static]

compares the square magniture of target - second to given float value

13.4.2.2 static bool Extensions.AlmostEquals ( this Vector2 target, Vector2 second, float sqrMagniturePrecision ) [static]

compares the square magniture of target - second to given float value

 $\textbf{13.4.2.3} \quad \textbf{static bool Extensions.} \textbf{AlmostEquals ( this Quaternion } \textit{target, Quaternion } \textit{second, float } \textit{maxAngle )} \quad \texttt{[static]}$ 

compares the angle between target and second to given float value

13.4.2.4 static bool Extensions.AlmostEquals (this float target, float second, float floatDiff) [static]

compares two floats and returns true of their difference is less than floatDiff

13.4.2.5 static bool Extensions.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.

13.4.2.6 static PhotonView Extensions.GetPhotonView ( this UnityEngine.GameObject go ) [static]

13.4.2.7 static PhotonView [] Extensions.GetPhotonViewsInChildren ( this UnityEngine.GameObject go ) [static]

13.4.2.8 static void Extensions.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.

13.4.2.9 static void Extensions.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 IDicitionary passed in plus all string-typed keys from the addHash.
addHash	A IDictionary that should be merged partly into target to update it.

13.4.2.10 static void Extensions. Strip Keys With Null Values (this IDictionary original) [static]

This removes all key-value pairs that have a null-reference as value. In 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.

13.4.2.11 static Hashtable Extensions.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 IDictonary to get string-typed keys from.

#### Returns

New Hashtable containing parts of tht original.

13.4.2.12 static string Extensions.ToStringFull (this IDictionary origin) [static]

Returns a string-representation of the IDictionary's content, inlcuding type-information. Note: This might turn out a "heavy-duty" call if used frequently but it's usfuly to debug Dictionary or Hashtable content.

#### **Parameters**

origin | Some Dictionary or Hashtable.

#### **Returns**

String of the content of the IDictionary.

The documentation for this class was generated from the following file:

• Photon Unity Networking/Plugins/PhotonNetwork/Extensions.cs

#### 13.5 GameProperties Class Reference

Class for constants. These (byte) values are for "well known" room/game properties used in Photon Loadbalancing. Pun uses these constants internally.

#### **Public Attributes**

• const byte MaxPlayers = 255

(255) Max number of players that "fit" into this room. 0 is for "unlimited".

• const byte IsVisible = 254

(254) Makes this room listed or not in the lobby on master.

• const byte IsOpen = 253

(253) Allows more players to join a room (or not).

- const byte PlayerCount = 252
  - (252) Current count od players in the room. Used only in the lobby on master.
- 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)
- 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 CleanupCacheOnLeave = 249
  - Equivalent of Operation Join parameter CleanupCacheOnLeave.

#### 13.5.1 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.

#### 13.5.2 Member Data Documentation

13.5.2.1 const byte GameProperties.CleanupCacheOnLeave = 249

Equivalent of Operation Join parameter CleanupCacheOnLeave.

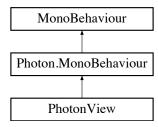
- 13.5.2.2 const byte GameProperties.IsOpen = 253
- (253) Allows more players to join a room (or not).
- 13.5.2.3 const byte GameProperties.IsVisible = 254
- (254) Makes this room listed or not in the lobby on master.
- 13.5.2.4 const byte GameProperties.MaxPlayers = 255
- (255) Max number of players that "fit" into this room. 0 is for "unlimited".
- 13.5.2.5 const byte GameProperties.PlayerCount = 252
- (252) Current count od players in the room. Used only in the lobby on master.
- 13.5.2.6 const byte GameProperties.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.
- 13.5.2.7 const byte GameProperties.Removed = 251
- (251) True if the room is to be removed from room listing (used in update to room list in lobby on master)

The documentation for this class was generated from the following file:

Photon Unity Networking/Plugins/PhotonNetwork/LoadbalancingPeer.cs

#### 13.6 Photon.MonoBehaviour Class Reference

This class adds the property photonView, while logging a warning when your game still uses the networkView. Inheritance diagram for Photon.MonoBehaviour:



#### **Properties**

- PhotonView photonView [get]
- new PhotonView networkView [get]

#### 13.6.1 Detailed Description

This class adds the property photonView, while logging a warning when your game still uses the networkView.

#### 13.6.2 Property Documentation

- 13.6.2.1 new PhotonView Photon.MonoBehaviour.networkView [get]
- **13.6.2.2 PhotonView Photon.MonoBehaviour.photonView** [get]

The documentation for this class was generated from the following file:

• Photon Unity Networking/Plugins/PhotonNetwork/PhotonClasses.cs

#### 13.7 OperationCode Class Reference

Class for constants. Contains operation codes. Pun uses these constants internally.

#### **Public Attributes**

- const byte Authenticate = 230
  - (230) Authenticates this peer and connects to a virtual application
- const byte JoinLobby = 229
  - (229) Joins lobby (on master)
- const byte LeaveLobby = 228
  - (228) Leaves lobby (on master)
- const byte CreateGame = 227
  - (227) Creates a game (or fails if name exists)
- const byte JoinGame = 226
  - (226) Join game (by name)
- const byte JoinRandomGame = 225

(225) Joins random game (on master)

const byte Leave = (byte)LiteOpCode.Leave

(254) Code for OpLeave, to get out of a room.

- const byte RaiseEvent = (byte)LiteOpCode.RaiseEvent
  - (253) Raise event (in a room, for other actors/players)
- const byte SetProperties = (byte)LiteOpCode.SetProperties

(252) Set Properties (of room or actor/player)

- const byte GetProperties = (byte)LiteOpCode.GetProperties
  - (251) Get Properties
- const byte ChangeGroups = (byte)LiteOpCode.ChangeGroups
  - (248) Operation code to change interest groups in Rooms (Lite application and extending ones).

#### 13.7.1 Detailed Description

Class for constants. Contains operation codes. Pun uses these constants internally.

- 13.7.2 Member Data Documentation
- 13.7.2.1 const byte OperationCode.Authenticate = 230
- (230) Authenticates this peer and connects to a virtual application
- 13.7.2.2 const byte OperationCode.ChangeGroups = (byte)LiteOpCode.ChangeGroups
- (248) Operation code to change interest groups in Rooms (Lite application and extending ones).
- 13.7.2.3 const byte OperationCode.CreateGame = 227
- (227) Creates a game (or fails if name exists)
- 13.7.2.4 const byte OperationCode.GetProperties = (byte)LiteOpCode.GetProperties
- (251) Get Properties
- 13.7.2.5 const byte OperationCode.JoinGame = 226
- (226) Join game (by name)
- 13.7.2.6 const byte OperationCode.JoinLobby = 229
- (229) Joins lobby (on master)
- 13.7.2.7 const byte OperationCode.JoinRandomGame = 225
- (225) Joins random game (on master)
- 13.7.2.8 const byte OperationCode.Leave = (byte)LiteOpCode.Leave
- (254) Code for OpLeave, to get out of a room.

```
13.7.2.9 const byte OperationCode.LeaveLobby = 228
```

(228) Leaves lobby (on master)

13.7.2.10 const byte OperationCode.RaiseEvent = (byte)LiteOpCode.RaiseEvent

(253) Raise event (in a room, for other actors/players)

13.7.2.11 const byte OperationCode.SetProperties = (byte)LiteOpCode.SetProperties

(252) Set Properties (of room or actor/player)

The documentation for this class was generated from the following file:

• Photon Unity Networking/Plugins/PhotonNetwork/LoadbalancingPeer.cs

#### 13.8 ParameterCode Class Reference

Class for constants. Codes for parameters of Operations and Events. Pun uses these constants internally.

#### **Public Attributes**

• const byte Address = 230

(230) Address of a (game) server to use.

• const byte PeerCount = 229

(229) Count of players in rooms (connected to game servers for this application, used in stats event)

• const byte GameCount = 228

(228) Count of games in this application (used in stats event)

• const byte MasterPeerCount = 227

(227) Count of players on the master server (connected to master server for this application, looking for games, used in stats event)

• const byte UserId = 225

(225) User's ID

const byte ApplicationId = 224

(224) Your application's ID: a name on your own Photon or a GUID on the Photon Cloud

• const byte Position = 223

(223) Not used (as "Position" currently). If you get queued before connect, this is your position

const byte MatchMakingType = 223

(223) Modifies the matchmaking algorithm used for OpJoinRandom. Allowed parameter values are defined in enum MatchmakingMode.

const byte GameList = 222

(222) List of RoomInfos about open / listed rooms

• const byte Secret = 221

(221) Internally used to establish encryption

• const byte AppVersion = 220

(220) Version of your application

• const byte AzureNodeInfo = 210

(210) Internally used in case of hosting by Azure

const byte AzureLocalNodeld = 209

(209) Internally used in case of hosting by Azure

• const byte AzureMasterNodeld = 208

(208) Internally used in case of hosting by Azure

const byte RoomName = (byte)LiteOpKey.GameId

(255) Code for the gameId/roomName (a unique name per room). Used in OpJoin and similar.

• const byte Broadcast = (byte)LiteOpKey.Broadcast

(250) Code for broadcast parameter of OpSetProperties method.

const byte ActorList = (byte)LiteOpKey.ActorList

(252) Code for list of players in a room. Currently not used.

• const byte ActorNr = (byte)LiteOpKey.ActorNr

(254) Code of the Actor of an operation. Used for property get and set.

const byte PlayerProperties = (byte)LiteOpKey.ActorProperties

(249) Code for property set (Hashtable).

const byte CustomEventContent = (byte)LiteOpKey.Data

(245) Code of data/custom content of an event. Used in OpRaiseEvent.

• const byte Data = (byte)LiteOpKey.Data

(245) Code of data of an event. Used in OpRaiseEvent.

const byte Code = (byte)LiteOpKey.Code

(244) Code used when sending some code-related parameter, like OpRaiseEvent's event-code.

• const byte GameProperties = (byte)LiteOpKey.GameProperties

(248) Code for property set (Hashtable).

• const byte Properties = (byte)LiteOpKey.Properties

(251) Code for property-set (Hashtable). This key is used when sending only one set of properties. If either Actor-Properties or GameProperties are used (or both), check those keys.

const byte TargetActorNr = (byte)LiteOpKey.TargetActorNr

(253) Code of the target Actor of an operation. Used for property set. Is 0 for game

• const byte ReceiverGroup = (byte)LiteOpKey.ReceiverGroup

(246) Code to select the receivers of events (used in Lite, Operation RaiseEvent).

• const byte Cache = (byte)LiteOpKey.Cache

(247) Code for caching events while raising them.

• 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 Group = LiteOpKey.Group

(240) Code for "group" operation-parameter (as used in Op RaiseEvent).

• const byte Remove = LiteOpKey.Remove

(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 Add = LiteOpKey.Add

(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.

#### 13.8.1 Detailed Description

Class for constants. Codes for parameters of Operations and Events. Pun uses these constants internally.

#### 13.8.2 Member Data Documentation

13.8.2.1 const byte ParameterCode.ActorList = (byte)LiteOpKey.ActorList

(252) Code for list of players in a room. Currently not used.

- 13.8.2.2 const byte ParameterCode.ActorNr = (byte)LiteOpKey.ActorNr
- (254) Code of the Actor of an operation. Used for property get and set.
- 13.8.2.3 const byte ParameterCode.Add = LiteOpKey.Add
- (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.
- 13.8.2.4 const byte ParameterCode.Address = 230
- (230) Address of a (game) server to use.
- 13.8.2.5 const byte ParameterCode.ApplicationId = 224
- (224) Your application's ID: a name on your own Photon or a GUID on the Photon Cloud
- 13.8.2.6 const byte ParameterCode.AppVersion = 220
- (220) Version of your application
- 13.8.2.7 const byte ParameterCode.AzureLocalNodeld = 209
- (209) Internally used in case of hosting by Azure
- 13.8.2.8 const byte ParameterCode.AzureMasterNodeld = 208
- (208) Internally used in case of hosting by Azure
- 13.8.2.9 const byte ParameterCode.AzureNodeInfo = 210
- (210) Internally used in case of hosting by Azure
- 13.8.2.10 const byte ParameterCode.Broadcast = (byte)LiteOpKey.Broadcast
- (250) Code for broadcast parameter of OpSetProperties method.
- 13.8.2.11 const byte ParameterCode.Cache = (byte)LiteOpKey.Cache
- (247) Code for caching events while raising them.
- 13.8.2.12 const byte ParameterCode.CleanupCacheOnLeave = (byte)241
- (241) Bool parameter of CreateGame Operation. If true, server cleans up roomcache of leaving players (their cached events get removed).

- 13.8.2.13 const byte ParameterCode.Code = (byte)LiteOpKey.Code
- (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.

- 13.8.2.14 const byte ParameterCode.CustomEventContent = (byte)LiteOpKey.Data
- (245) Code of data/custom content of an event. Used in OpRaiseEvent.
- 13.8.2.15 const byte ParameterCode.Data = (byte)LiteOpKey.Data
- (245) Code of data of an event. Used in OpRaiseEvent.
- 13.8.2.16 const byte ParameterCode.GameCount = 228
- (228) Count of games in this application (used in stats event)
- 13.8.2.17 const byte ParameterCode.GameList = 222
- (222) List of RoomInfos about open / listed rooms
- 13.8.2.18 const byte ParameterCode.GameProperties = (byte)LiteOpKey.GameProperties
- (248) Code for property set (Hashtable).
- 13.8.2.19 const byte ParameterCode.Group = LiteOpKey.Group
- (240) Code for "group" operation-parameter (as used in Op RaiseEvent).
- 13.8.2.20 const byte ParameterCode.MasterPeerCount = 227
- (227) Count of players on the master server (connected to master server for this application, looking for games, used in stats event)
- 13.8.2.21 const byte ParameterCode.MatchMakingType = 223
- (223) Modifies the matchmaking algorithm used for OpJoinRandom. Allowed parameter values are defined in enum MatchmakingMode.
- 13.8.2.22 const byte ParameterCode.PeerCount = 229
- (229) Count of players in rooms (connected to game servers for this application, used in stats event)
- 13.8.2.23 const byte ParameterCode.PlayerProperties = (byte)LiteOpKey.ActorProperties
- (249) Code for property set (Hashtable).

- 13.8.2.24 const byte ParameterCode.Position = 223
- (223) Not used (as "Position" currently). If you get queued before connect, this is your position
- 13.8.2.25 const byte ParameterCode.Properties = (byte)LiteOpKey.Properties
- (251) Code for property-set (Hashtable). This key is used when sending only one set of properties. If either Actor-Properties or GameProperties are used (or both), check those keys.
- 13.8.2.26 const byte ParameterCode.ReceiverGroup = (byte)LiteOpKey.ReceiverGroup
- (246) Code to select the receivers of events (used in Lite, Operation RaiseEvent).
- 13.8.2.27 const byte ParameterCode.Remove = LiteOpKey.Remove
- (239) The "Remove" operation-parameter can be used to remove something from a list. E.g. remove groups from player's interest groups.
- 13.8.2.28 const byte ParameterCode.RoomName = (byte)LiteOpKey.GameId
- (255) Code for the gameId/roomName (a unique name per room). Used in OpJoin and similar.
- 13.8.2.29 const byte ParameterCode.Secret = 221
- (221) Internally used to establish encryption
- 13.8.2.30 const byte ParameterCode.TargetActorNr = (byte)LiteOpKey.TargetActorNr
- (253) Code of the target Actor of an operation. Used for property set. Is 0 for game
- 13.8.2.31 const byte ParameterCode.UserId = 225
- (225) User's ID

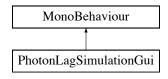
The documentation for this class was generated from the following file:

Photon Unity Networking/Plugins/PhotonNetwork/LoadbalancingPeer.cs

### 13.9 PhotonLagSimulationGui Class Reference

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.

Inheritance diagram for PhotonLagSimulationGui:



#### **Public Member Functions**

- void Start ()
- void OnGUI ()

#### **Public Attributes**

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

Positioning rect for window.

• int Windowld = 101

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

• bool Visible = true

Shows or hides GUI (does not affect settings).

#### **Properties**

• PhotonPeer Peer [get, set]

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

#### 13.9.1 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.

#### 13.9.2 Member Function Documentation

```
13.9.2.1 void PhotonLagSimulationGui.OnGUI ( )
```

13.9.2.2 void PhotonLagSimulationGui.Start ( )

#### 13.9.3 Member Data Documentation

13.9.3.1 bool PhotonLagSimulationGui.Visible = true

Shows or hides GUI (does not affect settings).

13.9.3.2 int PhotonLagSimulationGui.Windowld = 101

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

13.9.3.3 Rect PhotonLagSimulationGui.WindowRect = new Rect(0, 100, 120, 100)

Positioning rect for window.

#### 13.9.4 Property Documentation

 $\textbf{13.9.4.1} \quad \textbf{PhotonPeer PhotonLagSimulationGui.Peer} \quad \texttt{[get], [set]}$ 

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

The documentation for this class was generated from the following file:

• Photon Unity Networking/Plugins/PhotonNetwork/PhotonLagSimulationGui.cs

#### 13.10 PhotonMessageInfo Class Reference

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

#### **Public Member Functions**

- PhotonMessageInfo ()
  - Initializes a new instance of the PhotonMessageInfo class. To create an empty messageinfo only!
- PhotonMessageInfo (PhotonPlayer player, int timestamp, PhotonView view)
- override string ToString ()

#### **Public Attributes**

- · PhotonPlayer sender
- · PhotonView photonView

#### **Properties**

• double timestamp [get]

#### 13.10.1 Detailed Description

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

#### 13.10.2 Constructor & Destructor Documentation

13.10.2.1 PhotonMessageInfo.PhotonMessageInfo ( )

Initializes a new instance of the PhotonMessageInfo class. To create an empty messageinfo only!

