

Combu

3.0.15

Generated by Doxygen 1.8.14

Contents

- 1 API Reference** **1**
 - 1.1 Introduction 1
 - 1.2 Installation 1
 - 1.3 Documentation 2
 - 1.4 Customers from Asset Store 2

- 2 Off-line documentation** **3**

- 3 Web server setup** **5**
 - 3.1 Live server 5
 - 3.2 Local machine 5
 - 3.3 Server and Client configuration 6

- 4 Server and Client configuration** **7**
 - 4.1 Server configuration 7
 - 4.2 Client configuration 9

- 5 Email configuration and Reset Password** **11**
 - 5.1 Server configuration 11

- 6 Server-Client Security** **13**
 - 6.1 RSA + AES keys handshaking 13
 - 6.2 Data encryption 13

- 7 Error messages localization** **15**
 - 7.1 Client request 15
 - 7.2 Server response 15

8 Authentication and Users	17
8.1 Authentication	17
8.1.1 Auto-login	17
8.2 Registration	17
8.2.1 Send confirmation by email	18
8.2.2 Create a guest account	19
8.3 Custom Data	19
8.4 Load users	19
8.5 Online state	20
8.6 Create your User class	20
9 Server Settings	23
9.1 Server	23
9.2 Client	23
10 Linking external platforms	25
10.1 Authentication	25
10.2 Link a platform to the local user	25
10.3 Transfer the external platforms	25
10.4 Load users from platforms	26
11 Managing Contacts	27
11.1 Loading Contacts	27
11.2 Adding Contacts	27
11.3 Removing Contacts	28
12 Managing Files	29
12.1 Loading Files	29
12.2 Adding Files	29
12.3 Viewing Files	30
12.4 Liking Files	30
12.5 Removing Files	30

13 Managing Inventory	31
13.1 Loading Inventory	31
13.2 Adding and Editing Items	31
13.3 Removing Items	32
14 Managing Messages	33
14.1 Loading Messages	33
14.2 Sending Messages	33
14.2.1 Sending to UserGroup	34
14.3 Marking Messages as Read	34
14.4 Marking Messages as Unread	34
14.5 Removing Items	35
14.6 Loading Conversations	35
14.7 Counting Messages	35
15 Managing User Groups	37
15.1 Create a new group	37
15.2 Load groups	37
15.3 Join a group	37
15.4 Leave a group	38
16 Managing News	39
16.1 Loading News	39
17 Managing Leaderboards	41
17.1 Loading Scores	41
17.2 Reporting Scores	41
17.3 Loading Leaderboards data	42
17.4 Loading the score of a user	42
18 Managing Achievements	43
18.1 Loading Achievements descriptions	43
18.2 Reporting Progress	43

19 Managing Tournaments	45
19.1 Loading Tournaments	45
19.1.1 Loading by Tournament ID	45
19.1.2 Matches of the Tournament	45
19.2 Quick Tournament	46
19.2.1 Create your own Tournament	46
19.3 Removing Tournaments	46
20 Managing Matches	47
20.1 Loading Matches	47
20.1.1 Loading by Match ID	47
20.1.2 Rounds of the Match	47
20.2 Quick Match	47
20.2.1 Create your own Match	48
20.3 Sending Score	48
20.4 Removing Matches	48
21 Customize Web Administration	49
21.1 Javascript	49
21.2 CSS	49
22 Frequently Asked Questions	51
22.1 How do I install Combu on my local/live server?	51
22.2 How can I add my own properties to accounts?	51
22.3 I purchased it on Asset Store, can I download from your website?	52
22.4 When I try to navigate to the web admin, I see a blank page	52
23 Namespace Index	53
23.1 Packages	53
24 Hierarchical Index	55
24.1 Class Hierarchy	55

25 Class Index	57
25.1 Class List	57
26 Namespace Documentation	59
26.1 Combu Namespace Reference	59
26.1.1 Detailed Description	60
26.1.2 Enumeration Type Documentation	60
26.1.2.1 eContactType	60
26.1.2.2 eLeaderboardInterval	60
26.1.2.3 eLeaderboardTimeScope	61
26.1.2.4 eMailList	61
26.1.2.5 eSearchOperator	61
27 Class Documentation	63
27.1 Combu.Achievement Class Reference	63
27.1.1 Detailed Description	64
27.2 Combu.CombuEncryption Class Reference	64
27.2.1 Member Function Documentation	64
27.2.1.1 DecryptAES()	64
27.2.1.2 DecryptResponse()	65
27.2.1.3 EncryptAES()	65
27.2.1.4 EncryptMD5()	65
27.2.1.5 EncryptRSA()	66
27.2.1.6 EncryptSHA1()	66
27.2.1.7 LoadRSA() [1/2]	66
27.2.1.8 LoadRSA() [2/2]	67
27.2.1.9 SetToken()	67
27.3 Combu.CombuForm Class Reference	67
27.3.1 Member Function Documentation	68
27.3.1.1 AddBinaryData()	68
27.3.1.2 AddField()	68

27.3.1.3	GetBinaryField()	69
27.3.1.4	GetField()	69
27.3.1.5	GetForm()	69
27.4	Combu.CombuManager Class Reference	70
27.4.1	Detailed Description	72
27.4.2	Member Function Documentation	72
27.4.2.1	AutoPing()	73
27.4.2.2	CallWebservice()	73
27.4.2.3	CancelRequest()	73
27.4.2.4	CaptureScreenshot()	73
27.4.2.5	CreateForm()	74
27.4.2.6	DecryptAES()	74
27.4.2.7	DownloadUrl()	74
27.4.2.8	EncryptAES()	75
27.4.2.9	EncryptMD5()	75
27.4.2.10	EncryptSHA1()	75
27.4.2.11	GetServerInfo()	76
27.4.2.12	GetUrl()	76
27.4.2.13	Ping()	76
27.4.2.14	SetLocalUser()	77
27.4.3	Member Data Documentation	77
27.4.3.1	_instance	77
27.4.3.2	_platform	77
27.4.3.3	achievementUIFunction	77
27.4.3.4	achievementUIObject	77
27.4.3.5	appld	78
27.4.3.6	appSecret	78
27.4.3.7	COMBU_VERSION	78
27.4.3.8	dontDestroyOnLoad	78
27.4.3.9	encryption	78

27.4.3.10 language	78
27.4.3.11 leaderboardUIFunction	79
27.4.3.12 leaderboardUIObject	79
27.4.3.13 logDebugInfo	79
27.4.3.14 onlineSeconds	79
27.4.3.15 pingIntervalSeconds	79
27.4.3.16 playingSeconds	79
27.4.3.17 rememberCredentials	80
27.4.3.18 setAsDefaultSocialPlatform	80
27.4.3.19 urlRootProduction	80
27.4.3.20 urlRootStage	80
27.4.3.21 useStage	80
27.4.4 Property Documentation	80
27.4.4.1 defaultSocialPlatform	80
27.4.4.2 instance	81
27.4.4.3 isAuthenticated	81
27.4.4.4 isCancelling	81
27.4.4.5 isDownloading	81
27.4.4.6 isInitialized	81
27.4.4.7 localUser	81
27.4.4.8 platform	82
27.4.4.9 serverInfo	82
27.4.4.10 sessionToken	82
27.5 Combu.CombuPlatform Class Reference	82
27.5.1 Detailed Description	84
27.5.2 Member Function Documentation	84
27.5.2.1 Authenticate() [1/3]	84
27.5.2.2 Authenticate() [2/3]	85
27.5.2.3 Authenticate() [3/3]	85
27.5.2.4 Authenticate< T >()	85

27.5.2.5	CreateAchievement()	86
27.5.2.6	CreateLeaderboard()	86
27.5.2.7	GetLoading()	87
27.5.2.8	LoadAchievementDescriptions()	87
27.5.2.9	LoadAchievements()	87
27.5.2.10	LoadAchievements< T >()	88
27.5.2.11	LoadFriends()	88
27.5.2.12	LoadScores() [1/4]	89
27.5.2.13	LoadScores() [2/4]	89
27.5.2.14	LoadScores() [3/4]	90
27.5.2.15	LoadScores() [4/4]	90
27.5.2.16	LoadScoresByUser()	91
27.5.2.17	LoadUsers()	91
27.5.2.18	Logout()	92
27.5.2.19	ReportProgress() [1/2]	92
27.5.2.20	ReportProgress() [2/2]	92
27.5.2.21	ReportScore() [1/3]	93
27.5.2.22	ReportScore() [2/3]	93
27.5.2.23	ReportScore() [3/3]	94
27.5.2.24	ResetAllAchievements()	94
27.5.2.25	SetLocalUser()	94
27.5.2.26	ShowAchievementsUI()	95
27.5.2.27	ShowLeaderboardUI()	95
27.6	Combu.CombuServerInfo Class Reference	95
27.6.1	Detailed Description	96
27.6.2	Member Function Documentation	96
27.6.2.1	ToString()	96
27.7	Combu.Inventory Class Reference	96
27.7.1	Constructor & Destructor Documentation	97
27.7.1.1	Inventory() [1/3]	97

27.7.1.2	Inventory() [2/3]	97
27.7.1.3	Inventory() [3/3]	97
27.7.2	Member Function Documentation	98
27.7.2.1	Delete() [1/2]	98
27.7.2.2	Delete() [2/2]	98
27.7.2.3	FromHashtable()	98
27.7.2.4	FromJson()	98
27.7.2.5	Load()	99
27.7.2.6	Load< T >()	99
27.7.2.7	Update()	100
27.8	Combu.Leaderboard Class Reference	100
27.8.1	Member Function Documentation	101
27.8.1.1	FromJson()	101
27.8.1.2	Load()	101
27.8.1.3	Load< T >()	102
27.8.1.4	LoadScores()	102
27.8.1.5	LoadScoresByUser() [1/5]	103
27.8.1.6	LoadScoresByUser() [2/5]	103
27.8.1.7	LoadScoresByUser() [3/5]	103
27.8.1.8	LoadScoresByUser() [4/5]	104
27.8.1.9	LoadScoresByUser() [5/5]	104
27.8.1.10	SetUserFilter()	104
27.9	Combu.Mail Class Reference	105
27.9.1	Member Function Documentation	106
27.9.1.1	Count()	106
27.9.1.2	Delete() [1/2]	106
27.9.1.3	Delete() [2/2]	106
27.9.1.4	FromHashtable()	107
27.9.1.5	FromJson()	107
27.9.1.6	Load< T >()	107