- 13.10.2.2 PhotonMessageInfo.PhotonMessageInfo ( PhotonPlayer player, int timestamp, PhotonView view )
- 13.10.3 Member Function Documentation
- 13.10.3.1 override string PhotonMessageInfo.ToString ( )
- 13.10.4 Member Data Documentation
- 13.10.4.1 PhotonView PhotonMessageInfo.photonView
- 13.10.4.2 PhotonPlayer PhotonMessageInfo.sender
- 13.10.5 Property Documentation
- **13.10.5.1** double PhotonMessageInfo.timestamp [get]

The documentation for this class was generated from the following file:

Photon Unity Networking/Plugins/PhotonNetwork/PhotonClasses.cs

#### 13.11 PhotonNetwork Class Reference

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

#### Static Public Member Functions

static void NetworkStatisticsReset ()

Resets the traffic stats and re-enables them.

static string NetworkStatisticsToString ()

Only available when NetworkStatisticsEnabled was used to gather some stats.

static void InternalCleanPhotonMonoFromScenelfStuck ()

Internally used by Editor scripts, called on Hierarchy change (includes scene save) to remove surplus hidden Photon-Handlers.

static void ConnectUsingSettings (string gameVersion)

Connect to the configured Photon server: Reads PhotonNetwork.serverSettingsAssetPath and connects to cloud or your own server.

- static void ConnectUsingSettings ()
- static void Connect (string serverAddress, int port, string uniqueGameID)
- static void Connect (string serverAddress, int port, string appID, string gameVersion)

Connect to the photon server by address, port, appID and game(client) version. This method is used by Connect-UsingSettings which applies values from the settings file.

static void Disconnect ()

Makes this client disconnect from the photon server, a process that leaves any room and calls OnDisconnectedFrom-Photon on completition.

• static void InitializeSecurity ()

Used for compatibility with Unity networking only. Encryption is automatically initialized while connecting.

• static void CreateRoom (string roomName)

Creates a room with given name but fails if this room is existing already.

static void CreateRoom (string roomName, bool isVisible, bool isOpen, int maxPlayers)

Creates a room with given name but fails if this room is existing already.

• static void CreateRoom (string roomName, bool isVisible, bool isOpen, int maxPlayers, Hashtable custom-RoomProperties, string[] propsToListInLobby)

Creates a room with given name but fails if this room is existing already.

static void JoinRoom (RoomInfo listedRoom)

Join room by room. Name. This fails if the room is either full or no longer available (might close at the same time).

• static void JoinRoom (string roomName)

Join room with given title. This fails if the room is either full or no longer available (might close at the same time).

• static void JoinRandomRoom ()

Joins any available room but will fail if none is currently available.

static void JoinRandomRoom (Hashtable expectedCustomRoomProperties, byte expectedMaxPlayers)

Attempts to join an open room with fitting, custom properties but fails if none is currently available.

 static void JoinRandomRoom (Hashtable expectedCustomRoomProperties, byte expectedMaxPlayers, MatchmakingMode matchingType)

Attempts to join an open room with fitting, custom properties but fails if none is currently available.

• static void LeaveRoom ()

Leave the current room

• static RoomInfo[] GetRoomList ()

Gets an array of (currently) known rooms as RoomInfo. This list is automatically updated every few seconds while this client is in the lobby (on the Master Server). Not available while being in a room.

static void SetPlayerCustomProperties (Hashtable customProperties)

Sets this (local) player's properties. This caches the properties in PhotonNetwork.player.customProperties. Create-Room, JoinRoom and JoinRandomRoom will all apply your player's custom properties when you enter the room. While in a room, your properties are synced with the other players. 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.

static int AllocateViewID ()

Allocates a viewID that's valid for the current/local player.

static void UnAllocateViewID (int viewID)

Unregister a viewID (of manually instantiated and destroyed networked objects).

static GameObject Instantiate (string prefabName, Vector3 position, Quaternion rotation, int group)

Instantiate a prefab over the network. This prefab needs to be located in the root of a "Resources" folder.

static GameObject Instantiate (string prefabName, Vector3 position, Quaternion rotation, int group, object[] data)

Instantiate a prefab over the network. This prefab needs to be located in the root of a "Resources" folder.

• static GameObject InstantiateSceneObject (string prefabName, Vector3 position, Quaternion rotation, int group, object[] data)

Instantiate a scene-owned prefab over the network. The PhotonViews will be controllable by the MasterClient. This prefab needs to be located in the root of a "Resources" folder.

static int GetPing ()

The current roundtrip time to the photon server

static void SendOutgoingCommands ()

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

• static void CloseConnection (PhotonPlayer kickPlayer)

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

static void Destroy (PhotonView view)

Destroy supplied PhotonView. This will remove all Buffered RPCs and destroy the GameObject this view is attached to (plus all childs, if any) This has the same effect as calling Destroy by passing a GameObject

• static void Destroy (GameObject go)

Destroys given GameObject. This GameObject must've been instantiated using PhotonNetwork.Instantiate and must have a PhotonView at it's root. This has the same effect as calling Destroy by passing an attached PhotonView from this GameObject

static void DestroyPlayerObjects (PhotonPlayer destroyPlayer)

Destroy all GameObjects/PhotonViews of this player. can only be called on the local player. The only exception is the master client which call call this for all players.

• static void RemoveAllInstantiatedObjects ()

MasterClient method only: Destroy ALL instantiated GameObjects

static void RemoveAllInstantiatedObjects (PhotonPlayer targetPlayer)

Destroy ALL PhotonNetwork.Instantiated GameObjects by given player. Can only be called on the local player or MasterClient. The MasterClient can call this for all players.

static void RemoveRPCs ()

Remove ALL buffered RPCs of the local player

static void RemoveRPCs (PhotonPlayer targetPlayer)

Remove ALL buffered RPCs of a player

static void RemoveAllBufferedMessages ()

Remove ALL buffered messages of the local player (RPC's and Instantiation calls) Note that this only removed the buffered messages on the server, you will still need to remove the Instantiated GameObjects yourself.

static void RemoveAllBufferedMessages (PhotonPlayer targetPlayer)

Remove ALL buffered messages of a player (RPC's and Instantiation calls) Note that this only removed the buffered messages on the server, you will still need to remove the Instantiated GameObjects yourself.

static void RemoveRPCs (PhotonView view)

Remove all buffered RPCs on given PhotonView (if they are owned by this player).

static void RemoveRPCsInGroup (int group)

Remove all buffered RPCs with given group

static void SetReceivingEnabled (int group, bool enabled)

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

• static void SetSendingEnabled (int group, bool enabled)

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

static void SetLevelPrefix (short prefix)

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

static void LoadLevel (int levelNumber)

Loads the level and automatically pauses the network queue. Call this in OnJoinedRoom to make sure no cached RPCs are fired in the wrong scene.

static void LoadLevel (string levelTitle)

Loads the level and automatically pauses the network queue. Call this in OnJoinedRoom to make sure no cached RPCs are fired in the wrong scene.

#### **Public Attributes**

const string versionPUN = "1.18"

Version number of PUN. Also used in GameVersion to separate client version from each other.

- const string serverSettingsAssetFile = "PhotonServerSettings"
- const string serverSettingsAssetPath = "Assets/Photon Unity Networking/Resources/" + PhotonNetwork.serverSettingsAssetFile + ".asset"

Path to the PhotonServerSettings file.

#### **Static Public Attributes**

static readonly int MAX\_VIEW\_IDS = 1000

The maximum amount of assigned PhotonViews PER player (or scene). See the documentation on how to raise this limitation

static float precisionForVectorSynchronization = 0.000099f

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-axix, we use 0.01f\*0.01f=0.0001f. As a remedy against float inaccuracy we use 0.000099f instead of 0.0001f.

static float precisionForQuaternionSynchronization = 1.0f

The minimum angle that a rotation needs to change before we send it via a PhotonView's OnSerialize/Observing-Component

static float precisionForFloatSynchronization = 0.01f

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

• static PhotonLogLevel logLevel = PhotonLogLevel.ErrorsOnly

Network log level. Controls how verbose PUN is.

static bool UsePrefabCache = true

While enabled (true), Instantiate uses PhotonNetwork.PrefabCache to keep game objects in memory (improving instantiation of the same prefab).

• static Dictionary< string,

 $GameObject > \underline{PrefabCache} = new \ Dictionary < string, \ GameObject > ()$ 

Keeps references to GameObjects for frequent instantiation (out of memory instead of loading the Resources).

#### **Properties**

• static bool connected [get]

Are we connected to the photon server (can be IN or OUTSIDE a room)

• static ConnectionState connectionState [get]

Simplified connection state

• static PeerState connectionStateDetailed [get]

Detailed connection state (ignorant of PUN, so it can be "disconnected" while switching servers).

static Room room [get]

Get the room we're currently in. Null if we aren't in any room.

static PhotonPlayer player [get]

The local PhotonPlayer. Always available and represents this player. CustomProperties can be set before entering a room and will be synced as well.

static PhotonPlayer masterClient [get]

The PhotonPlayer of the master client. The master client is the 'virtual owner' of the room. You can use it if you need authorative decision made by one of the players.

• static string playerName [get, set]

This local player's name.

static PhotonPlayer[] playerList [get]

The full PhotonPlayer list, including the local player.

• static PhotonPlayer[] otherPlayers [get]

The other PhotonPlayers, not including our local player.

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

• static int maxConnections [get, set]

The maximum number of players for a room. Better: Set it in CreateRoom. If no room is opened, this will return 0.

static bool automaticallySyncScene [get, set]

If true, PUN will make sure that all users are in the same scene at all times. If the MasterClient switches, all clients will load the new scene. This also takes care of smooth loading of the game scene after joining a game from your main menu.

static bool autoCleanUpPlayerObjects [get, set]

This setting defines if players in a room should destroy a leaving player's instantiated GameObjects and PhotonViews.

• static bool autoJoinLobby [get, set]

Defines if the PhotonNetwork should join the "lobby" when connected to the Master server. If this is false, On-ConnectedToMaster() will be called when connection to the Master is available. OnJoinedLobby() will NOT be called if this is false.

• static bool insideLobby [get]

Returns true when we are connected to Photon and in the lobby state

• 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 'sendRateOnSerialize'.

• static int sendRateOnSerialize [get, set]

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

• static bool isMessageQueueRunning [get, set]

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

• static int unreliableCommandsLimit [get, set]

Used once per dispatch to limit unreliable commands per channel (so after a pause, many channels can still cause a lot of unreliable commands)

• static double time [get]

Photon network time, synched with the server

static bool isMasterClient [get]

Are we the master client?

• static bool isNonMasterClientInRoom [get]

True if we are in a room (client) and NOT the room's masterclient

static int countOfPlayersOnMaster [get]

The count of players currently looking for a room. This is updated on the MasterServer (only) in 5sec intervals (if any count changed).

• static int countOfPlayersInRooms [get]

The count of players currently inside a room This is updated on the MasterServer (only) in 5sec intervals (if any count changed).

• static int countOfPlayers [get]

The count of players currently using this application. This is updated on the MasterServer (only) in 5sec intervals (if any count changed).

• static int countOfRooms [get]

The count of rooms currently in use. When inside the lobby this is based on PhotonNetwork.GetRoomList().Length. When not inside the lobby, this value updated on the MasterServer (only) in 5sec intervals (if any count changed).

static bool NetworkStatisticsEnabled [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.

#### 13.11.1 Detailed Description

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

#### 13.11.2 Member Function Documentation

```
13.11.2.1 static int PhotonNetwork.AllocateViewID() [static]
```

Allocates a viewID that's valid for the current/local player.

**Returns** 

A viewID that can be used for a new PhotonView.

```
13.11.2.2 static void PhotonNetwork.CloseConnection ( PhotonPlayer kickPlayer ) [static]
```

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

#### **Parameters**

```
kickPlayer The PhotonPlayer to kick.
```

```
13.11.2.3 static void PhotonNetwork.Connect ( string serverAddress, int port, string uniqueGameID ) [static]
```

```
13.11.2.4 static void PhotonNetwork.Connect ( string serverAddress, int port, string applD, string gameVersion ) [static]
```

Connect to the photon server by address, port, appID and game(client) version. This method is used by Connect-UsingSettings which applies values from the settings file.

To connect to the Photon Cloud, a valid Appld must be in the settings file (shown in the Photon Cloud Dashboard). https://cloud.exitgames.com/dashboard

Connecting to the Photon Cloud might fail due to:

- Network issues (calls: OnFailedToConnectToPhoton())
- Invalid region (calls: OnConnectionFail() with DisconnectCause.InvalidRegion)

• Subscription CCU limit reached (calls: OnConnectionFail() with DisconnectCause.MaxCcuReached. also calls: OnPhotonMaxCccuReached())

More about the connection limitations: http://doc.exitgames.com/photon-cloud/

#### **Parameters**

serverAddress	The master server's address (either your own or Photon Cloud address).
port	The master server's port to connect to.
appID	Your application ID (Photon Cloud provides you with a GUID for your game).
gameVersion	This client's version number. Users are separated from each other by gameversion (which
	allows you to make breaking changes).

13.11.2.5 static void PhotonNetwork.ConnectUsingSettings (string gameVersion) [static]

Connect to the configured Photon server: Reads PhotonNetwork.serverSettingsAssetPath and connects to cloud or your own server.

The PUN Setup Wizard stores your appID in a settings file and applies a server address/port. This is used for Connect(string serverAddress, int port, string appID, string gameVersion).

To connect to the Photon Cloud, a valid Appld must be in the settings file (shown in the Photon Cloud Dashboard). https://cloud.exitgames.com/dashboard

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

- Network issues (calls: OnFailedToConnectToPhoton())
- Invalid region (calls: OnConnectionFail() with DisconnectCause.InvalidRegion)
- Subscription CCU limit reached (calls: OnConnectionFail() with DisconnectCause.MaxCcuReached. also calls: OnPhotonMaxCccuReached())

More about the connection limitations: http://doc.exitgames.com/photon-cloud/

#### **Parameters**

gameVersion	This client's version number. Users are separated from each other by gameversion (which
	allows you to make breaking changes).

13.11.2.6 static void PhotonNetwork.ConnectUsingSettings() [static]

13.11.2.7 static void PhotonNetwork.CreateRoom ( string roomName ) [static]

Creates a room with given name but fails if this room is existing already.

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). Call this only on the master server. Internally, the master will respond with a server-address (and roomName, if needed). Both are used internally to switch to the assigned game server and roomName.

PhotonNetwork.autoCleanUpPlayerObjects will become this room's AutoCleanUp property and that's used by all clients that join this room.

#### **Parameters**

roomName Unique name of the room to create.
---

13.11.2.8 static void PhotonNetwork.CreateRoom ( string *roomName*, bool *isVisible*, bool *isOpen*, int *maxPlayers* ) [static]

Creates a room with given name but fails if this room is existing already.

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). Call this only on the master server. Internally, the master will respond with a server-address (and roomName, if needed). Both are used internally to switch to the assigned game server and roomName

#### **Parameters**

roomName	Unique name of the room to create. Pass null or "" to make the server generate a name.
isVisible	Shows (or hides) this room from the lobby's listing of rooms.
isOpen	Allows (or disallows) others to join this room.
maxPlayers	Max number of players that can join the room.

13.11.2.9 static void PhotonNetwork.CreateRoom ( string *roomName*, bool *isVisible*, bool *isOpen*, int *maxPlayers*, Hashtable *customRoomProperties*, string[] *propsToListInLobby* ) [static]

Creates a room with given name but fails if this room is existing already.

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). Call this only on the master server. Internally, the master will respond with a server-address (and roomName, if needed). Both are used internally to switch to the assigned game server and roomName.

PhotonNetwork.autoCleanUpPlayerObjects will become this room's AutoCleanUp property and that's used by all clients that join this room.

#### **Parameters**

roomName	Unique name of the room to create. Pass null or "" to make the server generate a name.
isVisible	Shows (or hides) this room from the lobby's listing of rooms.
isOpen	Allows (or disallows) others to join this room.
maxPlayers	Max number of players that can join the room.
customRoom-	Custom properties of the new room (set on create, so they are immediately available).
Properties	
propsToListIn-	Array of custom-property-names that should be forwarded to the lobby (include only the useful
Lobby	ones).

13.11.2.10 static void PhotonNetwork.Destroy ( PhotonView view ) [static]

Destroy supplied PhotonView. This will remove all Buffered RPCs and destroy the GameObject this view is attached to (plus all childs, if any) This has the same effect as calling Destroy by passing a GameObject

#### **Parameters**

view		
------	--	--

13.11.2.11 static void PhotonNetwork.Destroy ( GameObject go ) [static]

Destroys given GameObject. This GameObject must've been instantiated using PhotonNetwork.Instantiate and must have a PhotonView at it's root. This has the same effect as calling Destroy by passing an attached Photon-View from this GameObject

#### **Parameters**

go

13.11.2.12 static void PhotonNetwork.DestroyPlayerObjects ( PhotonPlayer destroyPlayer ) [static]

Destroy all GameObjects/PhotonViews of this player. can only be called on the local player. The only exception is the master client which call call this for all players.

#### **Parameters**

player

13.11.2.13 static void PhotonNetwork.Disconnect() [static]

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

When the client is connected, the server is being informed that this client disconnects. 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 OnDisconnectedFromPhoton are immediate. Offline mode is set to false as well. Once disconnected, the client can connect again. Use ConnectUsingSettings.

13.11.2.14 static int PhotonNetwork.GetPing() [static]

The current roundtrip time to the photon server

#### Returns

Roundtrip time (to server and back).

13.11.2.15 static RoomInfo [] PhotonNetwork.GetRoomList ( ) [static]

Gets an array of (currently) known rooms as RoomInfo. This list is automatically updated every few seconds while this client is in the lobby (on the Master Server). Not available while being in a room.

Creates a new instance of the list each time called. Copied from networkingPeer.mGameList.

#### Returns

RoomInfo[] of current rooms in lobby.

13.11.2.16 static void PhotonNetwork.InitializeSecurity() [static]

Used for compatibility with Unity networking only. Encryption is automatically initialized while connecting.

13.11.2.17 static GameObject PhotonNetwork.Instantiate ( string *prefabName*, Vector3 *position*, Quaternion *rotation*, int *group* ) [static]

Instantiate a prefab over the network. This prefab needs to be located in the root of a "Resources" folder.

Instead of using prefabs in the Resources folder, you can manually Instantiate and assign PhotonViews. See doc.

**Parameters** 

prefabName	Name of the prefab to instantiate.
position	Position Vector3 to apply on instantiation.
rotation	Rotation Quaternion to apply on instantiation.
group	The group for this PhotonView.

#### Returns

The new instance of a GameObject with initialized PhotonView.

13.11.2.18 static GameObject PhotonNetwork.Instantiate ( string *prefabName*, Vector3 *position*, Quaternion *rotation*, int *group*, object[] *data* ) [static]

Instantiate a prefab over the network. This prefab needs to be located in the root of a "Resources" folder.