27.9.1.7 LoadConversations()	108
27.9.1.8 Read() [1/4]	108
27.9.1.9 Read() [2/4]	108
27.9.1.10 Read() [3/4]	109
27.9.1.11 Read() [4/4]	109
27.9.1.12 Send() [1/5]	109
27.9.1.13 Send() [2/5]	110
27.9.1.14 Send() [3/5]	110
27.9.1.15 Send() [4/5]	110
27.9.1.16 Send() [5/5]	111
27.9.1.17 SendMailToGroup()	111
27.9.1.18 Unread() [1/2]	112
27.9.1.19 Unread() [2/2]	112
27.10Combu.MailCount Class Reference	112
27.11Combu.Match Class Reference	113
27.11.1 Constructor & Destructor Documentation	114
27.11.1.1 Match() [1/2]	114
27.11.1.2 Match() [2/2]	114
27.11.2 Member Function Documentation	114
27.11.2.1 AddUser()	115
27.11.2.2 Delete() [1/2]	115
27.11.2.3 Delete() [2/2]	115
27.11.2.4 FromHashtable()	115
27.11.2.5 FromJson()	116
27.11.2.6 Load() [1/2]	116
27.11.2.7 Load() [2/2]	116
27.11.2.8 QuickMatch()	117
27.11.2.9 RemoveUser() [1/4]	117
27.11.2.10RemoveUser() [2/4]	117
27.11.2.11RemoveUser() [3/4]	117

27.11.2.12	RemoveUser() [4/4]	118
27.11.2.13	Save()	118
27.11.2.14	Score()	118
27.12	Combu.MatchAccount Class Reference	119
27.12.1	Member Function Documentation	119
27.12.1.1	FromHashtable()	119
27.12.1.2	FromJson()	119
27.13	Combu.MatchRound Class Reference	121
27.13.1	Member Function Documentation	121
27.13.1.1	FromHashtable()	121
27.13.1.2	FromJson()	122
27.14	Combu.Match.MatchRoundData Class Reference	122
27.15	Combu.MiniJSON Class Reference	122
27.15.1	Member Function Documentation	123
27.15.1.1	getLastErrorIndex()	123
27.15.1.2	getLastErrorSnippet()	123
27.15.1.3	jsonDecode()	123
27.15.1.4	jsonEncode()	124
27.15.1.5	lastDecodeSuccessful()	124
27.15.2	Member Data Documentation	124
27.15.2.1	lastErrorIndex	124
27.16	Combu.News Class Reference	125
27.16.1	Member Function Documentation	125
27.16.1.1	FromHashtable()	125
27.16.1.2	FromJson()	125
27.16.1.3	Load()	126
27.16.1.4	Load< T >()	126
27.17	Combu.Profile Class Reference	127
27.17.1	Constructor & Destructor Documentation	128
27.17.1.1	Profile() [1/2]	128

27.17.1.2 Profile() [2/2]	128
27.17.2 Member Function Documentation	129
27.17.2.1 FromHashtable()	129
27.17.2.2 FromJson()	129
27.17.3 Member Data Documentation	129
27.17.3.1 appCustomData	129
27.17.3.2 customData	129
27.17.3.3 email	130
27.17.4 Property Documentation	130
27.17.4.1 id	130
27.17.4.2 idLong	130
27.17.4.3 image	130
27.17.4.4 isFriend	130
27.17.4.5 lastSeen	130
27.17.4.6 sessionToken	131
27.17.4.7 state	131
27.17.4.8 userName	131
27.18Combu.ProfilePlatform Class Reference	131
27.19Combu.Score Class Reference	131
27.19.1 Member Function Documentation	132
27.19.1.1 Initialize() [1/2]	132
27.19.1.2 Initialize() [2/2]	133
27.19.1.3 ReportScore()	133
27.20Combu.SearchCustomData Class Reference	133
27.20.1 Detailed Description	134
27.21Combu.Tournament Class Reference	134
27.21.1 Detailed Description	135
27.21.2 Constructor & Destructor Documentation	135
27.21.2.1 Tournament() [1/2]	135
27.21.2.2 Tournament() [2/2]	135

27.21.3 Member Function Documentation	135
27.21.3.1 Delete() [1/2]	135
27.21.3.2 Delete() [2/2]	136
27.21.3.3 FromHashtable()	136
27.21.3.4 FromJson()	136
27.21.3.5 Load() [1/2]	136
27.21.3.6 Load() [2/2]	137
27.21.3.7 QuickTournament()	137
27.21.3.8 Save()	138
27.22Combu.User Class Reference	138
27.22.1 Detailed Description	140
27.22.2 Member Function Documentation	140
27.22.2.1 AddContact() [1/3]	141
27.22.2.2 AddContact() [2/3]	141
27.22.2.3 AddContact() [3/3]	141
27.22.2.4 Authenticate() [1/3]	142
27.22.2.5 Authenticate() [2/3]	142
27.22.2.6 Authenticate() [3/3]	142
27.22.2.7 AuthenticatePlatform()	142
27.22.2.8 AutoLogin()	143
27.22.2.9 CanAutoLogin()	143
27.22.2.10ChangePassword() [1/2]	143
27.22.2.11ChangePassword() [2/2]	144
27.22.2.12CreateGuest()	144
27.22.2.13Delete() [1/2]	144
27.22.2.14Delete() [2/2]	145
27.22.2.15Exists()	145
27.22.2.16FromUser()	145
27.22.2.17GetContact()	146
27.22.2.18GetContact< T >()	146

27.22.2.19	LinkAccount()	146
27.22.2.20	LinkPlatform()	148
27.22.2.21	Load() [1/6]	148
27.22.2.22	Load() [2/6]	148
27.22.2.23	Load() [3/6]	149
27.22.2.24	Load() [4/6]	149
27.22.2.25	Load() [5/6]	149
27.22.2.26	Load() [6/6]	150
27.22.2.27	Load< T >()	150
27.22.2.28	LoadFriends() [1/2]	150
27.22.2.29	LoadFriends() [2/2]	152
27.22.2.30	LoadFriends< T >()	152
27.22.2.31	Random< T >()	153
27.22.2.32	RemoveContact() [1/3]	153
27.22.2.33	RemoveContact() [2/3]	153
27.22.2.34	RemoveContact() [3/3]	154
27.22.2.35	ResetPassword() [1/3]	154
27.22.2.36	ResetPassword() [2/3]	154
27.22.2.37	ResetPassword() [3/3]	154
27.22.2.38	Update()	155
27.23	Combu.UserFile Class Reference	155
27.23.1	Constructor & Destructor Documentation	156
27.23.1.1	UserFile() [1/3]	156
27.23.1.2	UserFile() [2/3]	156
27.23.1.3	UserFile() [3/3]	157
27.23.2	Member Function Documentation	157
27.23.2.1	Delete() [1/2]	157
27.23.2.2	Delete() [2/2]	157
27.23.2.3	Download()	158
27.23.2.4	FromHashtable()	158

27.23.2.5 FromJson()	158
27.23.2.6 Like() [1/3]	158
27.23.2.7 Like() [2/3]	159
27.23.2.8 Like() [3/3]	159
27.23.2.9 Load()	159
27.23.2.10Load< T >()	160
27.23.2.11Update()	160
27.23.2.12View() [1/3]	161
27.23.2.13View() [2/3]	161
27.23.2.14View() [3/3]	161
27.24Combu.UserGroup Class Reference	162
27.24.1 Constructor & Destructor Documentation	163
27.24.1.1 UserGroup() [1/3]	163
27.24.1.2 UserGroup() [2/3]	163
27.24.1.3 UserGroup() [3/3]	163
27.24.2 Member Function Documentation	163
27.24.2.1 Delete()	164
27.24.2.2 FromHashtable()	164
27.24.2.3 FromJson()	164
27.24.2.4 Join() [1/3]	164
27.24.2.5 Join() [2/3]	165
27.24.2.6 Join() [3/3]	165
27.24.2.7 Join< T >()	165
27.24.2.8 Leave() [1/3]	166
27.24.2.9 Leave() [2/3]	166
27.24.2.10Leave() [3/3]	166
27.24.2.11Leave< T >()	167
27.24.2.12Save()	167
Index	169

Chapter 1

API Reference

1.1 Introduction

Combu is a full featured solution to add online storage management for your player login system, highscores, friends, inventory and more for your games using a web server and MySQL database. It is shipped with client source code for Unity3D.

This documentation is for **version 3.x** only. The documentation for **version 2.x** is located at [here](#).

The 3rd generation of **Combu** features huge improvement to security: now every call from client to server is encrypted through AES keys generated on the fly at the beginning of every launch and exchanged with the server with RSA key (also generated on the fly for each new connection).

1.2 Installation

1. Purchase **Combu**

You can purchase **Combu** from **Skared Creations website** or Unity3D **Asset Store**.

2. Import **Combu** package

Create an empty scene and import the **Combu** unitypackage (or from Asset Store window).

3. Setup the web server

[Click here](#) to see how to configure your web server, you can use a local web server before going live to production for example **XAMPP** and extract the file **combu_web.zip** (you find it in the folder *Install* of **Combu** in Unity) in the root of the web server (or in any other folder of your choice inside the web server tree).

4. Navigate to **/_setup**

You can use your web browser to navigate to http://yourserver/combu_folder_path/_setup to generate the content for your configuration file (located at */lib/config.php*) and the SQL queries to create the tables on your database.

5. Configure **Combu** Manager

Drag the prefab *Combu/Prefabs/CombuManager* in the scene and set the correct settings in the **Combu**↔**Manager** component inspector in order to connect to your web server. To get started you can just set the properties **Url Root Production** to the URL of your live production environment and **Url Root Stage** to the URL of your development environment (usually your local machine), then select the checkbox **Use Stage** to run the development environment or leave it unchecked to run the live production.

The first time you will access the admin console (at: http://yourserver/combu_folder_path/admin) the system will create a new admin account with username and password **admin** and you should be automatically logged. As soon as you access the admin console for the first time on your live server you're strongly suggested to change the password of user **admin** in the section **Admins** or delete it and create a new user with a different username and password.

To start now please visit the welcome page [Getting started](#) or go through the [Related Pages](#) and [API reference](#) in this documentation.

1.3 Documentation

You can download the API documentation as offline [PDF](#) or navigate online at <https://www.skaredcreations.com/api/combu>

You can also use [Doxygen](#) to build the HTML/PDF documentation directly from the Unity scripts of [Combu](#).

1.4 Customers from Asset Store

If you purchased [Combu](#) on **Unity Asset Store**, now you can redeem the download on this website too at no cost by providing the Invoice No. [Click here](#) to redeem your invoice now.