Instead of using prefabs in the Resources folder, you can manually Instantiate and assign PhotonViews. See doc.

#### **Parameters**

prefabName	Name of the prefab to instantiate.
position	Position Vector3 to apply on instantiation.
rotation	Rotation Quaternion to apply on instantiation.
group	The group for this PhotonView.
data	Optional instantiation data. This will be saved to it's PhotonView.instantiationData.

#### Returns

The new instance of a GameObject with initialized PhotonView.

13.11.2.19 static GameObject PhotonNetwork.InstantiateSceneObject ( string *prefabName*, Vector3 *position*, Quaternion *rotation*, int *group*, object[] *data* ) [static]

Instantiate a scene-owned prefab over the network. The PhotonViews will be controllable by the MasterClient. This prefab needs to be located in the root of a "Resources" folder.

Only the master client can Instantiate scene objects. Instead of using prefabs in the Resources folder, you can manually Instantiate and assign PhotonViews. See doc.

# **Parameters**

prefabName	Name of the prefab to instantiate.
position	Position Vector3 to apply on instantiation.
rotation	Rotation Quaternion to apply on instantiation.
group	The group for this PhotonView.
data	Optional instantiation data. This will be saved to it's PhotonView.instantiationData.

# Returns

The new instance of a GameObject with initialized PhotonView.

13.11.2.20 static void PhotonNetwork.InternalCleanPhotonMonoFromScenelfStuck( ) [static]

Internally used by Editor scripts, called on Hierarchy change (includes scene save) to remove surplus hidden PhotonHandlers.

13.11.2.21 static void PhotonNetwork.JoinRandomRoom() [static]

Joins any available room but will fail if none is currently available.

If this fails, you can still create a room (and make this available for the next who uses JoinRandomRoom). Alternatively, try again in a moment.

13.11.2.22 static void PhotonNetwork.JoinRandomRoom ( Hashtable expectedCustomRoomProperties, byte expectedMaxPlayers ) [static]

Attempts to join an open room with fitting, custom properties but fails if none is currently available.

If this fails, you can still create a room (and make this available for the next who uses JoinRandomRoom). Alternatively, try again in a moment.

#### **Parameters**

expected-	Filters for rooms that match these custom properties (string keys and values). To ignore, pass
CustomRoom-	null.
Properties	
expectedMax-	Filters for a particular maxplayer setting. Use 0 to accept any maxPlayer value.
Players	

13.11.2.23 static void PhotonNetwork.JoinRandomRoom ( Hashtable expectedCustomRoomProperties, byte expectedMaxPlayers, MatchmakingMode matchingType ) [static]

Attempts to join an open room with fitting, custom properties but fails if none is currently available.

If this fails, you can still create a room (and make this available for the next who uses JoinRandomRoom). Alternatively, try again in a moment.

#### **Parameters**

expected-	Filters for rooms that match these custom properties (string keys and values). To ignore, pass
CustomRoom-	null.
Properties	
expectedMax-	Filters for a particular maxplayer setting. Use 0 to accept any maxPlayer value.
Players	
matchingType	Selects one of the available matchmaking algorithms. See MatchmakingMode enum for op-
	tions.

13.11.2.24 static void PhotonNetwork.JoinRoom ( RoomInfo listedRoom ) [static]

Join room by room. Name. This fails if the room is either full or no longer available (might close at the same time).

### **Parameters**

roomName The room instance to join (only listedRoom.Name is used).
--

13.11.2.25 static void PhotonNetwork.JoinRoom ( string roomName ) [static]

Join room with given title. This fails if the room is either full or no longer available (might close at the same time).

#### **Parameters**

roomName Unique name of the room to create.

13.11.2.26 static void PhotonNetwork.LeaveRoom() [static]

Leave the current room

13.11.2.27 static void PhotonNetwork.LoadLevel (int levelNumber) [static]

Loads the level and automatically pauses the network queue. Call this in OnJoinedRoom to make sure no cached RPCs are fired in the wrong scene.

#### **Parameters**

levelNumber | Number of the level to load (make sure it's in the build preferences).

13.11.2.28 static void PhotonNetwork.LoadLevel ( string levelTitle ) [static]

Loads the level and automatically pauses the network queue. Call this in OnJoinedRoom to make sure no cached RPCs are fired in the wrong scene.

#### **Parameters**

levelTitle Name of the level to load.

13.11.2.29 static void PhotonNetwork.NetworkStatisticsReset() [static]

Resets the traffic stats and re-enables them.

13.11.2.30 static string PhotonNetwork.NetworkStatisticsToString ( ) [static]

Only available when NetworkStatisticsEnabled was used to gather some stats.

#### Returns

A string with vital networking statistics.

13.11.2.31 static void PhotonNetwork.RemoveAllBufferedMessages() [static]

Remove ALL buffered messages of the local player (RPC's and Instantiation calls) Note that this only removed the buffered messages on the server, you will still need to remove the Instantiated GameObjects yourself.

13.11.2.32 static void PhotonNetwork.RemoveAllBufferedMessages ( PhotonPlayer targetPlayer ) [static]

Remove ALL buffered messages of a player (RPC's and Instantiation calls) Note that this only removed the buffered messages on the server, you will still need to remove the Instantiated GameObjects yourself.

13.11.2.33 static void PhotonNetwork.RemoveAllInstantiatedObjects() [static]

MasterClient method only: Destroy ALL instantiated GameObjects

13.11.2.34 static void PhotonNetwork.RemoveAllInstantiatedObjects ( PhotonPlayer targetPlayer ) [static]

Destroy ALL PhotonNetwork.Instantiated GameObjects by given player. Can only be called on the local player or MasterClient. The MasterClient can call this for all players.

#### **Parameters**

player

13.11.2.35 static void PhotonNetwork.RemoveRPCs() [static]

Remove ALL buffered RPCs of the local player

13.11.2.36 static void PhotonNetwork.RemoveRPCs ( PhotonPlayer targetPlayer ) [static]

Remove ALL buffered RPCs of a player

13.11.2.37 static void PhotonNetwork.RemoveRPCs ( PhotonView view ) [static]

Remove all buffered RPCs on given PhotonView (if they are owned by this player).

#### **Parameters**

view

13.11.2.38 static void PhotonNetwork.RemoveRPCsInGroup (int group) [static]

Remove all buffered RPCs with given group

#### **Parameters**

group

13.11.2.39 static void PhotonNetwork.SendOutgoingCommands() [static]

Can be used to immediately send the RPCs and Instantiates just made, 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.

13.11.2.40 static void PhotonNetwork.SetLevelPrefix ( short 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 = 32767

13.11.2.41 static void PhotonNetwork.SetPlayerCustomProperties ( Hashtable customProperties ) [static]

Sets this (local) player's properties. This caches the properties in PhotonNetwork.player.customProperties. Create-Room, JoinRoom and JoinRandomRoom will all apply your player's custom properties when you enter the room. While in a room, your properties are synced with the other players. 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**

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

13.11.2.42 static void PhotonNetwork.SetReceivingEnabled (int group, bool enabled) [static]

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

#### **Parameters**

group	
enabled	

13.11.2.43 static void PhotonNetwork.SetSendingEnabled (int group, bool enabled) [static]

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

#### **Parameters**

group	
enabled	

13.11.2.44 static void PhotonNetwork.UnAllocateViewID (int viewID) [static]

Unregister a viewID (of manually instantiated and destroyed networked objects).

# **Parameters**

viewID   A viewID manually allocated by this player		
Wewlb   A viewib mandally allocated by this player.	viewID	ID   A viewID manually allocated by this player.

# 13.11.3 Member Data Documentation

13.11.3.1 PhotonLogLevel PhotonNetwork.logLevel = PhotonLogLevel.ErrorsOnly [static]

Network log level. Controls how verbose PUN is.

```
13.11.3.2 readonly int PhotonNetwork.MAX_VIEW_IDS = 1000 [static]
```

The maximum amount of assigned PhotonViews PER player (or scene). See the documentation on how to raise this limitation

```
13.11.3.3 float PhotonNetwork.precisionForFloatSynchronization = 0.01f [static]
```

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

```
13.11.3.4 float PhotonNetwork.precisionForQuaternionSynchronization = 1.0f [static]
```

The minimum angle that a rotation needs to change before we send it via a PhotonView's OnSerialize/Observing-Component

```
13.11.3.5 float PhotonNetwork.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-axix, we use 0.01f\*0.01f=0.0001f. As a remedy against float inaccuracy we use 0.000099f instead of 0.0001f.

```
13.11.3.6 Dictionary < string, GameObject > PhotonNetwork.PrefabCache = new Dictionary < string, GameObject > () [static]
```

Keeps references to GameObjects for frequent instantiation (out of memory instead of loading the Resources).

You should be able to modify the cache anytime you like, except while Instantiate is used. Best do it only in the main-Thread.

```
13.11.3.7 const string PhotonNetwork.serverSettingsAssetFile = "PhotonServerSettings"
```

13.11.3.8 const string PhotonNetwork.serverSettingsAssetPath = "Assets/Photon Unity Networking/Resources/" + PhotonNetwork.serverSettingsAssetFile + ".asset"

Path to the PhotonServerSettings file.

```
13.11.3.9 bool PhotonNetwork.UsePrefabCache = true [static]
```

While enabled (true), Instantiate uses PhotonNetwork.PrefabCache to keep game objects in memory (improving instantiation of the same prefab).

Setting UsePrefabCache to false during runtime will not clear PrefabCache but will ignore it right away. You could clean and modify the cache yourself. Read its comments.

```
13.11.3.10 const string PhotonNetwork.versionPUN = "1.18"
```

Version number of PUN. Also used in GameVersion to separate client version from each other.

# 13.11.4 Property Documentation

```
13.11.4.1 bool PhotonNetwork.autoCleanUpPlayerObjects [static], [get], [set]
```

This setting defines if players in a room should destroy a leaving player's instantiated GameObjects and Photon-Views.

When "this client" creates a room/game, autoCleanUpPlayerObjects is copied to that room's properties and used by all PUN clients in that room (no matter what their autoCleanUpPlayerObjects value is).

If room. AutoCleanUp is enabled in a room, the PUN clients will destroy a player's objects on leave.

When enabled, the server will clean RPCs, instantiated GameObjects and PhotonViews of the leaving player and joining players won't get those at anymore.

Once a room is created, this setting can't be changed anymore.

Enabled by default.

```
13.11.4.2 bool PhotonNetwork.autoJoinLobby [static], [get], [set]
```

Defines if the PhotonNetwork should join the "lobby" when connected to the Master server. If this is false, On-ConnectedToMaster() will be called when connection to the Master is available. OnJoinedLobby() will NOT be called if this is false.

Enabled by default.

The room listing will not become available. Rooms can be created and joined (randomly) without joining the lobby (and getting sent the room list).

```
13.11.4.3 bool PhotonNetwork.automaticallySyncScene [static], [get], [set]
```

If true, PUN will make sure that all users are in the same scene at all times. If the MasterClient switches, all clients will load the new scene. This also takes care of smooth loading of the game scene after joining a game from your main menu.

true if automatically sync scene; otherwise, false.

```
13.11.4.4 bool PhotonNetwork.connected [static], [get]
```

Are we connected to the photon server (can be IN or OUTSIDE a room)

```
13.11.4.5 ConnectionState PhotonNetwork.connectionState [static], [get]
```

Simplified connection state

```
13.11.4.6 PeerState PhotonNetwork.connectionStateDetailed [static], [get]
```

Detailed connection state (ignorant of PUN, so it can be "disconnected" while switching servers).

```
13.11.4.7 int PhotonNetwork.countOfPlayers [static], [get]
```

The count of players currently using this application. This is updated on the MasterServer (only) in 5sec intervals (if any count changed).

```
13.11.4.8 int PhotonNetwork.countOfPlayersInRooms [static], [get]
```

The count of players currently inside a room This is updated on the MasterServer (only) in 5sec intervals (if any count changed).

```
13.11.4.9 int PhotonNetwork.countOfPlayersOnMaster [static], [get]
```

The count of players currently looking for a room. This is updated on the MasterServer (only) in 5sec intervals (if any count changed).

```
13.11.4.10 int PhotonNetwork.countOfRooms [static], [get]
```

The count of rooms currently in use. When inside the lobby this is based on PhotonNetwork.GetRoomList().Length. When not inside the lobby, this value updated on the MasterServer (only) in 5sec intervals (if any count changed).

```
13.11.4.11 bool PhotonNetwork.insideLobby [static], [get]
```

Returns true when we are connected to Photon and in the lobby state

```
13.11.4.12 bool PhotonNetwork.isMasterClient [static], [get]
```

Are we the master client?

```
13.11.4.13 bool PhotonNetwork.isMessageQueueRunning [static], [get], [set]
```

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

```
13.11.4.14 bool PhotonNetwork.isNonMasterClientInRoom [static], [get]
```

True if we are in a room (client) and NOT the room's masterclient

```
13.11.4.15 PhotonPlayer PhotonNetwork.masterClient [static], [get]
```

The PhotonPlayer of the master client. The master client is the 'virtual owner' of the room. You can use it if you need authorative decision made by one of the players.

The masterClient is null until a room is joined and becomes null again when the room is left.

```
13.11.4.16 int PhotonNetwork.maxConnections [static], [get], [set]
```

The maximum number of players for a room. Better: Set it in CreateRoom. If no room is opened, this will return 0.

```
13.11.4.17 bool PhotonNetwork.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

```
13.11.4.18 bool PhotonNetwork.offlineMode [static], [get], [set]
```

Offline mode can be set to re-use your multiplayer code in singleplayer game modes. When this is on Photon-Network will not create any connections and there is near to no overhead. Mostly usefull for reusing RPC's and PhotonNetwork.Instantiate

```
13.11.4.19 PhotonPlayer[] PhotonNetwork.otherPlayers [static], [get]
```

The other PhotonPlayers, not including our local player.

```
13.11.4.20 PhotonPlayer PhotonNetwork.player [static], [get]
```

The local PhotonPlayer. Always available and represents this player. CustomProperties can be set before entering a room and will be synced as well.

```
13.11.4.21 PhotonPlayer[] PhotonNetwork.playerList [static], [get]
```

The full PhotonPlayer list, including the local player.

```
13.11.4.22 string PhotonNetwork.playerName [static], [get], [set]
```

This local player's name.

Setting the name will automatically send it, if connected. Setting null, won't change the name.

```
13.11.4.23 Room PhotonNetwork.room [static], [get]
```

Get the room we're currently in. Null if we aren't in any room.

```
13.11.4.24 int PhotonNetwork.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 'sendRateOnSerialize'.

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.

```
13.11.4.25 int PhotonNetwork.sendRateOnSerialize [static], [get], [set]
```

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

Choose this value in relation to 'sendRate'. OnPhotonSerialize will creart the commands to be put into packages. A lower rate takes up less performance but will cause more lag.

```
13.11.4.26 double PhotonNetwork.time [static], [get]
```

Photon network time, synched with the server

v1.3: This time reflects milliseconds since start of the server, cut down to 4 bytes. It will overflow every 49 days from a high value to 0. We do not (yet) compensate this overflow. Master- and Game-Server will have different time values. v1.10: Fixed issues with precision for high server-time values. This should update with 15ms precision by default.

```
13.11.4.27 int PhotonNetwork.unreliableCommandsLimit [static], [get], [set]
```

Used once per dispatch to limit unreliable commands per channel (so after a pause, many channels can still cause a lot of unreliable commands)

The documentation for this class was generated from the following file:

• Photon Unity Networking/Plugins/PhotonNetwork/PhotonNetwork.cs

# 13.12 PhotonPlayer Class Reference

Summarizes a "player" within a room, identified (in that room) by actorID.

#### **Public Member Functions**

PhotonPlayer (bool isLocal, int actorID, string name)

Creates a PhotonPlayer instance.

• override string ToString ()

Gives the name.

· override bool Equals (object p)

Makes PhotonPlayer comparable

- override int GetHashCode ()
- void SetCustomProperties (Hashtable propertiesToSet)

Updates and synchronizes the named properties of this Player with the values of propertiesToSet.

#### Static Public Member Functions

static PhotonPlayer Find (int ID)

Try to get a specific player by id.

# **Public Attributes**

• readonly bool isLocal = false

Only one player is controlled by each client. Others are not local.

# **Protected Member Functions**

PhotonPlayer (bool isLocal, int actorID, Hashtable properties)

Internally used to create players from event Join

# **Properties**

```
• int ID [get]
```

This player's actorID

• string name [get, set]

Nickname of this player.

• bool isMasterClient [get]

The player with the lowest actorID is the master and could be used for special tasks.

• Hashtable customProperties [get, set]

Read-only cache for custom properties of player. Set via Player.SetCustomProperties.

• Hashtable allProperties [get]

Creates a Hashtable with all properties (custom and "well known" ones).

# 13.12.1 Detailed Description

Summarizes a "player" within a room, identified (in that room) by actorID.

Each player has an actorld (or ID), valid for that room. It's -1 until it's assigned by server. Each client can set it's player's custom properties with SetCustomProperties, even before being in a room. They are synced when joining a room.

#### 13.12.2 Constructor & Destructor Documentation

13.12.2.1 PhotonPlayer.PhotonPlayer (bool isLocal, int actorID, string name)

Creates a PhotonPlayer instance.

#### **Parameters**

isLocal	If this is the local peer's player (or a remote one).
actorID	ID or ActorNumber of this player in the current room (a shortcut to identify each player in room)
name	Name of the player (a "well known property").

13.12.2.2 PhotonPlayer.PhotonPlayer (bool isLocal, int actorID, Hashtable properties ) [protected]

Internally used to create players from event Join

#### 13.12.3 Member Function Documentation

13.12.3.1 override bool PhotonPlayer.Equals (object p)

Makes PhotonPlayer comparable

13.12.3.2 static PhotonPlayer PhotonPlayer.Find (int ID) [static]

Try to get a specific player by id.

#### **Parameters**

ID ActorID	

#### Returns

The player with matching actorID or null, if the actorID is not in use.

13.12.3.3 override int PhotonPlayer.GetHashCode ( )

13.12.3.4 void PhotonPlayer.SetCustomProperties ( Hashtable propertiesToSet )

Updates and synchronizes the named properties of this Player with the values of propertiesToSet.

Any player's properties are available in a Room only and only until the player disconnect or leaves. Access any player's properties by: Player.CustomProperties (read-only!) but don't modify that hashtable.

New properties are added, existing values are updated. Other values will not be changed, so only provide values that changed or are new. To delete a named (custom) property of this player, use null as value. Only string-typed keys are applied (everything else is ignored).

Local cache is updated immediately, other players are updated through Photon with a fitting operation. To reduce network traffic, set only values that actually changed.

# **Parameters**

propertiesToSet   Hashtable of props to udpate, set and sync. See description.
--

13.12.3.5 override string PhotonPlayer.ToString ( )

Gives the name.

#### 13.12.4 Member Data Documentation

13.12.4.1 readonly bool PhotonPlayer.isLocal = false

Only one player is controlled by each client. Others are not local.

# 13.12.5 Property Documentation

```
13.12.5.1 Hashtable PhotonPlayer.allProperties [get]
```

Creates a Hashtable with all properties (custom and "well known" ones).

If used more often, this should be cached.

```
13.12.5.2 Hashtable PhotonPlayer.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.

```
13.12.5.3 int PhotonPlayer.ID [get]
```

This player's actorID

```
13.12.5.4 bool PhotonPlayer.isMasterClient [get]
```

The player with the lowest actorID is the master and could be used for special tasks.

```
13.12.5.5 string PhotonPlayer.name [get], [set]
```

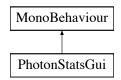
Nickname of this player.

The documentation for this class was generated from the following file:

• Photon Unity Networking/Plugins/PhotonNetwork/PhotonPlayer.cs

# 13.13 PhotonStatsGui Class Reference

Basic GUI to show traffic and health statistics of the connection to Photon, toggled by shift+tab. Inheritance diagram for PhotonStatsGui:



#### **Public Member Functions**

- void Start ()
- · void Update ()

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

- void OnGUI ()
- void TrafficStatsWindow (int windowID)

# **Public Attributes**

bool statsWindowOn = true

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

bool statsOn = true

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

· bool healthStatsVisible

Shows additional "health" values of connection.

· bool trafficStatsOn

Shows additional "lower level" traffic stats.

bool buttonsOn

Show buttons to control stats and reset them.

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

Positioning rect for window.

• int Windowld = 100

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

#### 13.13.1 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).

# 13.13.2 Member Function Documentation

```
13.13.2.1 void PhotonStatsGui.OnGUI ( )
```

13.13.2.2 void PhotonStatsGui.Start ( )

13.13.2.3 void PhotonStatsGui.TrafficStatsWindow (int windowID)

13.13.2.4 void PhotonStatsGui.Update ( )

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

# 13.13.3 Member Data Documentation

13.13.3.1 bool PhotonStatsGui.buttonsOn

Show buttons to control stats and reset them.

13.13.3.2 bool PhotonStatsGui.healthStatsVisible

Shows additional "health" values of connection.

13.13.3.3 bool PhotonStatsGui.statsOn = true

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

13.13.3.4 Rect PhotonStatsGui.statsRect = new Rect(0, 100, 200, 50)

Positioning rect for window.

13.13.3.5 bool PhotonStatsGui.statsWindowOn = true

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

13.13.3.6 bool PhotonStatsGui.trafficStatsOn

Shows additional "lower level" traffic stats.

13.13.3.7 int PhotonStatsGui.Windowld = 100

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

The documentation for this class was generated from the following file:

• Photon Unity Networking/Plugins/PhotonNetwork/PhotonStatsGui.cs

# 13.14 PhotonStream Class Reference

This "container" class is used to carry your data as written by OnPhotonSerializeView.

# **Public Member Functions**

- PhotonStream (bool write, object[] incomingData)
- object ReceiveNext ()
- void SendNext (object obj)
- object[] ToArray ()
- void Serialize (ref bool myBool)
- void Serialize (ref int myInt)
- · void Serialize (ref string value)
- void Serialize (ref char value)
- void Serialize (ref short value)
- · void Serialize (ref float obj)
- · void Serialize (ref PhotonPlayer obj)
- void Serialize (ref Vector3 obj)
- void Serialize (ref Vector2 obj)
- void Serialize (ref Quaternion obj)

# **Properties**

- bool isWriting [get] • bool isReading [get]
- int Count [get]

# 13.14.1 Detailed Description

This "container" class is used to carry your data as written by OnPhotonSerializeView.

# See Also

# PhotonNetworkingMessage

13.14.2	Constructor & Destructor Documentation
13.14.2.1	PhotonStream.PhotonStream ( bool write, object[] incomingData )
13.14.3	Member Function Documentation
13.14.3.1	object PhotonStream.ReceiveNext ( )
13.14.3.2	void PhotonStream.SendNext(object obj)
13.14.3.3	void PhotonStream.Serialize ( ref bool myBool )
13.14.3.4	void PhotonStream.Serialize ( ref int mylnt )
13.14.3.5	void PhotonStream.Serialize ( ref string value )
13.14.3.6	void PhotonStream.Serialize ( ref char value )
13.14.3.7	void PhotonStream.Serialize ( ref short value )
13.14.3.8	void PhotonStream.Serialize ( ref float obj )
13.14.3.9	void PhotonStream.Serialize ( ref PhotonPlayer obj )
13.14.3.10	void PhotonStream.Serialize ( ref Vector3 obj )
13.14.3.11	void PhotonStream.Serialize ( ref Vector2 obj )
13.14.3.12	void PhotonStream.Serialize ( ref Quaternion obj )
13.14.3.13	object [] PhotonStream.ToArray ( )
13.14.4	Property Documentation
13.14.4.1	int PhotonStream.Count [get]
13.14.4.2	bool PhotonStream.isReading [get]
13.14.4.3	bool PhotonStream.isWriting [get]

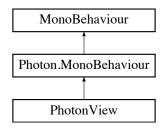
The documentation for this class was generated from the following file:

• Photon Unity Networking/Plugins/PhotonNetwork/PhotonClasses.cs

# 13.15 PhotonView Class Reference

PUN's NetworkView replacement class for networking. Use it like a NetworkView.

Inheritance diagram for PhotonView:



#### **Public Member Functions**

- · void Awake ()
  - Called by Unity on start of the application and does a setup the PhotonView.
- void OnApplicationQuit ()
- void OnDestroy ()
- void RPC (string methodName, PhotonTargets target, params object[] parameters)
- void RPC (string methodName, PhotonPlayer targetPlayer, params object[] parameters)
- override string ToString ()

# **Static Public Member Functions**

- static PhotonView Get (Component component)
- static PhotonView Get (GameObject gameObj)
- static PhotonView Find (int viewID)

#### **Public Attributes**

- int subId
- · int ownerld
- int group = 0
- int prefixBackup = -1
- · Component observed
- · ViewSynchronization synchronization
- OnSerializeTransform onSerializeTransformOption = OnSerializeTransform.PositionAndRotation
- OnSerializeRigidBody onSerializeRigidBodyOption = OnSerializeRigidBody.All
- · 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)

• int viewID [get, set]

- bool isSceneView [get]
- PhotonPlayer owner [get]
- int OwnerActorNr [get]
- boolisMine [get]

Is this photonView mine? True in case the owner matches the local PhotonPlayer ALSO true if this is a scene photonview on the Master client

# 13.15.1 Detailed Description

PUN's NetworkView replacement class for networking. Use it like a NetworkView.

#### 13.15.2 Member Function Documentation

```
13.15.2.1 void PhotonView.Awake ( )
```

Called by Unity on start of the application and does a setup the PhotonView.

```
13.15.2.2 static PhotonView PhotonView.Find (int viewID) [static]
13.15.2.3 static PhotonView PhotonView.Get ( Component component ) [static]
13.15.2.4 static PhotonView PhotonView.Get ( GameObject gameObj ) [static]
13.15.2.5 void PhotonView.OnApplicationQuit ( )
13.15.2.6 void PhotonView.OnDestroy ( )
13.15.2.7 void PhotonView.RPC ( string methodName, PhotonTargets target, params object[] parameters )
13.15.2.8 void PhotonView.RPC ( string methodName, PhotonPlayer targetPlayer, params object[] parameters )
13.15.2.9 override string PhotonView.ToString ( )
13.15.3 Member Data Documentation
13.15.3.1 int PhotonView.group = 0
13.15.3.2 int PhotonView.instantiationId
13.15.3.3 Component PhotonView.observed
13.15.3.4 OnSerializeRigidBody PhotonView.onSerializeRigidBodyOption = OnSerializeRigidBody.All
13.15.3.5 OnSerializeTransform PhotonView.onSerializeTransformOption = OnSerializeTransform.PositionAndRotation
13.15.3.6 int PhotonView.ownerld
13.15.3.7 int PhotonView.prefixBackup = -1
13.15.3.8 int PhotonView.subId
```

13.15.3.9 ViewSynchronization PhotonView.synchronization

# 13.15.4 Property Documentation

```
13.15.4.1 object[] PhotonView.instantiationData [get], [set]
```

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

```
13.15.4.2 bool PhotonView.isMine [get]
```

Is this photonView mine? True in case the owner matches the local PhotonPlayer ALSO true if this is a scene photonview on the Master client

```
13.15.4.3 bool PhotonView.isSceneView [get]
13.15.4.4 PhotonPlayer PhotonView.owner [get]
13.15.4.5 int PhotonView.OwnerActorNr [get]
13.15.4.6 int PhotonView.prefix [get], [set]
13.15.4.7 int PhotonView.viewID [get], [set]
```

The documentation for this class was generated from the following file:

• Photon Unity Networking/Plugins/PhotonNetwork/PhotonView.cs

# 13.16 Room Class Reference

This class resembles a room that PUN joins (or joined). The properties are settable as opposed to those of a RoomInfo and you can close or hide "your" room.

Inheritance diagram for Room:



# **Public Member Functions**

void SetCustomProperties (Hashtable propertiesToSet)

Updates and synchronizes the named properties of this Room with the values of properties ToSet.

# **Properties**

- new int playerCount [get]
  - Count of players in this room.
- new string name [get, set]

The name of a room. Unique identifier (per Loadbalancing group) for a room/match.

• new int maxPlayers [get, set]

Sets a limit of players to this room. This property is shown in lobby, too. If the room is full (players count == maxplayers), joining this room will fail.

• new bool open [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 before they are joined. Simply re-connect to master and find another. Use property "visible" to not list the room.

new bool visible [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.

• string[] propertiesListedInLobby [get, set]

A list of custom properties that should be forwarded to the lobby and listed there.

• bool autoCleanUp [get]

Gets if this room uses autoCleanUp to remove all (buffered) RPCs and instantiated GameObjects when a player leaves

#### **Additional Inherited Members**

#### 13.16.1 Detailed Description

This class resembles a room that PUN joins (or joined). The properties are settable as opposed to those of a RoomInfo and you can close or hide "your" room.

#### 13.16.2 Member Function Documentation

13.16.2.1 void Room.SetCustomProperties ( Hashtable propertiesToSet )

Updates and synchronizes the named properties of this Room with the values of propertiesToSet.

Any player can set a Room's properties. Room properties are available until changed, deleted or until the last player leaves the room. Access them by: Room.CustomProperties (read-only!).

New properties are added, existing values are updated. Other values will not be changed, so only provide values that changed or are new. To delete a named (custom) property of this room, use null as value. Only string-typed keys are applied (everything else is ignored).

Local cache is updated immediately, other clients are updated through Photon with a fitting operation. To reduce network traffic, set only values that actually changed.

#### **Parameters**

properties To Set | Hashtable of props to udpate, set and sync. See description.

# 13.16.3 Property Documentation

13.16.3.1 bool Room.autoCleanUp [get]

Gets if this room uses autoCleanUp to remove all (buffered) RPCs and instantiated GameObjects when a player leaves.

```
13.16.3.2 new int Room.maxPlayers [get], [set]
```

Sets a limit of players to this room. This property is shown in lobby, too. If the room is full (players count == maxplayers), joining this room will fail.

```
13.16.3.3 new string Room.name [get], [set]
```

The name of a room. Unique identifier (per Loadbalancing group) for a room/match.

```
13.16.3.4 new bool Room.open [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 before they are joined. Simply re-connect to master and find another. Use property "visible" to not list the room.

```
13.16.3.5 new int Room.playerCount [get]
```

Count of players in this room.

```
13.16.3.6 string [] Room.propertiesListedInLobby [get], [set]
```

A list of custom properties that should be forwarded to the lobby and listed there.

```
13.16.3.7 new bool Room.visible [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.

The documentation for this class was generated from the following file:

Photon Unity Networking/Plugins/PhotonNetwork/Room.cs

# 13.17 RoomInfo Class Reference

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 (open, maxPlayers, etc).

Inheritance diagram for RoomInfo:



#### **Public Member Functions**

- override bool Equals (object p)
  - Makes RoomInfo comparable (by name).
- override int GetHashCode ()

Accompanies Equals, using the name's HashCode as return.

• override string ToString ()

Simple printingin method.

#### **Protected Attributes**

byte maxPlayersField = 0

Backing field for property.

• bool openField = true

Backing field for property.

• bool visibleField = true

Backing field for property.

bool autoCleanUpField = false

Backing field for property. False unless the GameProperty is set to true (else it's not sent).

• string nameField

Backing field for property.

# **Properties**

bool removedFromList [get, set]

Used internally in lobby, to mark rooms that are no longer listed.

• Hashtable customProperties [get]

Read-only "cache" of custom properties of a room. Set via Room.SetCustomProperties (not available for RoomInfo class!).

• string name [get]

The name of a room. Unique identifier (per Loadbalancing group) for a room/match.

int playerCount [get, set]

Only used internally in lobby, to display number of players in room (while you're not in).

• bool isLocalClientInside [get, set]

State if the local client is already in the game or still going to join it on gameserver (in lobby always false).

• byte maxPlayers [get]

Sets a limit of players to this room. This property is shown in lobby, too. If the room is full (players count == maxplayers), joining this room will fail.

• bool open [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 before they are joined. Simply re-connect to master and find another. Use property "IsVisible" to not list the room.

• bool visible [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.

#### 13.17.1 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 (open, 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).

#### 13.17.2 Member Function Documentation

13.17.2.1 override bool RoomInfo.Equals (object p)

Makes RoomInfo comparable (by name).

```
13.17.2.2 override int RoomInfo.GetHashCode ( )
Accompanies Equals, using the name's HashCode as return.
Returns
13.17.2.3 override string RoomInfo.ToString ( )
Simple printingin method.
Returns
    String showing the RoomInfo.
13.17.3 Member Data Documentation
13.17.3.1 bool RoomInfo.autoCleanUpField = false [protected]
Backing field for property. False unless the GameProperty is set to true (else it's not sent).
13.17.3.2 byte RoomInfo.maxPlayersField = 0 [protected]
Backing field for property.
13.17.3.3 string RoomInfo.nameField [protected]
Backing field for property.
13.17.3.4 bool RoomInfo.openField = true [protected]
Backing field for property.
13.17.3.5 bool RoomInfo.visibleField = true [protected]
Backing field for property.
13.17.4 Property Documentation
13.17.4.1 Hashtable RoomInfo.customProperties [get]
Read-only "cache" of custom properties of a room. Set via Room.SetCustomProperties (not available for RoomInfo
All keys are string-typed and the values depend on the game/application.
13.17.4.2 bool RoomInfo.isLocalClientInside [get], [set]
```

State if the local client is already in the game or still going to join it on gameserver (in lobby always false).

13.17.4.3 byte Roominfo.maxPlayers [get]

Sets a limit of players to 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.

```
13.17.4.4 string RoomInfo.name [get]
```

The name of a room. Unique identifier (per Loadbalancing group) for a room/match.

```
13.17.4.5 bool RoomInfo.open [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 before they are joined. 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.

```
13.17.4.6 int RoomInfo.playerCount [get], [set]
```

Only used internally in lobby, to display number of players in room (while you're not in).

```
13.17.4.7 bool RoomInfo.removedFromList [get], [set]
```

Used internally in lobby, to mark rooms that are no longer listed.

```
13.17.4.8 bool RoomInfo.visible [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.

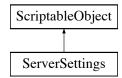
The documentation for this class was generated from the following file:

• Photon Unity Networking/Plugins/PhotonNetwork/RoomInfo.cs

# 13.18 ServerSettings Class Reference

Collection of connection-relevant settings, used internally by PhotonNetwork.ConnectUsingSettings.

Inheritance diagram for ServerSettings:



# **Public Types**

• enum HostingOption { NotSet, PhotonCloud, SelfHosted, OfflineMode }

#### **Public Member Functions**

- void UseCloud (string cloudAppid, int regionIndex)
- void UseMyServer (string serverAddress, int serverPort, string application)
- override string ToString ()

#### Static Public Member Functions

- · static int FindRegionForServerAddress (string server)
- static string FindServerAddressForRegion (int regionIndex)

#### **Public Attributes**

- HostingOption HostType = HostingOption.NotSet
- string ServerAddress = DefaultServerAddress
- int ServerPort = 5055
- string AppID = ""
- · bool DisableAutoOpenWizard

#### **Static Public Attributes**

- static string DefaultCloudServerUrl = "app-eu.exitgamescloud.com"
- static string[] CloudServerRegionNames = new string[] { "EU", "US", "Asia", "Japan" }
- static string[] CloudServerRegionPrefixes = new string[] {"app-eu", "app-us", "app-asia", "app-jp"}
- static string DefaultServerAddress = "127.0.0.1"
- static int DefaultMasterPort = 5055
- static string DefaultAppID = "Master"

# 13.18.1 Detailed Description

Collection of connection-relevant settings, used internally by PhotonNetwork.ConnectUsingSettings.