Chapter 2

Off-line documentation

Download this documentation as [PDF](#)

Chapter 3

Web server setup

In this section you will learn how to setup a web server on your local machine and install [Combu](#).

3.1 Live server

[Combu](#) will work correctly in almost every hosting provider, since the production servers are usually configured correctly. So in production environment you only need to upload all files including the folder `_setup`, navigate with your browser to http://yourserver/yourcombu/_setup and complete both the configuration file section and the database tables creation.

You can skip the configuration file section if you already configured it locally, in this case you will edit `lib/config.php` for the database connection and root URL defines (`URL_ROOT`, `GAME_DB_SERVER`, `GAME_DB_NAME`, `GAME_DB_USER` and `GAME_DB_PASS`), as explained in the [tutorial](#) video.

After the configuration file has been eventually edited and the database tables are created, you should delete the folder `_setup` from your production server.

3.2 Local machine

Before using a live server you may want to test your game earlier and work locally on your machine, in this case you have few alternatives (Microsoft IIS, Apache, XAMPP, etc) and what's to get only depends on your expertise and skills.

1. For the sake of this sample we will assume you have installed [XAMPP](#), it's a well-known free package that contains both the web server Apache and the MySQL server engine (it exists few similar packages, like LAMPP, they're almost the same for this example)
2. **MySQL** must be configured with case-sensitivity for table names, it's usually already configured on live servers by hosting providers; if you're running on your local **Windows** machine then edit the file `my.ini` (you'll find it in one of the folders inside XAMPP installation) and add a new line with `lower_case_table_names=2` just below the line that contains `[mysqld]` (if you haven't the section `[mysqld]` then add both lines at the end of the file)
3. Start both HTTP and MySQL services (if you're using XAMPP, it can be done through the XAMPP Control Panel else from Windows Services)
4. Create an empty database with name **combu** (or any other name of your choice) in phpMyAdmin (in XAMPP it's usually installed at <http://localhost/phpmyadmin>) or [MySQL Workbench](#)
5. Uncompress the file `combu_web.zip` into your local web server root (if you're using XAMPP then it's usually located in the folder `xamppfiles\htdocs`) and use your web browser to navigate to http://yourserver/combu_folder_path/_setup (for example http://localhost/combu/_setup) to generate the content for your configuration file (located at `lib/config.php`) and the SQL queries to create the tables on your database.

3.3 Server and Client configuration

Now that you learned how to setup your web server and have installed **Combu** server, you can configure the system at your needs.

Continue to [Server and Client configuration](#).

Chapter 4

Server and Client configuration

In this section you will learn how to setup your server and client.

4.1 Server configuration

We really suggest to navigate to http://yourserver/combu_folder_path/_setup of your development environment to get the content for your configuration file, because it's maintained up to date with all the settings that can be configured with detailed descriptions.

The configuration file is located at **lib/config.php** and contains the PHP defines of every configuration setting (if you are new to PHP, "define" is the equivalent of "const" in C#):

- Generic
 - **URL_ROOT** (default: /combu/): Define the relative URL of **Combu** folder, including the last path separator (e.g.: /your_combu_path/)
 - **WS_DEBUG** (default: FALSE): Debug the web services data: if TRUE then the data is loaded from both GET and POST, else it is loaded only from POST (set to FALSE or comment in the production environment)
 - **DEFAULT_LIST_LIMIT** (default: 20): Default size of a records list when limit is not specified in the REQUEST (must be greater than zero)
 - **RANDOM_CODE_CHARS** (default: bcdghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTUVWXYZ0123456789)↔ : Alphanumeric characters used to generate a random code (e.g.: Activation Code and Reset Password Authorization Code)
 - **RANDOM_CODE_LENGTH** (default: 10): Default length of a random code (e.g.: Activation Code and Reset Password Authorization Code)
- Database connection
 - **GAME_DB_USE_PDO** (default: TRUE): Should use PDO for the database functions? (If FALSE then it uses mysqli)
 - **GAME_DB_TYPE** (default: mysql): Database engine (only if PDO is enabled, this must be supported by PDO)
 - **GAME_DB_SERVER**: Database Server hostname or IP (if the engine is running on the same machine use 'localhost')
 - **GAME_DB_PORT** (default: 3306): The port of the database connection (default MySQL port is 3306)
 - **GAME_DB_NAME**: The name of the database

- **GAME_DB_PREFIX**: Prefix for table names
- **GAME_DB_USER**: The username of the database connection
- **GAME_DB_PASS**: The password of the database connection
- Security
 - **RSA_PRIVATE_KEY_BITS** (default: 2048): Key bits length for RSA generation (higher values increase security but require more time to generate keys), can be 1024, 2048 or 4096
 - **RESPONSE_ENCRYPTED** (default: TRUE): Should encrypt the server response to web services? (TRUE increase security but requires more resources/time both in server and in client)
 - **DENY_UNVERSIONED_CLIENT** (default: FALSE): Should refuse the connections when the clients don't send their version number?
 - **MIN_CLIENT_VERSION** (default: empty): Minimum version of client's CombuManager.COMBU_VERSION to be considered as out of range
 - **MAX_CLIENT_VERSION** (default: empty): Maximum version of client's CombuManager.COMBU_VERSION to be considered as out of range
 - **DENY_OUTRANGE_VERSIONS** (default: FALSE): Should refuse the connections with out of range versions?
- Email
 - **EMAIL_SMTP** (default: FALSE): it allows to send the emails through SMTP server
 - **EMAIL_SMTP_SECURE** (default: NULL): secure authentication type for the SMTP connection (can be: NULL, 'ssl' or 'tls')
 - **EMAIL_SMTP_HOSTNAME** (default: NULL): server hostname or IP for the SMTP connection
 - **EMAIL_SMTP_PORT** (default: NULL): server port for the SMTP connection (e.g. 465)
 - **EMAIL_SMTP_USERNAME** (default: NULL): username for the SMTP connection
 - **EMAIL_SMTP_PASSWORD** (default: NULL): password for the SMTP connection
 - **EMAIL_SENDER_ADDRESS** (default: `noreply@yourserver.com`): the sender address of outgoing mail
 - **EMAIL_SENDER_NAME** (default: Your App Title): the sender name of outgoing mail
- Email
 - **NEWSLETTER_SENDER_ADDRESS** (default: `noreply@yourserver.com`): the sender address of outgoing newsletters
 - **NEWSLETTER_SENDER_NAME** (default: Your Newsletter): the sender name of outgoing newsletters
- User Registration
 - **REGISTER_EMAIL_REQUIRED** (default: FALSE): it will require a valid email address upon new user creation
 - **REGISTER_EMAIL_MULTIPLE** (default: FALSE): it allows to use the same email address for multiple accounts
 - **REGISTER_EMAIL_ACTIVATION** (default: FALSE): it will create an activation code during user creation and sends it by email, then will require to activate the account before being able to login
 - **REGISTER_EMAIL_SUBJECT** (default: Confirm account registration): the subject text of the user registration email
 - **REGISTER_EMAIL_MESSAGE** (default: email_register.html): Name of the HTML/text file containing the mail message (the file must exist in /email_templates); the HTML/text file can contain the following special words:
 - * **{ACTIVATION_URL}**: it will be replaced with the URL to activate the account (if **REGISTER_EMAIL_ACTIVATION** is *TRUE*)
 - * **{USERNAME}**: it will be replaced with the chosen username
 - **REGISTER_EMAIL_HTML**: it establishes if the user registration email is in HTML or text

- **RESETPWD_EMAIL_SUBJECT** (default: Reset your password): the reset-password mail subject
- **RESETPWD_EMAIL_MESSAGE** (default: email_resetpwd.html): Name of the HTML/text file containing the mail message (the file must exist in /email_templates); the HTML/text file can contain the following special words:
 - * **{CODE}**: it will be replaced with the code to pass to User.ChangePassword to set a new password (if the user has an e-mail address stored)
 - * **{USERNAME}**: it will be replaced with the username
- Users
 - **ONLINE_SECONDS**: time interval in seconds to consider a user online from last action registered
 - **CLEAR_PLAYER_SESSIONS**: if set to TRUE then every time a player logs in, the system will delete older login sessions to maintain the sessions table cleaned and as smaller as possible
 - **GUEST_PREFIX**: the prefix string attached to the id for guest accounts (ex.: "Guest-")
 - **FRIENDS_REQUIRE_ACCEPT** (default: FALSE): if set to TRUE then the friend add action will require the destination user to accept or decline the request before it appears in the friends list
- Upload
 - **UPLOAD_DELETE_OLD_FILES** (default: TRUE): if TRUE will keep your upload folder clean with latest changes to UserFile by deleting old file upon upload of new content
- Log
 - **LOG_FILENAME** (default: app.log): Name of the log file (in the folder /_logs)
 - **LOG_MAXFILESIZE**: it is the maximum size in bytes of the log file
 - **LOG_BANNED** (default: FALSE): if set to TRUE then the system will add the requests from banned IP address in the log file
 - **LOG_MAXFILESIZE** (default: FALSE): if set to TRUE then the system will add the unauthorized/invalid requests in the log file

4.2 Client configuration

In the inspector of **CombuManager** script (the prefab is just a `GameObject` with *CombuManager* script attached) you will find the following properties:

- **Dont Destroy On Load** (default: TRUE): if checked, this `GameObject` will be alive for the whole lifetime of your game/app; if you want to login in one scene (for example main menu) and last until the game quits then you should enable this flag
- **Set As Default Social Platform**: since **Combu** implements the Unity **ISocialPlatform** interface, if you enable this flag the **Combu** will be set as current platform on Unity and you will be able to use it also through **Social**.↔
Active
- **Url Root Production**: is the URL to the folder where you installed the web services on your production server (e.g.: <http://www.yourserver.com/combu/>)
- **Url Root Stage**: is the URL to the folder where you installed the web services on your stage/development server, usually local machine (e.g.: <http://localhost/combu/>)
- **Use Stage**: if checked, the web services calls will be directed to the stage server instead of production
- **Log Debug Info**: if checked, all the calls to webservices will add the result text in the console log
- **Remember Credentials**: if checked, the last successful login action will store the username and password (encrypted) in `PlayerPrefs` to be used with `User.AutoLogin()` later

- **Ping Interval Seconds** (default: 0): it's the interval in seconds to send auto-ping to server in order to maintain the online state of local user; to disable auto-ping set this to 0
- **Online Seconds** (default: 120): it's the time in seconds from last action registered to be considered as Online (last action and online state are cached in User class, you will need to reload over time if you need a precise info)
- **Playing Seconds** (default: 60): it's the time in seconds from last action to be considered as Playing
- **Language** (default: en): the language for server response messages localization
- **Achievement UI Object**: the GameObject that handles the user interface of Achievements
- **Achievement UI Function**: the name of method to call on **Achievement UI Object** as result to **Social**.↔
ShowAchievementsUI
- **Leaderboard UI Object**: the GameObject that handles the user interface of Leaderboard
- **Leaderboard UI Function**: the name of method to call on **Leaderboard UI Object** as result to **Social**.↔
ShowLeaderboardUI

Chapter 5

Email configuration and Reset Password

In this section you will learn how to setup your server to send email to clients and reset the user's password.

5.1 Server configuration

To enable sending mails to your users you need to set **REGISTER_EMAIL_REQUIRED** to **TRUE** in */lib/config.php* and start to store the users email address.

If you want to send an email as welcome after registration then you also need to set **REGISTER_EMAIL_ACTIVATION** to **TRUE**, then set your custom message for subject in **REGISTER_EMAIL_SUBJECT** and a file name (it must exist in */email_templates*) in **REGISTER_EMAIL_MESSAGE**.

We suggest to create a copy of */email_templates/email_register.html* and use it as your own, same for any other sample files that you find in */email_templates* which may be overwritten during updates).

The built-in method **User.ResetPassword** (to reset the password of a user when it has been lost) sends a random generated code by email to the user, so this feature also requires **REGISTER_EMAIL_REQUIRED** to **TRUE**. In your interface you need a frame where he can input the code received by email and you call **User.ChangePassword** like here:

```
// Generate a random code and send to the user's email address
User.ResetPassword("username", (bool success, string error) =>
{
    Debug.Log(success + " -- " + error);
});
// User has received the code 'codeReceivedByEmail' by email after ResetPassword
User.ChangePassword(0, "username", "codeReceivedByEmail", "newPasswordChosen", (bool success, string error)
=>
{
    Debug.Log("Success: " + success + " --- Error: " + error);
});
```

You can customize the email sent to the user by setting **RESETPWD_EMAIL_SUBJECT** and **RESETPWD_EMAIL_MESSAGE** (same as above for *REGISTER_EMAIL_**).

Chapter 6

Server-Client Security

Let's take a look at the implementation of security in [Combu](#)

6.1 RSA + AES keys handshaking

What happens when the Unity client starts:

1. **CombuManager** calls a web service to initialize the connection to the server (remember we talk about "connection" but HTTP is an asynchronous protocol, so we haven't a real-time synchronized connection like it could be in TCP)
2. the server generates private and public RSA keys on the fly for the incoming connection and returns back the public key to the client
3. the client then generates AES keys and send them back to the server encrypting them with the RSA public key received
4. now both the server and client have the same AES keys that can be used to encrypt/decrypt the content of every web service and the client is ready to work

The reason why we added a double layer of security and don't use only RSA is because RSA encryption/decryption requires more resources (and so time) than AES, besides the fact it effectively adds more security and standard management for such situations.

6.2 Data encryption

Every call to the web services after the initialization is sent to the server encrypted with the AES keys, here is an example of request:

```
http://yourserver/your_combu_path/server.php?token=Q0ItNThhNmQzNjVlMzlkYTguMTIzNTgxNDc%3d&data=2f
```

The server response is also encrypted with the same keys if `RESPONSE_ENCRYPTED` is defined and set to `TRUE` in `config.php`:

```
{"t": "2017-04-18 20:32:09", "d": "vWH4DE0wj+GI34Q0gHzuYWR\|e8rqqlG7h0ZxW5nQYzPG6Cv4aQ6BNY4L4T5LruFZA
```


Chapter 7

Error messages localization

Localize the error messages sent from server

7.1 Client request

You can decide in what language the server will output the error messages by setting the property **language** in the **CombuManager** instance.

7.2 Server response

The default language for messages output is 'en' (english), but if the client sends a specific language request then it is used instead. The file **'errors_{language}.php'** must exist in the folder */error_messages* of the server.

Chapter 8

Authentication and Users

In this section you will learn how to authenticate the local user, create a new user and load your users.

8.1 Authentication

To authenticate the local user you need to call **CombuManager.instance.platform.Authenticate**:

```
CombuManager.platform.Authenticate( "username", "password", (bool success, string error) => {
    if (success)
        Debug.Log("Login success: ID " + CombuManager.localUser.id);
    else
        Debug.Log("Login failed: " + error);
});
```

You can store the last successful login credentials when **Remember Credentials** is set to **true** on **CombuManager** inspector (or programmatically), the credentials will be stored in PlayerPref (password encrypted) to be used later with **User.AutoLogin()** method.

8.1.1 Auto-login

If you want to auto-login a previously saved session then you must have stored the credentials by setting **Remember Credentials** to **true** on the instance of **CombuManager** and then call **User.AutoLogin()**:

```
User.AutoLogin ((bool loginSuccess, string loginError) => {
    Debug.Log ("AutoLogin: " + loginSuccess + " -- " + loginError);
});
```

8.2 Registration

To create a new user you need to create a new instance of **User** class, set at least *username* and *password* and then call **Update** on the instance:

```
User newUser = new User();
newUser.userName = "username";
newUser.password = "password";
newUser.Update( (bool success, string error) => {
    // NB: registration does not make the user logged
    if (success)
        Debug.Log("Save success: ID " + newUser.id);
    else
        Debug.Log("Save failed: " + error);
});
```

8.2.1 Send confirmation by email

If you set **REGISTER_EMAIL_REQUIRED** to **TRUE** in *config.php* on your server, then you also need to assign the property **email** of your *User* object and, if also **REGISTER_EMAIL_ACTIVATION** is set to **TRUE**, it will send an email to the new users with a link to activate their account (to prevent spam registrations).

You can customize the content of the registration email by editing the HTML file */email_templates/email_register.html*, for more information please read the constants *EMAIL_** at [Server configuration](#).

If you want to test the sending of email on your local development server then you need to either install a mail server or set the *EMAIL_SMTP** defines in *config.php*

About sending out emails, **Combu** uses the library **PHPMailer**, so if you're having issues with sending options with SMTP you can create a small PHP test script that uses the PHPMailer class (doing a search on google produces tons of examples for the most common free mail servers like gmail) and when you find the correct settings with it then you can bring them in your *config.php*.

Here is the correspondence of *config.php* with PHPMailer properties:

- *EMAIL_SMTP*: if TRUE then call *PHPMailer->isSMTP()* to set PHPMailer to use SMTP
- *EMAIL_SMTP_HOSTNAME*: *PHPMailer->Host* and *PHPMailer->Hostname*
- *EMAIL_SMTP_PORT*: *PHPMailer->Port*
- *EMAIL_SMTP_SECURE*: *PHPMailer->SMTPSecure*
- *EMAIL_SMTP_USERNAME*: *PHPMailer->Username* (and set *PHPMailer->SMTPAuth* to TRUE)
- *EMAIL_SMTP_PASSWORD*: *PHPMailer->Password*

Example of */combu/test_mail.php*:

```
<?php
use PHPMailer\PHPMailer\PHPMailer;
use PHPMailer\PHPMailer\Exception;

include_once "vendor/autoload.php";

$mail = new PHPMailer(true);
try {
    $mail->isSMTP();
    $mail->Host = 'smtp1.example.com';
    $mail->SMTPAuth = true;
    $mail->Username = 'user@example.com';
    $mail->Password = 'secret';
    $mail->SMTPSecure = 'tls';
    $mail->Port = 587;

    $mail->setFrom('from@example.com', 'Mailer');
    $mail->addAddress('ellen@example.com');

    $mail->isHTML(true);
    $mail->Subject = 'Here is the subject';
    $mail->Body = 'This is the HTML message body <b>in bold!</b>';
    $mail->AltBody = 'This is the body in plain text for non-HTML mail clients';

    $mail->send();
    echo 'Message has been sent';
} catch (Exception $e) {
    echo 'Message could not be sent. Mailer Error: ', $mail->ErrorInfo;
}
```

8.2.2 Create a guest account

If you want to create a guest account then you need to call **CreateGuest** on a **User** object, you could activate **rememberCredentials** to use **AutoLogin** later:

```
// First try to auto-login
CombuManager.instance.rememberCredentials = true;
string storedUsername, storedPassword;
if (User.CanAutoLogin(out storedUsername, out storedPassword)) {
    Debug.Log ("Try to auto-login with username ' " + storedUsername + "' and password ' " + storedPassword +
        "'");
    User.AutoLogin ((bool loginSuccess, string loginError) => {
        Debug.Log ("AutoLogin: " + loginSuccess + " -- " + loginError);
        if (loginSuccess)
            Debug.Log ("Logged in");
    });
    return;
}
// If autologin fails then create a guest account
var user = new User ();
user.CreateGuest ((bool success, string error) => {
    if (success) {
        Debug.Log ("CreateGuest - Success=" + success + " > Error=" + error);
    }
});
```

8.3 Custom Data

Each user has a *Hashtable* property **customData** that you can use to store your own strings, numbers, JSON etc that should be visible from all Apps (for example a Premium/VIP status of the user in your games-hub).

Recently it was introduced another similar property **appCustomData** that you can use to set the properties, attributes, statistics etc which are relevant to the App that is currently running in the game.

By using these key/value dictionaries you can create the profile that better fits your need and extend it by code without any limitation at no cost, for example:

```
// Global custom data (all Apps)
CombuManager.localUser.customData["VIP"] = true;
// Local custom data (only current App)
CombuManager.localUser.appCustomData["Nickname"] = "MyNickname";
CombuManager.localUser.appCustomData["Gold"] = 100;
CombuManager.localUser.appCustomData["Experience"] = 1000;
// Send the changes to the server
CombuManager.localUser.Update((bool success, string error) => {
    if (success)
        Debug.Log("Saved");
    else
        Debug.Log("Failed: " + error);
});
```

8.4 Load users

To load the users data you can call **CombuManager.instance.platform.LoadUsers()**, or one of the **User.Load()** overloads (with *User.Load* form you will not need to cast back from **IUserProfile** to **User**):

```
// Load a user by Id
User.Load ( 123, ( User user ) => {
    Debug.Log("Success: " + (user == null ? "false" : "true"));
});
// Load a user by userName
User.Load ( "user1", ( User user ) => {
    Debug.Log("Success: " + (user == null ? "false" : "true"));
});
// Load users by Id
User.Load ( new long[] { 123, 456 }, ( User[] users ) => {
    Debug.Log("Loaded: " + users.Length);
});
// Search users by Username, Email, CustomData
// Filter players with custom data "Level" between 5 and 10
SearchCustomData[] searchData = new SearchCustomData[] {
    new SearchCustomData("Level", eSearchOperator.GreaterOrEquals, "5"),
    new SearchCustomData("Level", eSearchOperator.LowerOrEquals, "10")
};
User.Load("part-of-username", "email@server.com", searchData, 1, 1, (User[] users, int resultsCount, int
    pageCount) => {
    Debug.Log("Loaded: " + users.Length);
});
```

You can also load a list of random users (excluding localUser):

```
// Filter players with custom data "Level" between 5 and 10
SearchCustomData[] searchData = new SearchCustomData[] {
    new SearchCustomData("Level", eSearchOperator.GreaterOrEquals, "5"),
    new SearchCustomData("Level", eSearchOperator.LowerOrEquals, "10")
};
User.Random(searchData, 3, (User[] users) => {
    foreach (User user in users)
    {
        if (user.lastSeen == null)
            Debug.Log(user.userName + " Never seen");
        else
        {
            System.DateTime seen = (System.DateTime)user.lastSeen;
            Debug.Log(user.userName + " Last seen: " + seen.ToLongDateString() + " at " + seen.
                ToLongTimeString() + " - Online state: " + user.state);
        }
    }
});
```

8.5 Online state

The Profile class implements the **IUserProfile** interface, so it also provides the *state* property to get the online state of your users. Of course remember that we are in an asynchronous environment, so your players are not really connected in real-time and if you need to rely on the online state then you will need to implement your own polling system to refresh your lists from time to time.