# 13.18.2 Member Enumeration Documentation

# 13.18.2.1 enum ServerSettings.HostingOption

#### **Enumerator:**

NotSet

**PhotonCloud** 

SelfHosted

OfflineMode

```
13.18.3 Member Function Documentation
13.18.3.1 static int ServerSettings.FindRegionForServerAddress ( string server ) [static]
13.18.3.2 static string ServerSettings.FindServerAddressForRegion (int regionIndex) [static]
13.18.3.3 override string ServerSettings.ToString ( )
13.18.3.4 void ServerSettings.UseCloud ( string cloudAppid, int regionIndex )
13.18.3.5 void ServerSettings.UseMyServer ( string serverAddress, int serverPort, string application )
13.18.4 Member Data Documentation
13.18.4.1 string ServerSettings.AppID = ""
13.18.4.2 string [] ServerSettings.CloudServerRegionNames = new string[] { "EU", "US", "Asia", "Japan" } [static]
13.18.4.3 string [] ServerSettings.CloudServerRegionPrefixes = new string[] {"app-eu", "app-us", "app-asia", "app-jp"}
          [static]
13.18.4.4 string ServerSettings.DefaultAppID = "Master" [static]
13.18.4.5 string ServerSettings.DefaultCloudServerUrl = "app-eu.exitgamescloud.com" [static]
13.18.4.6 int ServerSettings.DefaultMasterPort = 5055 [static]
13.18.4.7 string ServerSettings.DefaultServerAddress = "127.0.0.1" [static]
13.18.4.8 bool ServerSettings.DisableAutoOpenWizard
13.18.4.9 HostingOption ServerSettings.HostType = HostingOption.NotSet
13.18.4.10 string ServerSettings.ServerAddress = DefaultServerAddress
13.18.4.11 int ServerSettings.ServerPort = 5055
```

The documentation for this class was generated from the following file:

• Photon Unity Networking/Plugins/PhotonNetwork/ServerSettings.cs

# **Chapter 14**

# **File Documentation**

- 14.1 \_Doc/general.md File Reference
- 14.2 \_Doc/main.md File Reference
- 14.3 \_Doc/optionalGui.md File Reference
- 14.4 \_Doc/photonStatsGui.md File Reference
- 14.5 \_Doc/publicApi.md File Reference
- 14.6 Photon Unity Networking/Plugins/PhotonNetwork/CustomTypes.cs File Reference

#### 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.

# 14.7 Photon Unity Networking/Plugins/PhotonNetwork/Enums.cs File Reference

# **Enumerations**

enum ConnectionState {
 Disconnected, Connecting, Connected, Disconnecting,
 InitializingApplication }

High level connection state of the client. Better use the more detailed PeerState.

• enum PeerState {

Uninitialized, PeerCreated, Connecting, Connected,
Queued, Authenticated, JoinedLobby, DisconnectingFromMasterserver,
ConnectingToGameserver, ConnectedToGameserver, Joining, Joined,
Leaving DisconnectingFromCompositionFromCompositi

 $Leaving, Disconnecting From Games erver, Connecting To Master server, Connected Coming From Games erver, Disconnecting, Disconnected, Connected To Master \}$ 

Detailed connection / networking peer state. PUN implements a loadbalancing and authentication workflow "behind the scenes", so some states will automatically advance to some follow up state. Those states are commented with "(will-change)".

90 File Documentation

• enum PhotonNetworkingMessage {

OnConnectedToPhoton, OnLeftRoom, OnMasterClientSwitched, OnPhotonCreateRoomFailed,

OnPhotonJoinRoomFailed, OnCreatedRoom, OnJoinedLobby, OnLeftLobby,

OnDisconnectedFromPhoton, OnConnectionFail, OnFailedToConnectToPhoton, OnReceivedRoomList-Update,

OnJoinedRoom, OnPhotonPlayerConnected, OnPhotonPlayerDisconnected, OnPhotonRandomJoinFailed, OnConnectedToMaster, OnPhotonSerializeView, OnPhotonInstantiate, OnPhotonMaxCccuReached }

This enum makes up the set of MonoMessages sent by Photon Unity Networking. Implement any of these constant names as method and it will be called in the respective situation.

enum DisconnectCause {

ExceptionOnConnect = StatusCode.ExceptionOnConnect, TimeoutDisconnect = StatusCode.TimeoutDisconnect, InternalReceiveException = StatusCode.InternalReceiveException, DisconnectByServer = StatusCode.DisconnectByServer,

DisconnectByServerLogic = StatusCode.DisconnectByServerLogic, DisconnectByServerUserLimit = Status-Code.DisconnectByServerUserLimit, Exception = StatusCode.Exception, InvalidRegion = ErrorCode.Invalid-Region,

MaxCcuReached = ErrorCode.MaxCcuReached }

Summarizes the cause for a disconnect. Used in: OnConnectionFail and OnFailedToConnectToPhoton.

# 14.7.1 Enumeration Type Documentation

#### 14.7.1.1 enum ConnectionState

High level connection state of the client. Better use the more detailed PeerState.

#### **Enumerator:**

Disconnected

Connecting

Connected

Disconnecting

InitializingApplication

# 14.8 Photon Unity Networking/Plugins/PhotonNetwork/Extensions.cs File Reference

#### **Classes**

class Extensions

This static class defines some useful extension methods for several existing classes (e.g. Vector3, float and others).

# 14.9 Photon Unity Networking/Plugins/PhotonNetwork/LoadbalancingPeer.cs File Reference

# Classes

· class LoadbalancingPeer

Internally used by PUN, a LoadbalancingPeer provides the operations and enum definitions needed to use the Photon Loadbalancing server (or the Photon Cloud).

class ErrorCode

Class for constants. These (int) values represent error codes, as defined and sent by the Photon LoadBalancing logic. Pun uses these constants internally.

· class ActorProperties

Class for constants. These (byte) values define "well known" properties for an Actor / Player. Pun uses these constants internally.

class GameProperties

Class for constants. These (byte) values are for "well known" room/game properties used in Photon Loadbalancing. Pun uses these constants internally.

class EventCode

Class for constants. These values are for events defined by Photon Loadbalancing. Pun uses these constants internally.

· class ParameterCode

Class for constants. Codes for parameters of Operations and Events. Pun uses these constants internally.

class OperationCode

Class for constants. Contains operation codes. Pun uses these constants internally.

#### **Enumerations**

enum MatchmakingMode : byte { FillRoom = 0, SerialMatching = 1, RandomMatching = 2 }
 Options for matchmaking rules for OpJoinRandom.

# 14.9.1 Enumeration Type Documentation

#### 14.9.1.1 enum MatchmakingMode: byte

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.

# 14.10 Photon Unity Networking/Plugins/PhotonNetwork/NetworkingPeer.cs File Reference

### Classes

class NetworkingPeer

Implements Photon LoadBalancing used in PUN. This class is used internally by PhotonNetwork and not intended as public API.

# 14.11 Photon Unity Networking/Plugins/PhotonNetwork/PhotonClasses.cs File Reference

#### **Classes**

class PhotonNetworkMessages

Class for constants. Defines photon-event-codes for PUN usage.

92 File Documentation

· class Photon.MonoBehaviour

This class adds the property photonView, while logging a warning when your game still uses the networkView.

class PhotonMessageInfo

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

class PhotonStream

This "container" class is used to carry your data as written by OnPhotonSerializeView.

### **Namespaces**

package Photon

#### **Enumerations**

enum PhotonTargets {
 All, Others, MasterClient, AllBuffered,
 OthersBuffered }

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

• enum PhotonLogLevel { ErrorsOnly, Informational, Full }

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

# 14.12 Photon Unity Networking/Plugins/PhotonNetwork/PhotonHandler.cs File Reference

#### Classes

· class PhotonHandler

Internal Monobehaviour that allows Photon to run an Update loop.

# 14.13 Photon Unity Networking/Plugins/PhotonNetwork/PhotonLagSimulationGui.cs File Reference

#### **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.

# 14.14 Photon Unity Networking/Plugins/PhotonNetwork/PhotonNetwork.cs File Reference

#### Classes

class PhotonNetwork

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

# **Typedefs**

using Object = UnityEngine.Object

# 14.14.1 Typedef Documentation

14.14.1.1 using Object = UnityEngine.Object

# 14.15 Photon Unity Networking/Plugins/PhotonNetwork/PhotonPlayer.cs File Reference

# Classes

· class PhotonPlayer

Summarizes a "player" within a room, identified (in that room) by actorID.

# 14.16 Photon Unity Networking/Plugins/PhotonNetwork/PhotonStatsGui.cs File Reference

#### **Classes**

· class PhotonStatsGui

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

# 14.17 Photon Unity Networking/Plugins/PhotonNetwork/PhotonView.cs File Reference

#### Classes

class PhotonView

PUN's NetworkView replacement class for networking. Use it like a NetworkView.

#### **Enumerations**

- enum ViewSynchronization { Off, ReliableDeltaCompressed, Unreliable }
- enum OnSerializeTransform {
   OnlyPosition, OnlyRotation, OnlyScale, PositionAndRotation,
   All }
- enum OnSerializeRigidBody { OnlyVelocity, OnlyAngularVelocity, All }

# 14.17.1 Enumeration Type Documentation

14.17.1.1 enum OnSerializeRigidBody

**Enumerator:** 

OnlyVelocity
OnlyAngularVelocity
All

14.17.1.2 enum OnSerializeTransform

**Enumerator:** 

OnlyPosition
OnlyRotation

94 File Documentation

OnlyScale

**PositionAndRotation** 

All

14.17.1.3 enum ViewSynchronization

**Enumerator:** 

Off

ReliableDeltaCompressed

Unreliable

# 14.18 Photon Unity Networking/Plugins/PhotonNetwork/Room.cs File Reference

#### **Classes**

class Room

This class resembles a room that PUN joins (or joined). The properties are settable as opposed to those of a Room-Info and you can close or hide "your" room.

# 14.19 Photon Unity Networking/Plugins/PhotonNetwork/RoomInfo.cs File Reference

#### Classes

· 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 (open, maxPlayers, etc).

# 14.20 Photon Unity Networking/Plugins/PhotonNetwork/ServerSettings.cs File Reference

# **Classes**

class ServerSettings

Collection of connection-relevant settings, used internally by PhotonNetwork.ConnectUsingSettings.

# Index

Doc/general.md, 89	PhotonNetwork, 69
Doc/main.md, 89	Awake
Doc/optionalGui.md, 89	PhotonView, 79
_Doc/photonStatsGui.md, 89	AzureLocalNodeld
Doc/publicApi.md, 89	ParameterCode, 50
	AzureMasterNodeId
ActorList	ParameterCode, 50
ParameterCode, 49	AzureNodeInfo
ActorNr	EventCode, 41
ParameterCode, 49	ParameterCode, 50
ActorProperties, 37	ŕ
PlayerName, 37	Broadcast
Add	ParameterCode, 50
ParameterCode, 50	buttonsOn
Address	PhotonStatsGui, 75
ParameterCode, 50	
All	Cache
PhotonView.cs, 93, 94	ParameterCode, 50
Public API, 34	ChangeGroups
AllBuffered	OperationCode, 47
Public API, 34	CleanupCacheOnLeave
allProperties	GameProperties, 45
PhotonPlayer, 74	ParameterCode, 50
AllocateViewID	CloseConnection
PhotonNetwork, 59	PhotonNetwork, 59
AlmostEquals	CloudServerRegionNames
Extensions, 42, 43	ServerSettings, 87
AlreadyMatched	CloudServerRegionPrefixes
ErrorCode, 38	ServerSettings, 87
AppID	Code
ServerSettings, 87	ParameterCode, 50
AppStats	Connect
EventCode, 41	PhotonNetwork, 59
AppVersion	ConnectUsingSettings
ParameterCode, 50	PhotonNetwork, 60
ApplicationId	Connected
ParameterCode, 50	Enums.cs, 90
Authenticate	Public API, 31
OperationCode, 47	connected
Authenticated	PhotonNetwork, 69
Public API, 32	ConnectedComingFromGameserver
autoCleanUp	Public API, 32
Room, 81	ConnectedToGameserver
autoCleanUpField	Public API, 32
RoomInfo, 84	ConnectedToMaster
autoCleanUpPlayerObjects	Public API, 32
PhotonNetwork, 68	Connecting
autoJoinLobby	Enums.cs, 90
PhotonNetwork, 69	Public API, 31
automaticallySyncScene	ConnectingToGameserver

Public API, 32	Public API, 32
ConnectingToMasterserver	Disconnecting
Public API, 32	Enums.cs, 90
ConnectionState	Public API, 32
Enums.cs, 90	DisconnectingFromGameserver
connectionState	Public API, 32
PhotonNetwork, 69	DisconnectingFromMasterserver
connectionStateDetailed	Public API, 32
PhotonNetwork, 69	, , , , , , , , , , , , , , , , , , , ,
Contains	Enums.cs
Extensions, 43	Connected, 90
Count	Connecting, 90
PhotonStream, 77	Disconnected, 90
countOfPlayers	Disconnecting, 90
PhotonNetwork, 69	InitializingApplication, 90
countOfPlayersInRooms	Enums.cs
PhotonNetwork, 69	ConnectionState, 90
countOfPlayersOnMaster	Equals
PhotonNetwork, 69	PhotonPlayer, 73
countOfRooms	RoomInfo, 83
	ErrorCode, 37
PhotonNetwork, 70 CreateGame	AlreadyMatched, 38
	GameClosed, 38
OperationCode, 47	GameDoesNotExist, 38
CreateRoom	GameFull, 38
PhotonNetwork, 60, 61	GameldAlreadyExists, 39
CustomEventContent	InternalServerError, 39
ParameterCode, 51	InvalidAuthentication, 39
customProperties	InvalidOperationCode, 39
PhotonPlayer, 74	InvalidRegion, 39
RoomInfo, 84	MaxCcuReached, 39
Data	NoRandomMatchFound, 39
ParameterCode, 51	Ok, 39
DefaultAppID	OperationNotAllowedInCurrentState, 39
ServerSettings, 87	ServerFull, 40
DefaultCloudServerUrl	UserBlocked, 40
ServerSettings, 87	ErrorsOnly
DefaultMasterPort	Public API, 32
ServerSettings, 87	EventCode, 40
DefaultServerAddress	AppStats, 41
ServerSettings, 87	AzureNodeInfo, 41
Destroy	GameList, 41
PhotonNetwork, 61	GameListUpdate, 41
DestroyPlayerObjects	Join, 41
PhotonNetwork, 62	Leave, 41
DisableAutoOpenWizard	Match, 41
ServerSettings, 87	PropertiesChanged, 41
Disconnect	QueueState, 41
PhotonNetwork, 62	SetProperties, 41
DisconnectByServer	Exception
Public API, 31	Public API, 31
DisconnectByServerLogic	ExceptionOnConnect
Public API, 31	Public API, 31
DisconnectByServerUserLimit	Extensions, 42
Public API, 31	AlmostEquals, 42, 43
DisconnectCause	Contains, 43
Public API, 31	GetPhotonView, 43
Disconnected	GetPhotonViewsInChildren, 43
Enums.cs, 90	Merge, 43
	-

MergeStringKeys, 43 StripKeysWithNullValues, 43	PhotonView, 79
StripToStringKeys, 44	healthStatsVisible
ToStringFull, 44	PhotonStatsGui, 75
	HostType
FillRoom	ServerSettings, 87
LoadbalancingPeer.cs, 91	HostingOption
Find	ServerSettings, 86
PhotonPlayer, 73	
PhotonView, 79	ID
FindRegionForServerAddress	PhotonPlayer, 74
ServerSettings, 87	Informational
FindServerAddressForRegion	Public API, 32
ServerSettings, 87	InitializeSecurity
Full	PhotonNetwork, 62
Public API, 32	InitializingApplication
	Enums.cs, 90
GameClosed	insideLobby
ErrorCode, 38	PhotonNetwork, 70
GameCount	Instantiate
ParameterCode, 51	PhotonNetwork, 62, 63
GameDoesNotExist	InstantiateSceneObject
ErrorCode, 38	PhotonNetwork, 63
GameFull	instantiationData
ErrorCode, 38	PhotonView, 80
GameldAlreadyExists	instantiationId
ErrorCode, 39	PhotonView, 79
GameList	InternalReceiveException
EventCode, 41	Public API, 31
ParameterCode, 51	InternalCleanPhotonMonoFromSceneIfStuck
GameListUpdate	PhotonNetwork, 63
EventCode, 41	InternalServerError
GameProperties, 44	ErrorCode, 39
CleanupCacheOnLeave, 45	InvalidRegion
IsOpen, 45	Public API, 31
IsVisible, 45	InvalidAuthentication
MaxPlayers, 45	ErrorCode, 39
ParameterCode, 51	InvalidOperationCode
PlayerCount, 45	ErrorCode, 39
PropsListedInLobby, 45	InvalidRegion
Removed, 45	ErrorCode, 39
Get	isLocal
PhotonView, 79	PhotonPlayer, 74
GetHashCode	isLocalClientInside
PhotonPlayer, 73	RoomInfo, 84
RoomInfo, 83	isMasterClient
GetPhotonView	PhotonNetwork, 70
Extensions, 43	PhotonPlayer, 74
GetPhotonViewsInChildren	isMessageQueueRunning
Extensions, 43	PhotonNetwork, 70
GetPing	isMine
PhotonNetwork, 62	PhotonView, 80
GetProperties	isNonMasterClientInRoom
OperationCode, 47	PhotonNetwork, 70
GetRoomList	IsOpen
PhotonNetwork, 62	GameProperties, 45
Group	isReading
ParameterCode, 51	PhotonStream, 77
group	isSceneView