To maintain the online state CombuManager uses the settings *pingIntervalSeconds*, *onlineSeconds* and *playingSeconds*. Besides the ping function that is called every *pingIntervalSeconds* seconds (set 0 to disable, anyway we'd recommend to have the interval not too small, a value of 30 should be fine else you may suffer of high traffic), every action served by the webservises updates the "last action" date/time of a user.

8.6 Create your User class

Since you're able to extend the basic **User** class with the *customData* property, the best way to work with the system is to create your own class for users by inheriting from **User**.

This way you can create your own account properties, that for sure are much more readable than *customData["myProperty"]* (especially if you need non-string values, it would be lot of explicit casts or Parse!).

Remember to:

- set their values in *customData*, so they will be passed to **Update** and saved to server
- override **FromHashtable** to fill the internal variables from the *customData* received from server

```
public class CombuDemoUser : Combu.User
{
    string _myProperty1 = "";
    int _myProperty2 = 0;

    public string myProperty1
    {
        get { return _myProperty1; }
        set { _myProperty1 = value; customData["myProperty1"] = _myProperty1; }
    }

    public int myProperty2
    {
        get { return _myProperty2; }
        set { _myProperty2 = value; customData["myProperty2"] = _myProperty2; }
    }

    public CombuDemoUser()
    {
        myProperty1 = "";
        myProperty2 = 0;
    }

    public override void FromHashtable (Hashtable hash)
    {
        // Set User class properties
        base.FromHashtable (hash);

        // Set our own custom properties that we store in customData
        if (customData.ContainsKey("myProperty1"))
            _myProperty1 = customData["myProperty1"].ToString();
        if (customData.ContainsKey("myProperty2"))
            _myProperty2 = int.Parse(customData["myProperty2"].ToString());
    }
}
```

To use the new class in your code, you will need to pass the referenced type to the user-wise methods:

```
// Authenticate user
CombuManager.platform.Authenticate <CombuDemoUser> ("username", "password", (bool success, string error) =
> {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Create new user
CombuDemoUser newUser = new CombuDemoUser();
newUser.userName = "username";
newUser.password = "password";
newUser.myProperty1 = "Value";
newUser.myProperty2 = 100;
newUser.Update( (bool success, string error) => {
    // NB: registration does not make the user logged
    if (success)
        Debug.Log("Save success: ID " + newUser.id);
    else
        Debug.Log("Save failed: " + error);
});
```


Chapter 9

Server Settings

In this section you will learn how to create server settings and access them from client.

9.1 Server

You can create server settings in the administration console, in the section **Settings**.

9.2 Client

The client automatically loads the settings in **CombuManager.instance.serverInfo.settings**:

```
while (!CombuManager.isInitialized)
    yield return null;
string mySetting = CombuManager.instance.serverInfo.settings["mySetting"].ToString();
```


Chapter 10

Linking external platforms

In this section you will learn how to authenticate and link the local user to an external platform like GameCenter, Facebook etc.

10.1 Authentication

To authenticate the local user with a platform Id you need to call **CombuManager.localUser.AuthenticatePlatform**. If the platform key+id exists then it will return the registered account, else it will create a new account with username **PlatformName_PlatformId** (including the underscore symbol).

```
// After you have logged in with Facebook SDK (http://u3d.as/5j1)
CombuManager.localUser.AuthenticatePlatform("Facebook", FB.UserId, (bool success, string error) => {
    if (success)
        Debug.Log("Login success: ID " + CombuManager.localUser.id);
    else
        Debug.Log("Login failed: " + error);
});
```

10.2 Link a platform to the local user

If the local user is already logged, you can link a platform Id to the account with **CombuManager.localUser.LinkPlatform**:

```
CombuManager.localUser.LinkPlatform("YourPlatformName", "YourPlatformId", (bool success, string error) => {
    if (success)
        Debug.Log("Link success");
    else
        Debug.Log("Link failed: " + error);
});
```

10.3 Transfer the external platforms

Sometimes may happen that you need to move the external platforms of the local user to another account, in this case you can use **CombuManager.localUser.LinkAccount** (the platforms key+id of the local user account will be transferred to the new account, the account and all its data/scores/etc deleted, and the new account will be assigned to the local user):

```
CombuManager.localUser.LinkAccount("other_username", "other_password", (bool success, string error) => {
    if (success)
        Debug.Log("Transfer success");
    else
        Debug.Log("Transfer failed: " + error);
});
```

10.4 Load users from platforms

If your users have logged with their platform profile then you can retrieve their [Combu](#) profiles later (for example after a call to Facebook API that returns the list of friends to show who is playing your game by matching the existence of a profile in [Combu](#)):

```
// Assign platform and related id to all users that you want to search, for example coming from Facebook
// friends API
List<string> platforms = new List<string>() { "Facebook", "Facebook" }; // you have passed "Facebook" as
// platform key to AuthenticatePlatform
List<string> ids = new List<string>() { "000000", "111111" };
User.LoadPlatform(platforms, ids, (User[] users) => {
    Debug.Log("Users found: " + users.Length);
});
```

Chapter 11

Managing Contacts

In this section you will learn how to retrieve the contacts lists of the local user and how to add/remove users to the friends/ignore lists.

11.1 Loading Contacts

To retrieve the list of contacts from the local user you need to call the *LoadFriends* method on **CombuManager.localUser** and then access to the results list accordingly to the *eContactType* value passed to the function (**CombuManager.localUser.friends** for *eContactType.Friend*, **CombuManager.localUser.requests** for *eContactType.Request*, **CombuManager.localUser.ignored** for *eContactType.Ignore*, **CombuManager.localUser.pendingRequests** for *eContactType.PendingRequest*):

```
CombuManager.localUser.LoadFriends( eContactType.Friend, (bool success) => {
    if (success)
        Debug.Log("Success: " + CombuManager.localUser.friends.Length);
    else
        Debug.Log("Failed");
});
```

11.2 Adding Contacts

To add another user to a contact list of the local user you need to call *AddContact* and pass either a User/Profile object or a username and *eContactType.Friend* as contact type:

```
// Add by User/Profile object
CombuManager.localUser.AddContact (otherUser, eContactType.Friend, (bool success, string error)
=> {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Add by username
CombuManager.localUser.AddContact ("username", eContactType.Friend, (bool success, string error)
=> {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```

11.3 Removing Contacts

To remove a user from the contact lists of the local user you need to call *RemoveContact*.

```
// Remove by User/Profile object
CombuManager.localUser.RemoveContact(otherUser, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Remove by username
CombuManager.localUser.RemoveContact("username", (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```

Chapter 12

Managing Files

In this section you will learn how to retrieve the files lists of the local user or anyway accessible by him and how to add/remove files.

12.1 Loading Files

To retrieve a list of files you need to call **UserFile.Load**:

```
bool includeShared = true;
int pageNumber = 1;
int countPerPage = 10;
UserFile.Load(CombuManager.localUser.id, includeShared, pageNumber, countPerPage, (UserFile[] files, int
    resultsCount, int pagesCount, string error) => {
    if (string.IsNullOrEmpty(error))
        Debug.Log("Files loaded: " + files.Length);
    else
        Debug.Log(error);
});
```

12.2 Adding Files

To add a file associated to the local user you need to create a new instance of **UserFile**, set *sharing* property and call **Update**:

```
byte[] screenshot = CombuManager.instance.CaptureScreenShot();
UserFile newFile = new UserFile();
newFile.sharing = UserFile.eShareType.Everybody;
newFile.customData ["Prop"] = "Value";
newFile.Update(screenshot, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```

12.3 Viewing Files

To increase the **View** count of a file you need to call **UserFile.View**, or call the method **View** on a **UserFile** instance:

```
// View by File ID
UserFile.View(123, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// View by UserFile object
myFile.View( (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```

12.4 Liking Files

To increase the **Like** count of a file you need to call **UserFile.Like**, or call the method **Like** on a **UserFile** instance:

```
// Like by File ID
UserFile.Like(123, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Like by UserFile object
myFile.Like( (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```

12.5 Removing Files

To remove a file owned by the local user you need to call **UserFile.Delete**, or call the method **Delete** on a **UserFile** instance:

```
// Remove by File ID
UserFile.Delete(123, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Remove from a UserFile object
myFile.Delete( (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```


Chapter 13

Managing Inventory

In this section you will learn how to manage the inventory of the local user.

13.1 Loading Inventory

To retrieve the list of items in the inventory of a user you need to call the **Inventory.Load**:

```
// Load the inventory of user ID '123'
Inventory.Load("123", (Inventory[] items, string error) => {
    if (success)
        Debug.Log("Success: " + items.Length);
    else
        Debug.Log("Failed: " + error);
});
```

13.2 Adding and Editing Items

To add a new item in the inventory of the local user you need to create a new **Inventory** instance, set *name*, *quantity* and *customData* and then call **Update**:

```
// Add a new item
Inventory newItem = new Inventory();
newItem.name = "My item";
newItem.quantity = 1;
newItem.customData["Durability"] = 100;
newItem.Update( (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Edit an item previously loaded
myItem.quantity--;
myItem.Update( (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```

13.3 Removing Items

To remove an item from the inventory of the local user you need to call **Inventory.Delete**:

```
// Remove by inventory ID '123'
Inventory.Delete(123, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Remove by Inventory object
myItem.Delete( (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```

Chapter 14

Managing Messages

In this section you will learn how to manage the in-game inbox of the local user.

14.1 Loading Messages

To retrieve the list of messages of a user you need to call the **Message.Load**:

```
// Load the received messages from the page 1 with 3 results per page
Mail.Load(eMailList.Received, 1, 3, (Mail[] messages, int count, int pageCount, string error) =>
{
    if (string.IsNullOrEmpty(error))
        Debug.Log("Success: " + messages.Length);
    else
        Debug.Log("Failed: " + error);
});
```

14.2 Sending Messages

To send a new message to a user you need to call one of the overloads of **Mail.Send**:

```
// Send a private message to a user by Id
Mail.Send(123, "Subject", "Message body", false, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Send a private message to a user by Username
Mail.Send("user1", "Subject", "Message body", false, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Send a private message to multiple users by Id
Mail.Send(new long[] { 123, 456 }, "Subject", "Message body", false, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Send a private message to multiple users by Username
Mail.Send(new string[] { "user1", "user2" }, "Subject", "Message body", false, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```

14.2.1 Sending to UserGroup

If you want to send a message to a UserGroup then you need to call **Mail.SendMailToGroup**:

```
// Send a message to the UserGroup ID '123'
Mail.SendMailToGroup(123, "Subject", "Message body", false, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```

14.3 Marking Messages as Read

To mark a message as **Read** you need to call **Mail.Read**, or call the method **Read** on a **Mail** instance:

```
// Mark as Read by Mail ID
Mail.Read( new long[] { 123, 456 }, new long[0], (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Mark as Read by Group ID
Mail.Read( new long[0], new long[] { 123, 456 }, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Mark as Read by Mail object
myMail.Read( (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```

14.4 Marking Messages as Unread

To mark a message as **Unread** you need to call **Mail.Unread**, or call the method **Unread** on a **Mail** instance:

```
// Mark as Unread by Mail ID
Mail.Unread( 123, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Mark as Unread by Mail object
myMail.Unread( (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```

14.5 Removing Items

To delete a message you need to call **Message.Delete**, or call the method **Delete** on a **Mail** instance:

```
// Remove by Message ID '123'
Mail.Delete(123, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Remove by Mail object
myMail.Delete( (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```

14.6 Loading Conversations

To load the list of discussions of the local user with others you can call **Mail.LoadConversations**, it will fill an **ArrayList** with objects of type **User** or **UserGroup** (based on the *idGroup* associated to the **Mail** object):

```
Mail.LoadConversations( (ArrayList conversations, int count, string error) => {
    if (string.IsNullOrEmpty(error))
        Debug.Log("Success: " + conversations.Count);
    else
        Debug.Log("Failed: " + error);
});
```

14.7 Counting Messages

To count the messages in the inbox of the local user you can call **Mail.Count**:

```
// Count the messages sent from users ID
Mail.Count( new long[] { 123, 456 }, new long[0], (MailCount[] counts, string error) => {
    if (string.IsNullOrEmpty(error))
        Debug.Log("Success: " + counts.Length);
    else
        Debug.Log("Failed: " + error);
});
// Count the messages sent from groups ID
Mail.Count( new long[0], new long[] { 123, 456 }, (MailCount[] counts, string error) => {
    if (string.IsNullOrEmpty(error))
        Debug.Log("Success: " + counts.Length);
    else
        Debug.Log("Failed: " + error);
});
```


Chapter 15

Managing User Groups

In this section you will learn how to manage the user groups.

15.1 Create a new group

You can create a new group with **UserGroup.Save** (call this also to edit a group that was loaded):

```
UserGroup group = new UserGroup();
group.name = "My Group";
// Add some users
group.users = new User[] { {userName="OtherUser1"}, {userName="OtherUser2"} };
group.Save((bool success, string error) => {
    Debug.Log("Group saved: " + success);
});
```

15.2 Load groups

To load groups you have 3 choices:

- load groups owned by local user

```
UserGroup.Load(CombuManager.localUser.idLong, (UserGroup[] groups, string error) => {
    Debug.Log(groups.Length);
});
```

- load groups in which local user is a member (owners are also members)

```
UserGroup.LoadMembership(CombuManager.localUser.idLong, (UserGroup[] groups, string error) => {
    Debug.Log(groups.Length);
});
```

15.3 Join a group

You can join a group with **UserGroup.Join**:

```
// Join local user
group.Join((bool success, string error) => {
    Debug.Log("Group joined: " + success);
});
// Join a list of users
group.Join(new string[] {"user1"}, (bool success, string error) => {
    Debug.Log("Group joined: " + success);
});
```

15.4 Leave a group

You can leave a group with **UserGroup.Leave**:

```
// Leave local user
group.Leave((bool success, string error) => {
    Debug.Log("Group joined: " + success);
});
// Leave a list of users
group.Leave(new string[] {"user1"}, (bool success, string error) => {
    Debug.Log("Group joined: " + success);
});
```


Chapter 16

Managing News

In this section you will learn how to retrieve the in-game news.

16.1 Loading News

To retrieve the list of latest news you need to call **News.Load**:

```
// Load the page 1 with 10 results per page
News.Load(1, 10, (News[] news, int resultsCount, int pageCount, string error) => {
    if (string.IsNullOrEmpty(error))
        Debug.Log("Success: " + news.Length);
    else
        Debug.Log("Failed: " + error);
});
```


Chapter 17

Managing Leaderboards

In this section you will learn how to access the leaderboards data and report a score.

17.1 Loading Scores

To retrieve the scores list of a leaderboard you can call **CombuManager.platform.LoadScores** (by default loads the first page with 10 results), or you can instantiate a new **Leaderboard** object (created with **CombuManager.platform.CreateLeaderboard**), set *timeScope*, *customTimescope* and *range* and then call **LoadScores**:

```
// Load the scores of the leaderboard ID '123' from the page 2 with 10 results per page
CombuManager.platform.LoadScores(123, 2, 10, (IScore[] scores) => {
    Debug.Log("Scores loaded: " + scores.Length);
});
// Load the scores of the leaderboard ID '123' from the page 1 with 10 results per page
ILeaderboard board = CombuManager.platform.CreateLeaderboard ();
board.id = "123";
board.timeScope = UnityEngine.SocialPlatforms.TimeScope.AllTime;
board.range = new UnityEngine.SocialPlatforms.Range(1, 10);
board.LoadScores((bool loaded) => {
    // Now you can access board.scores and board.localUserScore
});
```

17.2 Reporting Scores

To report a new score of the local user you need to call **CombuManager.platform.ReportScore**, or you can call the method **ReportScore** on a **Score** instance:

```
// Report a score of 1000 to the leaderboard ID '123'
CombuManager.platform.ReportScore(1000, "123", (bool success) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed");
});
// Report a score of 1000 on a Score object,
// (previously loaded with LoadScores or a new with leaderboardID set)
myScore.value = 1000;
myScore.ReportScore( (bool success) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed");
});
```

17.3 Loading Leaderboards data

To load the leaderboards data like the title, description etc, you need to call **Leaderboard.Load**:

```
// Load the leaderboard ID '123'
Leaderboard.Load("123", (Leaderboard leaderboard, string error) => {
    if (leaderboard != null)
        Debug.Log("Success: " + leaderboard.title);
    else
        Debug.Log("Failed: " + error);
});
```

17.4 Loading the score of a user

To retrieve the best score of a user you need to call **Leaderboard.LoadScoresByUser**:

```
// Load the best score of leaderboard ID '123' for the User object with 10 results per page (to be
// attainable 'page', the limit must be the same as used with LoadScores)
Leaderboard.LoadScoresByUser("123", user, eLeaderboardInterval.Week, 10, (Score score,
int page, string error) => {
    if (score != null)
        Debug.Log("Success: " + score.value + " at page " + page);
    else
        Debug.Log("Failed: " + error);
});
```

Chapter 18

Managing Achievements

In this section you will learn how to load the achievements and report a progress.

18.1 Loading Achievements descriptions

To retrieve the list of achievements you can call **CombuManager.platform.LoadAchievementDescriptions** or **CombuManager.platform.LoadAchievements** (the latter form is preferred because you will not need to cast back from *IAchievement* to *Achievement*):

```
// Load the achievement descriptions
CombuManager.platform.LoadAchievements( (Achievement[] achievements) => {
    Debug.Log("Achievements loaded: " + achievements.Length);
});
```

18.2 Reporting Progress

To report a new progress of the local user you need to call **CombuManager.platform.ReportProgress**, or you can call the method **ReportProgress** on a **Score** instance:

```
// Report a score of 1000 to the achievement ID '123'
CombuManager.platform.ReportProgress(1000, "123", (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Report a progress of 30% on an Achievement object,
// (previously loaded with LoadAchievements or created with
// <strong>CombuManager.platform.CreateAchievement</strong> and with <em>id</em> set)
myAchievement.percentCompleted = 0.3;
myAchievement.ReportProgress( (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```


Chapter 19

Managing Tournaments

In this section you will learn how to retrieve the tournaments list of the local user and how to add/remove tournaments.

19.1 Loading Tournaments

To retrieve the list of tournaments accessible by the local user you need to call **Tournament.Load**:

```
// Load the active tournaments, no filter by customData
Tournament.Load(false, null, (Tournament[] tournaments) => {
    Debug.Log("Tournaments loaded: " + tournaments.Length);
});
```

19.1.1 Loading by Tournament ID

You can also load a single Tournament by its ID:

```
// Load a Tournament by ID
Tournament.Load(123, (Tournament tournament) => {
    if (tournament != null)
        Debug.Log("Success: " + tournament.title);
    else
        Debug.Log("Failed");
});
```

19.1.2 Matches of the Tournament

Once that the Tournament has been loaded, the property *matches* gives you access to its matches and their extra data.

19.2 Quick Tournament

To create a quick tournament with other users you need to call **Tournament.QuickTournament**:

```
// Load 2 random users
User.Random( null, 2, (User[] users) => {
    Tournament t = Tournament.QuickTournament(users);
    if (t.matches.Count == 0)
    {
        Debug.Log("Something going wrong, no matches created");
    }
    else
    {
        t.title = "My Tournament 1";
        t.customData["Key1"] = "Value";

        t.Save((bool success, string error) => {
            Debug.Log("Success: " + success + " - Matches: " + t.matches.Count);
        });
    }
});
```

19.2.1 Create your own Tournament

You can take a look at the code in **Tournament.QuickTournament** to see how to create a **Tournament** and use the code as base to create your own type of tournaments.