PhotonView, 80 IsVisible GameProperties, 45 isWriting PhotonStream, 77 PhotonStream, 77  Join PhotonNetwork, 70  MaxPlayers GameProperties, 45 maxPlayers GameProperties, 45 maxPlayersField RoomInfo, 84  Morge OperationCode, 47  JoinRandomGame OperationCode, 47  JoinRandomGom PhotonNetwork, 63, 64  JoinRoom PhotonNetwork, 63, 64  Joined Public API, 32  Joined Public API, 32  Joined Public API, 32  Joining Public API, 32  Joining Public API, 32  Joining Public API, 32  Joined Public API, 32  Leave EventCode, 47  LeaveBoom PhotonNetwork, 65  LeaveBoom PhotonNetwork, 65  Leaving Public API, 32  LeaveBoom PhotonNetwork, 65  Leaving Public API, 32  LeaveBoom PhotonNetwork, 65  Leaving Public API, 32  Doined PhotonNetwork, 65  Leaving Public API, 32  Doined PhotonNetwork, 65  NetworkStatistics Tostring PhotonNetwork, 65  NetworkStatistics Tostri		
GameProperties, 45 isWriting PhotonStream, 77 PhotonStream, 77  Join  LeventCode, 41  JoinGame OperationCode, 47  JoinCode, 47  JoinCode, 47  JoinGame OperationCode, 47  JoinRandomGame OperationCode, 47  JoinRandomRoom PhotonNetwork, 63, 64  JoinRoom PhotonNetwork, 63, 64  JoinRoom PhotonNetwork, 64  Joined Public API, 32  JoinedLobby Public API, 32  Leave EventCode, 41 OperationCode, 47  LeaveRoom PhotonNetwork, 65 Leaving Public API, 32  LeaveLobby OperationCode, 47  LeaveRoom PhotonNetwork, 65  Leaving Public API, 32  LoadLevel PhotonNetwork, 65  Leaving Public API, 32  LoadLevel PhotonNetwork, 65  Leaving Public API, 32  LoadLavel PhotonNetwork, 65  LeaveRoom PhotonNetwork, 65  LoadbalancingPeer.cs FillRoom, 91  RandomMatching, 91 SerialMatching, 91 SerialMatching, 91 SerialMatching, 91 Code Room PhotonNetwork, 67  MasterPecCourt PhotonNetwork, 67  MasterPecCourt ParameterCode, 51  Match MatchMakingType ParameterCode, 51  MatchmakingMode  OnConnectedToMoton OnConnectedToMoton Public API, 32 OnConnectedToMoton Public API, 33 OnCreatedRoom Public API, 33 OnConnectedTomPhoton		MaxCcuReached
isWriting		Public API, 31
PhotonStream, 77  Join  EventCode, 41  JoinCame OperationCode, 47  JoinLobby OperationCode, 47  JoinRandomGame OperationCode, 47  JoinRandomGame OperationCode, 47  JoinRandomGame OperationCode, 47  JoinRandomGame OperationCode, 47  JoinRandomGom PhotonNetwork, 63, 64  JoinRand PhotonNetwork, 64  Joined Public API, 32  Joining Public API, 32  Leave Leave EventCode, 47  LeaveRoom PhotonNetwork, 65  Leaving PhotonNetwork, 65  LoadbalancingPeer.cs FillRoom, 91 SerialMatching, 91 SerialMatching, 91 SerialMatching, 91 SerialMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPercCount ParameterCode, 51 Match MatchMakingType ParameterCode, 51 MatchmakingMode OperationCode, 41 MatchMakingType ParameterCode, 51 MatchmakingMode OperationCode, 47 PhotonNetwork, 70 OperationCode, 41 MatchMakingType ParameterCode, 51 MatchmakingMode OperationCode, 47 PhotonNetwork, 70 OperationCode, 41 MatchmakingMode OperationCode, 51 MacchmakingMode OperationCode, 51 MacchmakingMode OperationCode, 47 OperationCode, 51 MacchmakingMode OperationCode, 47	GameProperties, 45	
Join EventCode, 41 JoinCame OperationCode, 47 JoinLobby OperationCode, 47 JoinRandomGame OperationCode, 47 JoinRandomRoam OperationCode, 47 JoinRandomRoam OperationCode, 47 JoinRandomRoam OperationCode, 47 JoinRandomRoam PhotonNetwork, 63, 64 JoinRoa PhotonNetwork, 64 Joined Public API, 32 JoinedLobby Public API, 32 JoinedLobby Public API, 32 JoinedLoby Public API, 32 Leave EventCode, 41 OperationCode, 47 Leavelcoby OperationCode, 47 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 CoadbalancingPeer.cs MatchmakingMode, 91 IogLevel PhotonNetwork, 67 MasterCilent PhotonNetwork, 70 MasterCilent PhotonNetwork, 70 MasterCount ParameterCode, 51 Match MatchMakingType ParameterCode, 51 MatchmakingMode OnelsconnectedFromPhoton Public API, 33 OnconnectedTopPhoton Public API, 33 OnconnectedFromPhoton	•	
Join EventCode, 41 JoinCame OperationCode, 47 JoinLobby OperationCode, 47 JoinCame PhotonNetwork, 63, 64 JoinCame PhotonNetwork, 64 Joined Public API, 32 JoinedLobby Public API, 32 Joining Public API, 32 Joining Public API, 32 Joining Public API, 32 Joining Public API, 32 IcaveCode, 41 OperationCode, 47 LeaveLobby OperationCode, 47 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 IogLevel PhotonNetwork, 67 MasterClient Public API, 34 masterClient Public API, 34 masterClient Public API, 34 masterClient Public API, 34 masterClient Public API, 33 MatchmakingType ParameterCode, 51 Match MatchMakingType ParameterCode, 51 MatchmakingMode OnDisconnectedTromPhoton	PhotonStream, 77	
EventCode, 41 JoinGame OperationCode, 47 JoinLobby OperationCode, 47 JoinRandomGame OperationCode, 47 JoinRandomGame OperationCode, 47 JoinRandomGame OperationCode, 47 JoinRandomGom PhotonNetwork, 63, 64 Joined Public API, 32 JoinedLobby Public API, 32 Joining Public API, 32 Joining Public API, 32 Joining Public API, 32 JoinedLobby Public API, 32 Leave EventCode, 41 OperationCode, 47 LeaveLobby OperationCode, 47 LeaveLobby OperationCode, 47 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SorialMatching, 91 Off PhotonNetwork, 67 MasterClient Public API, 34 masterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 Match MatchMakingType ParameterCode, 51 MatchmakingMode OnDisconnectedTromPhoton		
JoinGame OperationCode, 47 JoinLobby OperationCode, 47 JoinRandomGame OperationCode, 47 JoinRandomGame OperationCode, 47 JoinRandomGame OperationCode, 47 JoinRandomGame OperationCode, 47 JoinRandomGom PhotonNetwork, 63, 64 Joined Public API, 32 JoinedLobby Public API, 32 Joining Public API, 32 Leave Leave EventCode, 41 OperationCode, 47 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 LoadLaeler PhotonNetwork, 65 Leaveling PhotonNetwork, 65 Leaveling PhotonNetwork, 65 Leaveling PhotonNetwork, 65 Leaveling PhotonNetwork, 65 CadabalancingPeer.cs FillRoom, 91 RandomMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 IogLevel PhotonNetwork, 67 MasterClient Public API, 34 RasterClient Public API, 34 RasterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 Match MatchMakingType ParameterCode, 51 MatchmakingMode OnDisconnectedTomPhoton		
OperationCode, 47 JoinLobby OperationCode, 47 JoinRandomGame OperationCode, 47 JoinRandomGom Extensions, 43 PhotonNetwork, 63, 64 JoinRandomRoom Extensions, 43 PhotonNetwork, 64 Joined Public API, 32 JoinedLobby Public API, 32 Joining Public API, 32 Joining Public API, 32 Joining Public API, 32 JoinedLobby Public API, 32 Joining Public API, 32 Joining Public API, 32 Joining Public API, 32 Joining Public API, 32 JoinedLobby Recomplished Recomplished PhotonNetwork, 70 Leave EventCode, 41 OperationCode, 47 LeaveLobby NetworkStatistics Fleset PhotonNetwork, 65 Leavel Code, 41 CoperationCode, 47 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 LoadLevel Notaming PhotonNetwork, 65 Leaving PhotonNetwork, 65 Leaving PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 SerialMatching, 91 SerialMatching, 91 SerialMatching, 91 OperationCode, 47 PhotonNetwork, 67 MatchmakingMode, 91 loglevel PhotonNetwork, 67 MatchmakingMode, 91 loglevel PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 MatchmakingMode ParameterCode, 51 MatchmakingMode OnConnectedToPhoton Public API, 33 MatchmakingMode OnConnectedToPhoton Public API, 33 MatchmakingMode OnCreatedRoom Public API, 33 MatchmakingMode OnCreatedFoomPhoton		• •
JoinLobby OperationCode, 47		_
OperationCode, 47 JoinRandomGame OperationCode, 47 JoinRandomGame OperationCode, 47 JoinRandomGoom PhotonNetwork, 63, 64 JoinRoom PhotonNetwork, 64 Joined Public API, 32 Joining Public API, 32 Joining Public API, 32 Leave EventCode, 41 OperationCode, 47 LeaveLobby OperationCode, 47 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 SerialMatching, 91 SerialMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 MatchmakingMode ParameterCode, 51 MatchmakingMode MatchmakingMode OperatedFormPhoton ParameterCode, 51 MatchmakingMode  Public API, 33 MatchmakingMode OnDisconnectedFormPhoton Public API, 33 OnCreatedRoom Public API, 33 OnCreatedRoom PhotonNetwork, 70 Public API, 33 OnCreatedRoom PhotonNetwork, 70 Public API, 33 OnCreatedRoom PhotonNetwork, 70 Public API, 33 OnCreatedRoom Public API, 33 OnCreatedRoom Public API, 33 OnCreatedRoom Public API, 33 OnConnectedFormPhoton	·	,
JoinRandomGame OperationCode, 47 JoinRandomRoom PhotonNetwork, 63, 64 JoinRoom PhotonNetwork, 64 Joined Public API, 32 JoinedLobby Public API, 32 Joining Public API, 32 Public API, 32 Leave EventCode, 41 OperationCode, 47 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 LoadLebey PhotonNetwork, 65 Leaving Public API, 32 LoadLebey PhotonNetwork, 65 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 MasterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 MatchmakingMode PhotonNetwork, 70 MatchmakingMode PhotonNetwork, 70 PhotonNetwork, 65 PhotonNetwork, 65 PhotonNetwork, 65 Concept PhotonNetwork, 65 PhotonNetwork, 65 PhotonNetwork, 65 PhotonNetwork, 65 PhotonNetwork, 65 PhotonNetwork, 65 Corrocode, 39 NotSet PhotonNetwork, 67 PhotonNetwork, 70 Photon	•	
OperationCode, 47 JoinRandomRoom PhotonNetwork, 63, 64 JoinRoom PhotonNetwork, 64 Joined Public API, 32 Joined API, 32 Joining API	•	
JoinRandomRoom PhotonNetwork, 63, 64 JoinRoom PhotonNetwork, 64 Joined Public API, 32 JoinedLobby Public API, 32 Joining Public API, 32 PhotonNetwork, 64  Leave EventCode, 41 OperationCode, 47 LeaveLobby PhotonNetwork, 65 Leaving Public API, 32 Leave PhotonNetwork, 65 Leaving Public API, 32 Leave PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 MatchmakingMode ParameterCode, 51 MatchmakingMode ParameterCode, 51 MatchmakingMode ParameterCode, 51 MatchmakingMode  ParameterCode, 51 MatchmakingMode  PhotonNetwork API, 33 OnDisconnectedFormPhoton		
PhotonNetwork, 63, 64 JoinRoom PhotonNetwork, 64 Joined Public API, 32 JoinedLobby Public API, 32 Public API, 32 Joining Public API, 32 Public API, 32 Public API, 32 Joining Public API, 32 RoomInfo, 84 NetworkStatisticsEnabled PhotonNetwork, 70 NetworkStatisticsEnabled PhotonNetwork, 70 NetworkStatisticsToString OperationCode, 47 LeaveLobby OperationCode, 47 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 MAX_VIEW_IDS PhotonNetwork, 67 MasterClient Public API, 34 RerrorCode, 39 Moster PhotonNetwork, 67 MasterPeerCount ParameterCode, 51 MatchmakingType ParameterCode, 51 MatchmakingMode  MergeStringKeys Extensions, 43 MergeStringKeys Extensions, 43 MergestringKeys Extensions, 43 RoomInfo, 84 NetworkStatisticsEnabled PhotonNetwork, 65 NetworkStatisticsEnabled PhotonNetwork, 65 nameField RoomInto, 84 NetworkStatisticsEnabled PhotonNetwork, 70 PhotonNetwork, 65 nameField RoomInto, 84 NetworkStatisticsEnabled PhotonNetwork, 65 nameField RoomInto, 84 NetworkStatisticsEnabled PhotonNetwork, 70 PhotonNetwork, 65 nameField RoomInto, 84 NetworkStatisticsEnabled PhotonNetwork, 70 PhotonNetwork, 65 nameField RoomInto, 84 NetworkStatisticsEnabled PhotonNetwork, 65 nameField RoomInto, 84 NetworkStatisticsPeet PhotonNetwork, 65 nameField RoomInto, 84 NetworkStatisticsPeet Phot	•	•
JoinRoom PhotonNetwork, 64 Joined Public API, 32 JoinedLobby Public API, 32 Joining Public API, 32 RoomInfo, 85 RoomInfo, 85 RoomInfo, 84 NetworkStatisticsEnabled PhotonNetwork, 70 NetworkStatisticsReset PhotonNetwork, 70 NetworkStatisticsToString OperationCode, 47 LeaveLobby OperationCode, 47 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 IogLevel PhotonNetwork, 67 MasterClient Public API, 34 Room, 81 RetrorCode, 39 NotSet PhotonNetwork, 67 MasterClient PhotonNetwork, 67 MasterClient PhotonNetwork, 67 MasterPeerCount ParameterCode, 51 MatchmakingMode ParameterCode, 51 MatchmakingMode OnDisconnectedFromPhoton		
PhotonNetwork, 64 Joined Public API, 32 JoinedLobby Public API, 32 PhotonPlayer, 74 Room, 81 RoomInfo, 85 RoomInfo, 85 RoomInfo, 84 NetworkStatisticsEnabled PhotonNetwork, 70 Reversible API, 32 PhotonNetwork, 65 Leave PhotonNetwork, 65 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 PhotonNetwork, 65 Leaving Public API, 32 PhotonNetwork, 65 Leaving Public API, 32 EventCode, 47 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 EventCode, 39 NotSet PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 PhotonNetwork, 67 MasterClient Public API, 34 ErrorCode, 39 NotSet PhotonNetwork, 67 MasterClient PhotonNetwork, 67 MasterPeerCount ParameterCode, 51 MatchmakingMode ParameterCode, 51 MatchmakingMode OnDisconnectedFromPhoton		
Joined Public API, 32 JoinedLobby Public API, 32 Joining Public API, 32 PhotonPlayer, 74 Room, 81 Roomlnfo, 85 Roomlnfo, 85 Roomlnfo, 84 NetworkStatisticsEnabled PhotonNetwork, 70 NetworkStatisticsEnabled PhotonNetwork, 70 NetworkStatisticsToString OperationCode, 47 LeaveLobby OperationCode, 47 LeaveRoom PhotonNetwork, 65 PhotonNetwork, 65 Leaving Public API, 32 LeaveRoom PhotonNetwork, 65 PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FiilRoom, 91 RandomMatching, 91 SerialMatching, 91 SorialMatching, 91 Object PhotonNetwork, 67 PhotonNetwork, 67 MatchmakingMode, 91 IogLevel PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 MatchmakingType ParameterCode, 51 MatchmakingMode Photic API, 33 MatchmakingMode OnDisconnectedFromPhoton		Extensions, 43
Public API, 32 JoinedLobby Public API, 32 Joining Public API, 32 Roomlofo, 85 Roomlofo, 85 Roomlofo, 84 NetworkStatisticsEnabled PhotonNetwork, 70 NetworkStatisticsEnabled PhotonNetwork, 65 Leave PhotonNetwork, 65 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 IogLevel PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 MatchmakingType ParameterCode, 51 MatchmakingMode Public API, 33 OnCreatedRoom Public API, 33 OnCreatedFromPhoton Public API, 33 OnCinectedFromPhoton		
JoinedLobby Public API, 32  Joining Public API, 32  RoomInfo, 85  nameField RoomInfo, 84  NetworkStatisticsEnabled PhotonNetwork, 70  NetworkStatisticsReset PhotonNetwork, 65  LeaveLobby OperationCode, 47  LeaveRoom PhotonNetwork, 65  Leaving Public API, 32  Leaving PhotonNetwork, 65  Leaving PhotonNetwork, 65  Leaving PhotonNetwork, 65  LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67  MasterClient Public API, 34  masterClient PhotonNetwork, 70  MasterPeerCount ParameterCode, 51  MatchmakingType ParameterCode, 51  MatchmakingMode ParameterCode, 51  MatchmakingMode  NewnrkStatisticsReset PhotonNetwork, 65  NetworkStatisticsToString PhotonNetwork, 65  NetworkStatisticsToString PhotonNetwork, 65  NetworkStatisticsToString PhotonNetwork, 65  NetworkStatisticsToString PhotonNetwork, 65  NoRandomMatchFound ErrorCode, 39  NotSet ServerSettings, 86  Object PhotonNetwork.cs, 93  observed PhotonNetwork.cs, 93  observed PhotonNetwork, 79  Off PhotonView.cs, 94  OfflineMode ServerSettings, 86  offlineMode ServerSettings, 86  offlineMode ServerSettings, 86  offlineMode PhotonNetwork, 70  Ok ErrorCode, 39  OnConnectedToMaster Public API, 33  OnConnectedToPhoton Public API, 33  OnConnectionFail Public API, 33  MatchmakingType ParameterCode, 51  MatchmakingMode OnDisconnectedFromPhoton		
Public API, 32 Joining Public API, 32 RoomInfo, 85 nameField RoomInfo, 84 NetworkStatisticsEnabled PhotonNetwork, 70 NetworkStatisticsReset OperationCode, 47 LeaveLobby OperationCode, 47 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 Leavelsoby PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 SerialMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 IogLevel PhotonNetwork, 67 MAX_VIEW_IDS PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 MatchMakingType ParameterCode, 51 MatchmakingMode Polic API, 33 OnCreatedFromPhoton		
Joining Public API, 32 RoomInfo, 84 NetworkStatisticsEnabled PhotonNetwork, 70 NetworkStatisticsEnabled PhotonNetwork, 65 LeaveLobby OperationCode, 47 LeaveRoom PhotonNetwork, 65 Leaving PhotonNetwork, 65 Leaving PhotonNetwork, 65 Leaving PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 IogLevel PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 MatchmakingType ParameterCode, 51 MatchmakingMode  Leave PhotonNetwork Public API, 33 MatchmakingMode PhotonNetwork, 70 PhotonNetwork, 70 ParameterCode, 51 MatchmakingMode PhotonNetwork 70 Public API, 32 MatchmakingMode OnConnectedToPhoton Public API, 32 MatchmakingType ParameterCode, 51 Public API, 33 MatchmakingMode OnDisconnectedFromPhoton	•	
Public API, 32  RoomInfo, 84  NetworkStatisticsEnabled PhotonNetwork, 70  RevertCode, 41 OperationCode, 47  LeaveLobby OperationCode, 47  LeaveRoom PhotonNetwork, 65  LeaveRoom PhotonNetwork, 65  Leaving PhotonNetwork, 65  LoadbalancingPeer.cs FillRoom, 91  CoadbalancingPeer.cs MatchmakingMode, 91  Ioglevel PhotonNetwork, 67  MAX_VIEW_IDS PhotonNetwork, 67  MasterClient PhotonNetwork, 70  MasterPeerCount ParameterCode, 51  MatchmakingType ParameterCode, 51  MatchmakingMode  Public API, 33  MatchmakingMode  NetworkStatisticsToString PhotonNetwork, 65 NetworkStatisticsT		
Leave EventCode, 41 OperationCode, 47 LeaveLobby OperationCode, 47 LeaveRoom PhotonNetwork, 65 LeaveRoom PhotonNetwork, 65 Leaving PhotonNetwork, 65 PhotonNetwork, 65 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 SerialMatching, 91 CoadbalancingPeer.cs MatchmakingMode, 91 Ogf IngLevel PhotonNetwork, 67 MAX_VIEW_IDS PhotonNetwork, 67 MAX_VIEW_IDS PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 MatchMakingType ParameterCode, 51 MatchmakingMode PhotionNetwork, 33 MatchmakingMode PhotionPetonNetwork Public API, 33 MatchmakingMode OnConnectedToPhoton Public API, 33 MatchmakingMode OnConnectedFromPhoton	•	
Leave PhotonNetwork, 70 EventCode, 41 OperationCode, 47 LeaveLobby NetworkStatisticsToString OperationCode, 47 LeaveRoom PhotonNetwork, 65 Leaving PhotonNetwork, 65 Leaving PhotonNetwork, 65 Leaving PhotonNetwork, 65 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 Off MasterClient Ok Public API, 34 masterClient Public API, 34 MatchMakingType ParameterCode, 51 MatchmakingMode MatchmakingMode ParameterCode, 51 MatchmakingMode MatchmakingMode MatchmakingMode ParameterCode, 51 MatchmakingMode MatchmakingMode Network API, 33 MatchmakingMode OnDisconnectedFromPhoton MatchmakingMode OnDisconnectedFromPhoton	Public API, 32	
EventCode, 41 OperationCode, 47 LeaveLobby OperationCode, 47 LeaveRoom PhotonNetwork, 65 Leaving PhotonNetwork, 65 LoadblancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 PhotonNetwork, 67 OfflineMode ServerSettings, 86  MAX_VIEW_IDS PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPeerCount PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 Match MatchMakingType ParameterCode, 51 MatchmakingMode  NetworkStatistics ToString PhotonNetwork, 65 NetworkStatistics ToString PhotonNetwork, 65 NetworkStatistics ToString Photon:Network, 46 NoRandomMatching, 91 NoRandomMatchFound ErrorCode, 39 OfflineMode PhotonNetwork, 70 Ook Public API, 33 OnConnectedToPhoton Public API, 32 OnConnectedToPhoton Public API, 32 MatchMakingType ParameterCode, 51 Public API, 33 MatchmakingMode OnDisconnectedFromPhoton		
OperationCode, 47 LeaveLobby OperationCode, 47 LeaveRoom PhotonNetwork, 65 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 IogLevel PhotonNetwork, 67 AX_VIEW_IDS PhotonNetwork, 67 MasterClient Public API, 34 MatchMakingType ParameterCode, 51 MatchmakingMode PhatonNetwork, 70 MatchmakingMode ParameterCode, 51 MatchmakingMode PhatonNetwork, 33 MatchmakingMode PhatonNetwork, 70 OnConnectedToPhoton Public API, 32 MatchmakingType ParameterCode, 51 MatchmakingMode MatchmakingMode PhotonNetwork, 33 MatchmakingMode OnConnectedFromPhoton Public API, 33 MatchmakingMode OnDisconnectedFromPhoton		
LeaveLobby OperationCode, 47 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 Ax_VIEW_IDS PhotonNetwork, 67 PhotonNetwork, 67 PhotonNetwork, 67 MasterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 41 MatchMakingType ParameterCode, 51 MatchmakingMode PhotonNetwork, 67 Person Retails institution on ConnectedTophoton ParameterCode, 51 MatchmakingMode PhotonNetwork, 33 OnDisconnectedFromPhoton Public API, 33 MatchMakingType ParameterCode, 51 MatchmakingMode OnDisconnectedFromPhoton		
OperationCode, 47 LeaveRoom PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 41 MatchMakingType ParameterCode, 51 MatchmakingMode Platin Apilon ParameterCode, 51 MatchmakingMode PhotonNetwork, 67 Parion Network, 67 ParameterCode, 51 MatchmakingMode ParameterCode, 51 MatchmakingMode PhotonNetwork, 67 ParameterCode, 51 MatchmakingMode Public API, 33 OnConnectedFromPhoton ParameterCode, 51 MatchmakingMode OnDisconnectedFromPhoton Public API, 33 MatchmakingMode OnDisconnectedFromPhoton	•	
LeaveRoom PhotonNetwork, 65 Photon::MonoBehaviour, 46 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 MasterClient PhotonNetwork, 67 MasterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 MatchMakingType ParameterCode, 51 MatchMakingMode ParameterCode, 51 MatchmakingMode Plobic API, 33 MatchMakingMode ParameterCode, 51 MatchMakingMode PhotonNetwork, 70 PhotonNetwork, 70 ParameterCode, 51 MatchMakingMode PhotonNetwork, 70 ParameterCode, 51 MatchmakingMode PhotonNetwork, 70 ParameterCode, 51 Public API, 33 MatchmakingMode OnConnectedToom ParameterCode, 51 Public API, 33 MonoreatedRoom P	-	_
PhotonNetwork, 65 Leaving Public API, 32 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 RandomMatching, 91 Cobject PhotonNetwork.cs, 93 Observed PhotonView, 79 Off IngLevel PhotonNetwork, 67 MaterClient Public API, 34 MaterPeerCount ParameterCode, 51 MatchMakingType ParameterCode, 51 MatchMakingType ParameterCode, 51 MatchmakingMode Public API, 33 MatchmakingMode PhotonNetwork, 33 NonConnectedFromPhoton Public API, 33 MatchmakingType ParameterCode, 51 MatchmakingMode PhotonNetwork, 70 Public API, 33 MatchmakingMode OnDisconnectedFromPhoton	•	
Leaving Public API, 32 LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MatchmakingMode, 51 Match EventCode, 41 MatchMakingType ParameterCode, 51 MatchMakingType ParameterCode, 51 MatchMakingMode  PhotonNetwork, 67 RefrorCode, 39 NoRandomMatchFound ErrorCode, 39 NotSet ServerSettings, 86 Object PhotonNetwork.cs, 93 Observed PhotonNetwork.cs, 93 Observed PhotonView.cs, 94 OfflineMode PhotonView.cs, 94 OfflineMode ServerSettings, 86 OfflineMode PhotonNetwork, 70 Ok PhotonNetwork, 70 Ok ErrorCode, 39 OnConnectedToMaster Public API, 33 OnConnectedToPhoton Public API, 32 OnConnectionFail Public API, 33 MatchMakingType ParameterCode, 51 Public API, 33 MatchmakingMode OnDisconnectedFromPhoton		
Public API, 32  LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 MatchMakingType ParameterCode, 51 MatchmakingMode PhotonNetwork, 67 MatchmakingType ParameterCode, 51 MatchMakingType ParameterCode, 51 MatchmakingMode  ServerSettings, 86 Mother ServerSettings, 86 OfflineMode PhotonNetwork, 70 OfflineMode PhotonNetwork, 70 OnConnectedToMaster Public API, 33 OnConnectedToPhoton Public API, 33 OnCreatedRoom ParameterCode, 51 Public API, 33 MatchmakingMode OnDisconnectedFromPhoton		
LoadLevel PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 MatchMakingType ParameterCode, 51 MatchMakingMode  PhotonNetwork, 65  NotSet ServerSettings, 86 ServerSettings, 93 Object PhotonNetwork.cs, 93 Object PhotonNetwork.cs, 93 Object PhotonNetwork.cs, 93 Object PhotonNetwork.cs, 93 OfflineMode PhotonView.cs, 94 OfflineMode ServerSettings, 86 OfflineMode PhotonNetwork, 70 Ok Public API, 34 FrrorCode, 39 OnConnectedToMaster PhotonNetwork, 70 OnConnectedToPhoton Public API, 33 OnConnectedToPhoton Public API, 33 OnConnectedToPhoton Public API, 33 OnConnectedRoom Public API, 33 OnConnectedRoom Public API, 33 OnConnectedFromPhoton		
PhotonNetwork, 65 LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 Description observed LoadbalancingPeer.cs MatchmakingMode, 91 Doff DogLevel PhotonNetwork, 67 PhotonNetwork, 67 PhotonNetwork, 67 PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 MatchMakingType ParameterCode, 51 MatchmakingMode  Pobject Dobject PhotonNetwork.cs, 93 Dobject PhotonNetwork.cs, 93 Doff PhotonNetwork.cs, 93 Doff PhotonView.cs, 94 Doff PhotonView.cs, 94 Doff Doff DiffineMode DofflineMode PhotonNetwork, 70 DofflineMode PhotonNetwork, 70 DofflineMode PhotonNetwork, 70 DofflineMode PhotonNetwork, 70 DofflineMode DofflineMode DofflineMode PhotonNetwork, 70 DofflineMode D	•	
LoadbalancingPeer.cs FillRoom, 91 RandomMatching, 91 SerialMatching, 91 Object PhotonNetwork.cs, 93 Observed  LoadbalancingPeer.cs MatchmakingMode, 91  logLevel PhotonNetwork, 67 PhotonNetwork, 67 OfflineMode ServerSettings, 86  MAX_VIEW_IDS PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPeerCount PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 Match MatchMakingType ParameterCode, 51 MatchmakingMode  Object PhotonNetwork.cs, 93 PhotonNetwork.cs, 93 OfflineMode PhotonNetwo.cs, 94 OfflineMode PhotonNetwork, 70 PhotonNetwork, 70 PhotonNetwork, 70 Public API, 33 OnConnectedToMaster Public API, 33  MatchMakingType ParameterCode, 51 Public API, 33  MatchmakingMode OnDisconnectedFromPhoton		
FillRoom, 91 RandomMatching, 91 ReindMatching, 91 ReindMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 PhotonNetwork, 67 PhotonNetwork, 67 MasterClient Public API, 34 masterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 Match MatchMakingType ParameterCode, 51 MatchmakingMode  PhotonNetwork, 67 ReindMater PhotonNetwork, 70 PhotonNetwork, 70 Public API, 33 MatchMakingType ParameterCode, 51 MatchmakingMode  PhotonNetwork, 70 Public API, 33 MatchmakingMode  OnConnectedToPhoton Public API, 33 OnCreatedRoom Public API, 33 OnCreatedRoom Public API, 33 OnCreatedFromPhoton		ServerSettings, 86
RandomMatching, 91 SerialMatching, 91 LoadbalancingPeer.cs MatchmakingMode, 91 logLevel PhotonNetwork, 67 PhotonNetwork, 67 PhotonNetwork, 67  MAX_VIEW_IDS PhotonNetwork, 67 MasterClient Public API, 34 MasterPeerCount PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 Match MatchMakingType ParameterCode, 51 MatchMakingType ParameterCode, 51 MatchmakingMode  PhotonNetwork, cs, 94 PhotonView.cs, 94 PhotonView.cs, 94 OfflineMode ServerSettings, 86 OfflineMode PhotonNetwork, 70 Ok PhotonNetwork, 70 Ok ErrorCode, 39 OnConnectedToMaster Public API, 33 OnConnectedToPhoton Public API, 32 OnConnectionFail Public API, 33 MatchMakingType ParameterCode, 51 Public API, 33 OnCreatedRoom Public API, 33 MatchmakingMode OnDisconnectedFromPhoton		Ohioat
SerialMatching, 91 LoadbalancingPeer.cs     MatchmakingMode, 91  logLevel     PhotonNetwork, 67     PhotonNetwork, 67     ServerSettings, 86  MAX_VIEW_IDS     PhotonNetwork, 67     Asser Client     Public API, 34     master Client     PhotonNetwork, 70 MasterPeerCount     ParameterCode, 51     MatchMakingType     ParameterCode, 51 MatchmakingMode  PhotonNetwork, 70 Observed PhotonView, 79 OfflineMode ServerSettings, 86 OfflineMode PhotonNetwork, 70 Ok PhotonNetwork, 70 Ok ErrorCode, 39 OnConnectedToMaster Public API, 33 OnConnectedToPhoton Public API, 32 OnConnectionFail Public API, 33 OnCreatedRoom Public API, 33 OnCreatedRoom Public API, 33 OnCreatedRoom Public API, 33 OnCreatedFromPhoton		-
LoadbalancingPeer.cs MatchmakingMode, 91  logLevel PhotonNetwork, 67 PhotonNetwork, 67 PhotonNetwork, 67  MasterClient PhotonNetwork, 70  MasterClient PhotonNetwork, 70  MasterClient PhotonNetwork, 70  MasterPeerCount ParameterCode, 51  Match MatchMakingType ParameterCode, 51  MatchmakingMode  PhotonView, 79  Off PhotonView.cs, 94  OfflineMode ServerSettings, 86  OfflineMode PhotonNetwork, 70  Ook PhotonNetwork, 70  Ook Procode, 39  OnConnectedToMaster Public API, 33  OnConnectedToPhoton Public API, 32  OnConnectionFail Public API, 33  OnCreatedRoom Public API, 33  OnCreatedRoom Public API, 33  OnCreatedFromPhoton	<del>-</del>	
MatchmakingMode, 91  logLevel PhotonNetwork, 67  PhotonNetwork, 67  MAX_VIEW_IDS OfflineMode PhotonNetwork, 67  MasterClient Ok Public API, 34  masterClient OnConnectedToMaster PhotonNetwork, 70  MasterPeerCount ParameterCode, 51  Match EventCode, 41  MatchMakingType ParameterCode, 51  MatchmakingMode  OfflineMode ServerSettings, 86  OfflineMode PhotonNetwork, 70  Ok PhotonNetwork, 70  OnConnectedToMaster Public API, 33  OnConnectedToPhoton Public API, 32  OnConnectionFail Public API, 33  OnCreatedRoom ParameterCode, 51  Public API, 33  OnCreatedRoom ParameterCode, 51  Public API, 33  OnCreatedRoom ParameterCode, 51  OnConnectedFromPhoton	<b>.</b>	
IngLevel PhotonNetwork, 67 PhotonNetwork, 67 PhotonNetwork, 67  MAX_VIEW_IDS PhotonNetwork, 67 PhotonNetwork, 70  MasterClient Public API, 34  masterClient PhotonNetwork, 70  MasterPeerCount ParameterCode, 51  Match EventCode, 41  MatchMakingType ParameterCode, 51  MatchmakingMode  PhotonView.cs, 94  OfflineMode ServerSettings, 86  OfflineMode PhotonNetwork, 70  Ok PhotonNetwork, 70  Ok ErrorCode, 39  OnConnectedToMaster Public API, 33  OnConnectedToPhoton Public API, 32  OnConnectionFail Public API, 33  OnCreatedRoom ParameterCode, 51  Public API, 33  OnCreatedRoom Public API, 33  OnDisconnectedFromPhoton		
PhotonNetwork, 67  PhotonNetwork, 67  MAX_VIEW_IDS PhotonNetwork, 67  MasterClient Public API, 34  masterClient PhotonNetwork, 70  MasterPeerCount ParameterCode, 51  Match EventCode, 41  MatchMakingType ParameterCode, 51  MatchmakingMode  OfflineMode ServerSettings, 86  OfflineMode  PhotonNetwork, 70  Ok Prodoc, 39  OnConnectedToMaster Public API, 33  OnConnectedToPhoton Public API, 32  OnCreatedRoom Public API, 33  OnCreatedRoom OnDisconnectedFromPhoton	•	
ServerSettings, 86  MAX_VIEW_IDS PhotonNetwork, 67 PhotonNetwork, 70  MasterClient Public API, 34 ErrorCode, 39  masterClient PhotonNetwork, 70  MasterPeerCount ParameterCode, 51 Match EventCode, 41  MatchMakingType ParameterCode, 51  Public API, 33  MatchmakingMode  ServerSettings, 86  offlineMode OfflineMode OnConNetwork, 70 PhotonNetwork, 70 OnConnectedToMaster Public API, 33 OnConnectedToPhoton Public API, 32 OnConnectionFail Public API, 33 OnCreatedRoom ParameterCode, 51 Public API, 33 OnCreatedRoom Polic API, 33 OnDisconnectedFromPhoton	<del>-</del>	•
MAX_VIEW_IDS PhotonNetwork, 67 PhotonNetwork, 67 MasterClient Public API, 34 ErrorCode, 39 masterClient PhotonNetwork, 70 MasterPeerCount ParameterCode, 51 Match EventCode, 41 MatchMakingType ParameterCode, 51 MatchmakingMode  offlineMode PhotonNetwork, 70 PhotonNetwork, 70 OnConnectedToMaster Public API, 33 OnConnectedToPhoton Public API, 32 OnConnectionFail Public API, 33 OnCreatedRoom ParameterCode, 51 Public API, 33 OnCreatedRoom Polic API, 33 OnDisconnectedFromPhoton	PhotonNetwork, 67	
PhotonNetwork, 67  MasterClient Public API, 34  masterClient PhotonNetwork, 70  MasterClient PhotonNetwork, 70  MasterPeerCount ParameterCode, 51  Match EventCode, 41  MatchMakingType ParameterCode, 51  Public API, 33  MatchmakingMode  PhotonNetwork, 70  ConConnectedToPhoton Public API, 32  OnConnectionFail Public API, 33  OnCreatedRoom Public API, 33  OnCreatedRoom Public API, 33  OnDisconnectedFromPhoton	MAY VIEW IDC	<del>-</del>
MasterClientOkPublic API, 34ErrorCode, 39masterClientOnConnectedToMasterPhotonNetwork, 70Public API, 33MasterPeerCountOnConnectedToPhotonParameterCode, 51Public API, 32MatchOnConnectionFailEventCode, 41Public API, 33MatchMakingTypeOnCreatedRoomParameterCode, 51Public API, 33MatchmakingModeOnDisconnectedFromPhoton		
Public API, 34  masterClient		•
masterClient PhotonNetwork, 70 Public API, 33  MasterPeerCount ParameterCode, 51 Public API, 32  Match EventCode, 41 MatchMakingType ParameterCode, 51 Public API, 33  MatchmakingMode OnConnectedToPhoton Public API, 32  OnConnectionFail Public API, 33  OnCreatedRoom Public API, 33  OnCreatedRoom Public API, 33		
PhotonNetwork, 70 Public API, 33  MasterPeerCount ParameterCode, 51 Public API, 32  Match EventCode, 41 Public API, 33  MatchMakingType ParameterCode, 51 Public API, 33  MatchmakingMode OnCreatedRoom Public API, 33  OnCreatedRoom Public API, 33  OnDisconnectedFromPhoton		
MasterPeerCountOnConnectedToPhotonParameterCode, 51Public API, 32MatchOnConnectionFailEventCode, 41Public API, 33MatchMakingTypeOnCreatedRoomParameterCode, 51Public API, 33MatchmakingModeOnDisconnectedFromPhoton		
ParameterCode, 51  Match  EventCode, 41  MatchMakingType  ParameterCode, 51  MatchmakingMode  ParameterCode, 51  Public API, 33  OnCreatedRoom  Public API, 33  OnDisconnectedFromPhoton		•
MatchOnConnectionFailEventCode, 41Public API, 33MatchMakingTypeOnCreatedRoomParameterCode, 51Public API, 33MatchmakingModeOnDisconnectedFromPhoton		
EventCode, 41 Public API, 33  MatchMakingType OnCreatedRoom ParameterCode, 51 Public API, 33  MatchmakingMode OnDisconnectedFromPhoton		
MatchMakingTypeOnCreatedRoomParameterCode, 51Public API, 33MatchmakingModeOnDisconnectedFromPhoton		
ParameterCode, 51 Public API, 33  MatchmakingMode OnDisconnectedFromPhoton		,
MatchmakingMode OnDisconnectedFromPhoton		
		,
LoadbalancingPeer.cs, 91 Public API, 33		
	LoadbalancingPeer.cs, 91	Public API, 33