19.3 Removing Tournaments

To delete a tournament you need to call **Tournament.Delete**, or call the method **Delete** on a **Tournament** instance:

```
// Remove by Tournament ID
Tournament.Delete(123, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Remove by Tournament object
myTournament.Delete( (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```


Chapter 20

Managing Matches

In this section you will learn how to retrieve the matches list of the local user and how to add/remove matches.

20.1 Loading Matches

To retrieve the list of matches of the local user you need to call **Match.Load**:

```
// Load the active matches, no Tournament ID, no filter by title
Match.Load(0, true, string.Empty, (Matches[] matches) => {
    Debug.Log("Matches loaded: " + matches.Length);
});
```

20.1.1 Loading by Match ID

You can also load a single Matches by its ID:

```
// Load a Matches by ID
Match.Load(123, (Match match) => {
    if (match != null)
        Debug.Log("Success: " + match.title);
    else
        Debug.Log("Failed");
});
```

20.1.2 Rounds of the Match

Once that the Match has been loaded, the property *rounds* gives you access to its rounds and scores: the collection *users* contains the **MatchAccount** objects (relationship between Match and Account), the collection *scores* contains the **MatchRound** objects (relationship between the Match-Account association and a round).

20.2 Quick Match

To create a quick match with another user you need to call **Match.QuickMatch**:

```
// Load 2 random users (not only friends), no filter by customData, 1 round
Match.QuickMatch(false, null, 1, (Match match) => {
    if (match != null)
        Debug.Log("Success: " + match.title);
    else
        Debug.Log("Failed");
});
```

20.2.1 Create your own Match

You can take a look at the code in **Match.QuickMatch** to see how to create a **Match** and use the code as base to create your own matches.

20.3 Sending Score

To send a score for the next round you need to call the method **Score** on a **Match** instance:

```
// Send a score of 1000
myMatch.Score(1000, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```

20.4 Removing Matches

To delete a match you need to call **Match.Delete**, or call the method **Delete** on a **Match** instance:

```
// Remove by Match ID
Match.Delete(123, (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// Remove by Match object
myMatch.Delete( (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```

Chapter 21

Customize Web Administration

Learn how to customize the web administration through CSS and JS

21.1 Javascript

You can add your own JS code by creating a file called **custom.js** in the folder **/admin/js**, if this file exists then it will be automatically added to the HTML output of the web administration.

21.2 CSS

You can override any CSS class by creating a file called **custom.css** in the folder **/admin/css**, if this file exists then it will be automatically added to the HTML output of the web administration.

For example to change the [Combu](#) logo displayed at top left of every page with your own, you can upload your own logo into the folder **/admin/images** and create the file **/admin/css/custom.css** with the following content:

```
#logo-title {background-image: url('../images/your_logo.png');}
```


Chapter 22

Frequently Asked Questions

Read the FAQ if you're having issues before sending a post on forum.

22.1 How do I install Combu on my local/live server?

Read the section [Web server setup](#) and [Server and Client configuration](#).

22.2 How can I add my own properties to accounts?

In **Combu** you don't need to manually modify the core user class or database table to add your own properties to accounts like *FirstName*, *LastName*, *Coins* and so on.

All you need is to use user's **customData**: it's a Hashtable that can store every data you need. Let's see how you can extend the core user class and add a couple of custom properties as example.

First you need to create a class inheriting from *User*:

```
using System.Collections;
using Combu;

public class CombuDemoUser : User
{
    string _myProperty1 = "";
    int _myProperty2 = 0;
    int _coins = 0;

    public string myProperty1
    {
        get { return _myProperty1; }
        set { _myProperty1 = value; customData["myProperty1"] = _myProperty1; }
    }

    public int myProperty2
    {
        get { return _myProperty2; }
        set { _myProperty2 = value; customData["myProperty2"] = _myProperty2; }
    }

    public int coins
    {
        get { return _coins; }
        set { _coins = value; customData["Coins"] = _coins; }
    }

    public CombuDemoUser()

```

```

    {
        myProperty1 = "";
        myProperty2 = 0;
        coins = 0;
    }

    public override void FromHashtable (Hashtable hash)
    {
        // Set User class properties
        base.FromHashtable (hash);

        // Set our own custom properties that we store in customData
        if (customData.ContainsKey("myProperty1"))
            _myProperty1 = customData["myProperty1"].ToString();
        if (customData.ContainsKey("myProperty2"))
            _myProperty2 = int.Parse(customData["myProperty2"].ToString());
        if (customData.ContainsKey("Coins"))
            _coins = int.Parse(customData["Coins"].ToString());
    }
}

```

Now we can use this new class for Authenticate and AutLogin to safely cast **CombuManager.instance.loggedUser** after a successful login (similar syntax can be used for other methods like **User.AutoLogin** or **CombuManager.LoadContacts**):

```

CombuManager.platform.Authenticate<CombuDemoUser> (username, password, (bool loginSuccess, string
    loginError) => {
    CombuDemoUser myUser = (CombuDemoUser)CombuManager.localUser;
    Debug.Log("Login: " + loginSuccess + " -- " + loginError);
    if (loginSuccess)
        Debug.Log("Property1: " + myUser.myProperty1);
});

```

22.3 I purchased it on Asset Store, can I download from your website?

Yes, you can redeem your Unity invoice (both server and addons packages) from [here](#) and you will get free access to all downloads that you purchased on Asset Store.

22.4 When I try to navigate to the web admin, I see a blank page

If you see a blank page then there's probably an internal server error and your web server is configured to hide errors from web pages, so your first step is to enable "display_errors" variable in *php.ini* or *.htaccess* (search on internet or ask to your hosting provider how to do).

Remember the requirements that your web server should meet:

- PHP version must be 5.5 or greater
- the zip extension module must be enabled on PHP (if it isn't enabled then the auto-updater will not be able to uncompress and install the updates, but you can still use it to automatically execute the SQL queries if required)
- to successfully use the auto-updater feature, the server user (Apache user on OSX/Linux or IUSR/NETW↔ ORK_SERVICE on Windows) must have write permissions on **Combu** root folder

Also make sure you have set the correct connection settings in your */lib/config.php*.

Chapter 23

Namespace Index

23.1 Packages

Here are the packages with brief descriptions (if available):

Combu	This class encodes and decodes JSON strings. Spec. details, see http://www.json.org/	59
-----------------------	---	----

Chapter 24

Hierarchical Index

24.1 Class Hierarchy

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

Combu.CombuEncryption	64
Combu.CombuForm	67
Combu.CombuServerInfo	95
IAchievement	
Combu.Achievement	63
IAchievementDescription	
Combu.Achievement	63
IComparer	
Combu.Achievement	63
ILeaderboard	
Combu.Leaderboard	100
ILocalUser	
Combu.User	138
Combu.Inventory	96
IScore	
Combu.Score	131
ISocialPlatform	
Combu.CombuPlatform	82
IUserProfile	
Combu.Profile	127
Combu.User	138
Combu.Mail	105
Combu.MailCount	112
Combu.Match	113
Combu.MatchAccount	119
Combu.MatchRound	121
Combu.Match.MatchRoundData	122
Combu.MiniJSON	122
MonoBehaviour	
Combu.CombuManager	70
Combu.News	125
Combu.ProfilePlatform	131
Combu.SearchCustomData	133
Combu.Tournament	134
Combu.UserFile	155
Combu.UserGroup	162

Chapter 25

Class Index

25.1 Class List

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

Combu.Achievement	
Achievement class implementing the Unity built-in Social interfaces (IAchievement, I↔AchievementDescription).	63
Combu.CombuEncryption	64
Combu.CombuForm	67
Combu.CombuManager	
Combu Manager class, it follows the Singleton pattern design (accessible through the 'instance' static property), it means that you shouldn't have more than one instance of this component in your scene.	70
Combu.CombuPlatform	
Combu Platform implementation of Unity built-in Social interfaces (ISocialPlatform).	82
Combu.CombuServerInfo	
Class to handle Combu server informations.	95
Combu.Inventory	96
Combu.Leaderboard	100
Combu.Mail	105
Combu.MailCount	112
Combu.Match	113
Combu.MatchAccount	119
Combu.MatchRound	121
Combu.Match.MatchRoundData	122
Combu.MiniJSON	122
Combu.News	125
Combu.Profile	127
Combu.ProfilePlatform	131
Combu.Score	131
Combu.SearchCustomData	
Class to handle generic a generic search filter.	133
Combu.Tournament	
Tournaments class.	134
Combu.User	
User class implementing the Unity built-in Social interfaces (specialized IUserProfile, ILocalUser).	138
Combu.UserFile	155
Combu.UserGroup	162

Chapter 26

Namespace Documentation

26.1 Combu Namespace Reference

This class encodes and decodes JSON strings. Spec. details, see <http://www.json.org/>

Classes

- class [Achievement](#)
Achievement class implementing the Unity built-in Social interfaces (*IAchievement*, *IAchievementDescription*).
- class [CombuEncryption](#)
- class [CombuForm](#)
- class [CombuManager](#)
Combu Manager class, it follows the Singleton pattern design (accessible through the 'instance' static property), it means that you shouldn't have more than one instance of this component in your scene.
- class [CombuPlatform](#)
Combu Platform implementation of Unity built-in Social interfaces (*ISocialPlatform*).
- class [CombuServerInfo](#)
Class to handle *Combu* server informations.
- class **CombuUtils**
- class [Inventory](#)
- class [Leaderboard](#)
- class [Mail](#)
- class [MailCount](#)
- class [Match](#)
- class [MatchAccount](#)
- class [MatchRound](#)
- class [MiniJSON](#)
- class **MiniJsonExtensions**
- class [News](#)
- class [Profile](#)
- class [ProfilePlatform](#)
- class [Score](#)
- class [SearchCustomData](#)
Class to handle generic a generic search filter.
- class [Tournament](#)
Tournaments class.
- class [User](#)
User class implementing the Unity built-in Social interfaces (specialized *IUserProfile*, *ILocalUser*).
- class [UserFile](#)
- class [UserGroup](#)

Enumerations

- enum `eContactType` : int { **PendingRequest** = -1, **Friend** = 0, **Request**, **Ignore**, **MinValue** = Friend, **MaxValue** = Ignore }
Contact type.
- enum `eMailList` : int { **Received** = 0, **Sent**, **Both**, **Unread** }
- enum `eSearchOperator` { **Equals**, **Disequals**, **Like**, **Greater**, **GreaterOrEquals**, **Lower**, **LowerOrEquals** }
Custom search operator.
- enum `eLeaderboardInterval` : int { **Total** = 0, **Month**, **Week**, **Today** }
Leaderboard interval.
- enum `eLeaderboardTimeScope` : int { **None**, **Month** }
Leaderboard time scope.

26.1.1 Detailed Description

This class encodes and decodes JSON strings. Spec. details, see <http://www.json.org/>

All enum types used in `Combu` namespace are defined here

JSON uses Arrays and Objects. These correspond here to the datatypes `ArrayList` and `Hashtable`. All numbers are parsed to doubles.

26.1.2 Enumeration Type Documentation

26.1.2.1 eContactType