OnFailedToConnectToPhoton	openField
Public API, 33	RoomInfo, 84
OnJoinedLobby	OperationCode, 46
Public API, 33	Authenticate, 47
OnJoinedRoom	ChangeGroups, 47
Public API, 33	CreateGame, 47
OnLeftLobby	GetProperties, 47
Public API, 33	JoinGame, 47
OnLeftRoom	JoinLobby, 47
Public API, 32	JoinRandomGame, 47
OnMasterClientSwitched	Leave, 47
Public API, 32	LeaveLobby, 47
OnPhotonCreateRoomFailed	RaiseEvent, 48
Public API, 32	SetProperties, 48
OnPhotonInstantiate	OperationNotAllowedInCurrentState
Public API, 33	ErrorCode, 39
OnPhotonJoinRoomFailed	Optional Gui Elements, 29
Public API, 33	otherPlayers
OnPhotonMaxCccuReached	PhotonNetwork, 70
Public API, 33	Others
OnPhotonPlayerConnected	Public API, 34
Public API, 33	OthersBuffered
	Public API, 34
OnPhotonPlayerDisconnected	owner
Public API, 33	PhotonView, 80
OnPhotonRandomJoinFailed	OwnerActorNr
Public API, 33	PhotonView, 80
OnPhotonSerializeView	ownerld
Public API, 33	PhotonView, 79
OnReceivedRoomListUpdate	Thotonview, 70
Public API, 33	ParameterCode, 48
OnApplicationQuit	ActorList, 49
PhotonView, 79	ActorNr, 49
OnDestroy	Add, 50
PhotonView, 79	Address, 50
OnGUI	AppVersion, 50
PhotonLagSimulationGui, 53	ApplicationId, 50
PhotonStatsGui, 75	AzureLocalNodeld, 50
OnSerializeRigidBody	AzureMasterNodeld, 50
PhotonView.cs, 93	AzureNodeInfo, 50
onSerializeRigidBodyOption	Broadcast, 50
PhotonView, 79	Cache, 50
OnSerializeTransform	CleanupCacheOnLeave, 50
PhotonView.cs, 93	Code, 50
onSerializeTransformOption	CustomEventContent, 51
PhotonView, 79	Data, 51
OnlyAngularVelocity	GameCount, 51
PhotonView.cs, 93	GameList, 51
OnlyPosition	GameProperties, 51
PhotonView.cs, 93	Group, 51
OnlyRotation	MasterPeerCount, 51
PhotonView.cs, 93	MatchMakingType, 51
OnlyScale	PeerCount, 51
PhotonView.cs, 93	PlayerProperties, 51
OnlyVelocity	Position, 51
PhotonView.cs, 93	Properties, 52
open	
•	ReceiverGroup, 52
Room, 82	ReceiverGroup, 52 Remove, 52
Room, 82 RoomInfo, 85	ReceiverGroup, 52 Remove, 52 RoomName, 52

Sac	cret, 52	photonView, 46
	getActorNr, 52	PhotonLagSimulationGui, 52
	erld, 52	OnGUI, 53
Peer	eriu, 32	Peer, 53
	otonLagSimulationGui, 53	Start, 53
PeerCre	-	Visible, 53
		Windowld, 53
	blic API, 31	
PeerCou		WindowRect, 53
	rameterCode, 51	PhotonLogLevel
PeerStat		Public API, 32
	olic API, 31	PhotonMessageInfo, 54
Photon,		PhotonMessageInfo, 54
Photon	Unity Networking/Plugins/PhotonNetwork/-	photonView, 54
Dhatas	CustomTypes.cs, 89	PhotonMessageInfo, 54
Photon	Unity Networking/Plugins/PhotonNetwork/-	sender, 54
Dhatas	Enums.cs, 89	timestamp, 54
Photon	Unity Networking/Plugins/PhotonNetwork/-	ToString, 54
ъ.	Extensions.cs, 90	PhotonNetwork, 55
Photon	Unity Networking/Plugins/PhotonNetwork/-	AllocateViewID, 59
<b>5</b>	LoadbalancingPeer.cs, 90	autoCleanUpPlayerObjects, 68
Photon	Unity Networking/Plugins/PhotonNetwork/-	autoJoinLobby, 69
	NetworkingPeer.cs, 91	automaticallySyncScene, 69
Photon	Unity Networking/Plugins/PhotonNetwork/-	CloseConnection, 59
	PhotonClasses.cs, 91	Connect, 59
Photon	Unity Networking/Plugins/PhotonNetwork/-	ConnectUsingSettings, 60
	PhotonHandler.cs, 92	connected, 69
Photon	Unity Networking/Plugins/PhotonNetwork/-	connectionState, 69
	PhotonLagSimulationGui.cs, 92	connectionStateDetailed, 69
Photon	Unity Networking/Plugins/PhotonNetwork/-	countOfPlayers, 69
	PhotonNetwork.cs, 92	countOfPlayersInRooms, 69
Photon	Unity Networking/Plugins/PhotonNetwork/-	countOfPlayersOnMaster, 69
	PhotonPlayer.cs, 93	countOfRooms, 70
Photon	Unity Networking/Plugins/PhotonNetwork/-	CreateRoom, 60, 61
	PhotonStatsGui.cs, 93	Destroy, 61
Photon	Unity Networking/Plugins/PhotonNetwork/-	DestroyPlayerObjects, 62
	PhotonView.cs, 93	Disconnect, 62
Photon	Unity Networking/Plugins/PhotonNetwork/-	GetPing, 62
	Room.cs, 94	GetRoomList, 62
Photon	Unity Networking/Plugins/PhotonNetwork/-	InitializeSecurity, 62
	RoomInfo.cs, 94	insideLobby, 70
Photon	Unity Networking/Plugins/PhotonNetwork/-	Instantiate, 62, 63
	ServerSettings.cs, 94	InstantiateSceneObject, 63
Photon.	MonoBehaviour, 46	InternalCleanPhotonMonoFromSceneIfStuck, 63
PhotonC	Cloud	isMasterClient, 70
Ser	verSettings, 86	isMessageQueueRunning, 70
PhotonV	liew.cs	isNonMasterClientInRoom, 70
All,	93, 94	JoinRandomRoom, 63, 64
Off,	, 94	JoinRoom, 64
Onl	yAngularVelocity, 93	LeaveRoom, 65
Onl	yPosition, 93	LoadLevel, 65
Onl	yRotation, 93	logLevel, 67
Onl	yScale, 93	MAX_VIEW_IDS, 67
OnlyVelocity, 93		masterClient, 70
Pos	sitionAndRotation, 94	maxConnections, 70
Rel	iableDeltaCompressed, 94	NetworkStatisticsEnabled, 70
Unreliable, 94		NetworkStatisticsReset, 65
Photon::MonoBehaviour		NetworkStatisticsToString, 65
networkView, 46		offlineMode, 70

otherPlayers, 70	Count, 77
player, 71	isReading, 77
playerList, 71	isWriting, 77
playerName, 71	PhotonStream, 77
precisionForFloatSynchronization, 68	PhotonStream, 77
precisionForQuaternionSynchronization, 68	ReceiveNext, 77
precisionForVectorSynchronization, 68	SendNext, 77
PrefabCache, 68	Serialize, 77
RemoveAllBufferedMessages, 65	ToArray, 77
RemoveAllInstantiatedObjects, 65	PhotonTargets
RemoveRPCs, 66	Public API, 33
RemoveRPCsInGroup, 66	PhotonView, 78
room, 71	Awake, 79
SendOutgoingCommands, 66	Find, 79
sendRate, 71	Get, 79
sendRateOnSerialize, 71	
	group, 79
serverSettingsAssetFile, 68	instantiationData, 80
serverSettingsAssetPath, 68	instantiationId, 79
SetLevelPrefix, 66	isMine, 80
SetPlayerCustomProperties, 67	isSceneView, 80
SetReceivingEnabled, 67	observed, 79
SetSendingEnabled, 67	OnApplicationQuit, 79
time, 71	OnDestroy, 79
UnAllocateViewID, 67	onSerializeRigidBodyOption, 79
unreliableCommandsLimit, 71	onSerializeTransformOption, 79
UsePrefabCache, 68	owner, 80
versionPUN, 68	OwnerActorNr, 80
PhotonNetwork.cs	ownerld, 79
Object, 93	prefix, 80
PhotonNetworkingMessage	prefixBackup, 79
Public API, 32	RPC, 79
PhotonPlayer, 72	subld, 79
allProperties, 74	synchronization, 79
customProperties, 74	ToString, 79
Equals, 73	viewID, 80
Find, 73	photonView
GetHashCode, 73	Photon::MonoBehaviour, 46
ID, 74	PhotonMessageInfo, 54
isLocal, 74	PhotonView.cs
isMasterClient, 74	OnSerializeRigidBody, 93
name, 74	OnSerializeTransform, 93
PhotonPlayer, 73	ViewSynchronization, 94
PhotonPlayer, 73	player
SetCustomProperties, 73	PhotonNetwork, 71
ToString, 73	PlayerCount
PhotonStatsGui, 74	GameProperties, 45
buttonsOn, 75	playerCount
healthStatsVisible, 75	Room, 82
OnGUI, 75	RoomInfo, 85
Start, 75	playerList
statsOn, 76	PhotonNetwork, 71
statsRect, 76	PlayerName
statsWindowOn, 76	ActorProperties, 37
trafficStatsOn, 76	playerName
TrafficStatsWindow, 75	PhotonNetwork, 71
Update, 75	PlayerProperties
Windowld, 76	ParameterCode, 51
PhotonStream, 76	Position

ParameterCode, 51	OnDisconnectedFromPhoton, 33
PositionAndRotation	OnFailedToConnectToPhoton, 33
PhotonView.cs, 94	OnJoinedLobby, 33
precisionForFloatSynchronization	OnJoinedRoom, 33
PhotonNetwork, 68	OnLeftLobby, 33
precisionForQuaternionSynchronization	OnLeftRoom, 32
PhotonNetwork, 68	OnMasterClientSwitched, 32
precisionForVectorSynchronization	OnPhotonCreateRoomFailed, 32
PhotonNetwork, 68	OnPhotonInstantiate, 33
PrefabCache	OnPhotonJoinRoomFailed, 33
PhotonNetwork, 68	OnPhotonMaxCccuReached, 33
prefix	OnPhotonPlayerConnected, 33
PhotonView, 80	OnPhotonPlayerDisconnected, 33
prefixBackup	OnPhotonRandomJoinFailed, 33
PhotonView, 79	OnPhotonSerializeView, 33
Properties	OnReceivedRoomListUpdate, 33
ParameterCode, 52	Others, 34
PropertiesChanged	OthersBuffered, 34
EventCode, 41	PeerCreated, 31
propertiesListedInLobby	Queued, 31
Room, 82	QueuedComingFromGameserver, 32
	TimeoutDisconnect, 31
PropsListedInLobby	Uninitialized, 31
GameProperties, 45	Public API, 30
Public API	DisconnectCause, 31
All, 34	PeerState, 31
AllBuffered, 34	PhotonLogLevel, 32
Authenticated, 32	PhotonNetworkingMessage, 32
Connected, 31	PhotonTargets, 33
ConnectedComingFromGameserver, 32	i notorrargets, 33
ConnectedToGameserver, 32	QueueState
ConnectedToMaster, 32	EventCode, 41
Connecting, 31	Queued
ConnectingToGameserver, 32	Public API, 31
ConnectingToMasterserver, 32	QueuedComingFromGameserver
DisconnectByServer, 31	Public API, 32
DisconnectByServerLogic, 31	I dollo Al I, 02
DisconnectByServerUserLimit, 31	RPC
Disconnected, 32	PhotonView, 79
Disconnecting, 32	RaiseEvent
DisconnectingFromGameserver, 32	OperationCode, 48
DisconnectingFromMasterserver, 32	RandomMatching
ErrorsOnly, 32	LoadbalancingPeer.cs, 91
Exception, 31	ReceiveNext
ExceptionOnConnect, 31	PhotonStream, 77
Full, 32	ReceiverGroup
Informational, 32	•
InternalReceiveException, 31	ParameterCode, 52
InvalidRegion, 31	Reliable Delta Compressed
Joined, 32	PhotonView.cs, 94
	Remove
JoinedLobby, 32	ParameterCode, 52
Joining, 32	RemoveAllBufferedMessages
Leaving, 32	PhotonNetwork, 65
MasterClient, 34	RemoveAllInstantiatedObjects
MaxCcuReached, 31	PhotonNetwork, 65
OnConnectedToMaster, 33	RemoveRPCs
OnConnectedToPhoton, 32	PhotonNetwork, 66
OnConnectionFail, 33	RemoveRPCsInGroup
OnCreatedRoom, 33	PhotonNetwork, 66

Removed	ServerSettings, 87
GameProperties, 45	ServerFull
removedFromList	ErrorCode, 40
RoomInfo, 85	ServerPort
Room, 80	ServerSettings, 87
autoCleanUp, 81	ServerSettings, 85
maxPlayers, 81	AppID, 87
name, 81	CloudServerRegionNames, 87
open, <mark>82</mark>	CloudServerRegionPrefixes, 87
playerCount, 82	DefaultAppID, 87
propertiesListedInLobby, 82	DefaultCloudServerUrl, 87
SetCustomProperties, 81	DefaultMasterPort, 87
visible, 82	DefaultServerAddress, 87
room	DisableAutoOpenWizard, 87
PhotonNetwork, 71	FindRegionForServerAddress, 87
RoomInfo, 82	FindServerAddressForRegion, 87
autoCleanUpField, 84	HostType, 87
customProperties, 84	HostingOption, 86
Equals, 83	<b>5</b> ,
GetHashCode, 83	ServerAddress, 87
isLocalClientInside, 84	ServerPort, 87
maxPlayers, 84	ToString, 87
maxPlayersField, 84	UseCloud, 87
name, 85	UseMyServer, 87
nameField, 84	serverSettingsAssetFile
	PhotonNetwork, 68
open, 85 openField, 84	serverSettingsAssetPath
•	PhotonNetwork, 68
playerCount, 85	SetCustomProperties
removedFromList, 85	PhotonPlayer, 73
ToString, 84	Room, 81
visible, 85	SetLevelPrefix
visibleField, 84 RoomName	PhotonNetwork, 66
	SetPlayerCustomProperties
ParameterCode, 52	PhotonNetwork, 67
Secret	SetProperties
ParameterCode, 52	EventCode, 41
SelfHosted	OperationCode, 48
ServerSettings, 86	SetReceivingEnabled
SendNext	PhotonNetwork, 67
PhotonStream, 77	SetSendingEnabled
SendOutgoingCommands	PhotonNetwork, 67
PhotonNetwork, 66	Start
sendRate	PhotonLagSimulationGui, 53
PhotonNetwork, 71	PhotonStatsGui, 75
sendRateOnSerialize	statsOn
PhotonNetwork, 71	PhotonStatsGui, 76
sender	statsRect
PhotonMessageInfo, 54	PhotonStatsGui, 76
SerialMatching	statsWindowOn
LoadbalancingPeer.cs, 91	PhotonStatsGui, 76
Serialize	StripKeysWithNullValues
PhotonStream, 77	Extensions, 43
ServerSettings	StripToStringKeys
NotSet, 86	Extensions, 44
OfflineMode, 86	subId
PhotonCloud, 86	PhotonView, 79
SelfHosted, 86	synchronization
ServerAddress	PhotonView, 79
001 V01/ WW1000	i Hotorry IGVV, 7 3

TargetActorNr PhotonLagSimulationGui, 53 ParameterCode, 52 PhotonStatsGui, 76 WindowRect time PhotonNetwork, 71 PhotonLagSimulationGui, 53 TimeoutDisconnect Public API, 31 timestamp PhotonMessageInfo, 54 **ToArray** PhotonStream, 77 **ToString** PhotonMessageInfo, 54 PhotonPlayer, 73 PhotonView, 79 RoomInfo, 84 ServerSettings, 87 ToStringFull Extensions, 44 trafficStatsOn PhotonStatsGui, 76 TrafficStatsWindow PhotonStatsGui, 75 UnAllocateViewID PhotonNetwork, 67 Uninitialized Public API, 31 Unreliable PhotonView.cs, 94 unreliableCommandsLimit PhotonNetwork, 71 Update PhotonStatsGui, 75 UseCloud ServerSettings, 87 UseMyServer ServerSettings, 87 UsePrefabCache PhotonNetwork, 68 UserBlocked ErrorCode, 40 Userld ParameterCode, 52 versionPUN PhotonNetwork, 68 viewID PhotonView, 80 ViewSynchronization PhotonView.cs, 94 Visible PhotonLagSimulationGui, 53 visible Room, 82 RoomInfo, 85 visibleField RoomInfo, 84

Windowld