```
enum Combu.eContactType : int [strong]
```

Contact type.

26.1.2.2 eLeaderboardInterval

```
enum Combu.eLeaderboardInterval : int [strong]
```

Leaderboard interval.

26.1.2.3 eLeaderboardTimeScope

```
enum Combu.eLeaderboardTimeScope : int [strong]
```

Leaderboard time scope.

26.1.2.4 eMailList

```
enum Combu.eMailList : int [strong]
```

Mail list.

26.1.2.5 eSearchOperator

```
enum Combu.eSearchOperator [strong]
```

Custom search operator.

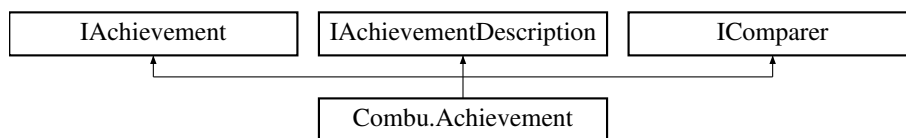
Chapter 27

Class Documentation

27.1 Combu.Achievement Class Reference

[Achievement](#) class implementing the Unity built-in Social interfaces (IAchievement, IAchievementDescription).

Inheritance diagram for Combu.Achievement:



Public Member Functions

- **Achievement** (string jsonString)
- virtual void **FromJson** (string jsonString)
- void **ReportProgress** (System.Action< bool > callback)
- int **Compare** (object x, object y)

Protected Attributes

- Texture2D **_image** = null

Properties

- string **id** [get, set]
- double **percentCompleted** [get, set]
- bool **completed** [get]
- bool **hidden** [get]
- System.DateTime **lastReportedDate** [get]
- string **title** [get]
- string **description** [get]
- int **finished** [get]
- Texture2D **image** [get]
- string **achievedDescription** [get]
- string **unachievedDescription** [get]
- int **points** [get]

27.1.1 Detailed Description

[Achievement](#) class implementing the Unity built-in Social interfaces (IAchievement, IAchievementDescription).

27.2 Combu.CombuEncryption Class Reference

Public Member Functions

- void [SetToken](#) (string sessionToken)
Sets the session token.
- void [LoadRSA](#) (string sessionToken, string xml)
Sets the session token and loads the RSA data from XML (Public Key).
- void [LoadRSA](#) (string sessionToken, string modulus, string exponent)
Sets the session token and loads the Modulus and Exponent of the RSA Public Key.
- string [EncryptRSA](#) (string inputString)
Encrypts a string with RSA.
- string [EncryptAES](#) (string inputString)
Encrypts a string with AES.
- string [DecryptAES](#) (string inputString)
Decrypts a string what was encrypted with the current AES Key/IV.
- string [EncryptMD5](#) (string inputString)
Encrypts a string in MD5 hash.
- string [EncryptSHA1](#) (string inputString)
Encrypts a string in SHA1 hash.
- string [DecryptResponse](#) (string text)
Decrypts the response from server.

Properties

- string **Token** [get]
- string **Key** [get]
- string **IV** [get]

27.2.1 Member Function Documentation

27.2.1.1 DecryptAES()

```
string Combu.CombuEncryption.DecryptAES (  
    string inputString )
```

Decrypts a string what was encrypted with the current AES Key/IV.

Returns

The decrypted string.

Parameters

<i>inputString</i>	Input string.
--------------------	---------------

27.2.1.2 DecryptResponse()

```
string Combu.CombuEncryption.DecryptResponse (  
    string text )
```

Decrypts the response from server.

Returns

The response decrypted.

Parameters

<i>text</i>	Response text.
-------------	----------------

27.2.1.3 EncryptAES()

```
string Combu.CombuEncryption.EncryptAES (  
    string inputString )
```

Encrypts a string with AES.

Returns

The AE.

Parameters

<i>inputString</i>	Input string.
--------------------	---------------

27.2.1.4 EncryptMD5()

```
string Combu.CombuEncryption.EncryptMD5 (  
    string inputString )
```

Encrypts a string in MD5 hash.

Returns

The MD5 encrypted string.

Parameters

<i>inputString</i>	Input string.
--------------------	---------------

27.2.1.5 EncryptRSA()

```
string Combu.CombuEncryption.EncryptRSA (  
    string inputString )
```

Encrypts a string with RSA.

Returns

The RS.

Parameters

<i>inputString</i>	Text to encrypt.
--------------------	------------------

27.2.1.6 EncryptSHA1()

```
string Combu.CombuEncryption.EncryptSHA1 (  
    string inputString )
```

Encrypts a string in SHA1 hash.

Returns

The SHA1 encrypted.

Parameters

<i>inputString</i>	Input string.
--------------------	---------------

27.2.1.7 LoadRSA() [1/2]

```
void Combu.CombuEncryption.LoadRSA (
```

```
string sessionToken,
string xml )
```

Sets the session token and loads the RSA data from XML (Public Key).

Parameters

<i>sessionToken</i>	Session token.
<i>xml</i>	RSA Xml.

27.2.1.8 LoadRSA() [2/2]

```
void Combu.CombuEncryption.LoadRSA (
    string sessionToken,
    string modulus,
    string exponent )
```

Sets the session token and loads the Modulus and Exponent of the RSA Public Key.

Parameters

<i>sessionToken</i>	Session token.
<i>modulus</i>	Modulus.
<i>exponent</i>	Exponent.

27.2.1.9 SetToken()

```
void Combu.CombuEncryption.SetToken (
    string sessionToken )
```

Sets the session token.

Parameters

<i>sessionToken</i>	Session token.
---------------------	----------------

27.3 Combu.CombuForm Class Reference

Public Member Functions

- void [AddField](#) (string fieldName, string value)
Adds a string parameter to the form.
- void [AddBinaryData](#) (string fieldName, byte[] content)

- Adds a binary parameter to the form.*

 - string [GetField](#) (string fieldName)

Gets a string parameter from the form.
- byte [] [GetBinaryField](#) (string fieldName)

Gets a binary parameter from the form.
- WWWForm [GetForm](#) ()

Gets the WWWForm representation to pass to the web services with encrypted data.

Static Public Member Functions

- static implicit **operator WWWForm** ([CombuForm](#) me)

27.3.1 Member Function Documentation

27.3.1.1 AddBinaryData()

```
void Combu.CombuForm.AddBinaryData (
    string fieldName,
    byte [] content )
```

Adds a binary parameter to the form.

Parameters

<i>fieldName</i>	Field name.
<i>content</i>	Content.

27.3.1.2 AddField()

```
void Combu.CombuForm.AddField (
    string fieldName,
    string value )
```

Adds a string parameter to the form.

Parameters

<i>fieldName</i>	Field name.
<i>value</i>	Value.

27.3.1.3 GetBinaryField()

```
byte [] Combu.CombuForm.GetBinaryField (
    string fieldName )
```

Gets a binary parameter from the form.

Returns

The binary field.

Parameters

<i>fieldName</i>	Field name.
------------------	-------------

27.3.1.4 GetField()

```
string Combu.CombuForm.GetField (
    string fieldName )
```

Gets a string parameter from the form.

Returns

The field.

Parameters

<i>fieldName</i>	Field name.
------------------	-------------

27.3.1.5 GetForm()

```
WWWForm Combu.CombuForm.GetForm ( )
```

Gets the WWWForm representation to pass to the web services with encrypted data.

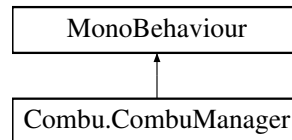
Returns

The form.

27.4 Combu.CombuManager Class Reference

Combu Manager class, it follows the Singleton pattern design (accessible through the 'instance' static property), it means that you shouldn't have more than one instance of this component in your scene.

Inheritance diagram for Combu.CombuManager:



Public Member Functions

- void [CallWebservice](#) (string url, WWWForm form, System.Action< string, string > onComplete)
 - Calls a webservice.*
- void [CancelRequest](#) ()
 - Cancels the current request (next frame).*
- [CombuForm CreateForm](#) ()
 - Creates a new form to be passed to a webservice.*
- string [GetUrl](#) (string relativeUrl)
 - Gets the absolute URL from a relative.*
- virtual void [SetLocalUser](#) ([User](#) user)
 - Sets the local user. For internal use only (e.g. [User.Authenticate](#)), it's not recommended to call this method directly.*
- virtual void [GetServerInfo](#) (Action< bool, [CombuServerInfo](#) > callback)
 - Gets the server info.*
- virtual void [Ping](#) (bool onlyIfAuthenticated=true, Action< bool > callback=null)
 - Ping the server.*

Static Public Member Functions

- static byte [] [CaptureScreenshot](#) (int thumbnailHeight=720, List< Camera > excludeCams=null)
 - Captures the screen shot.*
- static string [EncryptMD5](#) (string inputString)
 - Encrypts a string in MD5.*
- static string [EncryptSHA1](#) (string inputString)
 - Encrypts a string in SHA1.*
- static string [EncryptAES](#) (string inputString)
 - Encrypts a string in AES.*
- static string [DecryptAES](#) (string inputString)
 - Decrypts a string in AES.*

Public Attributes

- const string `COMBU_VERSION` = "3.0.16"
Current API version.
- bool `dontDestroyOnLoad` = true
Should call DontDestroyOnLoad on the `CombuManager` gameObject? Recommended: set to true
- bool `setAsDefaultSocialPlatform`
Should set `Combu` as the active social platform? The previous platform is accessible from `defaultSocialPlatform`
- string `appld`
The app identifier.
- string `appSecret`
The app secret key.
- string `urlRootProduction`
The URL root for the production environment.
- string `urlRootStage`
The URL root for the stage environment.
- bool `useStage`
If true sets the stage as current environment (default: false for production).
- bool `logDebugInfo`
Print debug info in the console log.
- bool `rememberCredentials`
Store the user credentials in PlayerPrefs to auto-login on next connection.
- float `pingIntervalSeconds` = 30f
The ping interval in seconds (set 0 to disable automatic pings). Ping is currently used to maintain the online state of the local user and is automatically called only if the local user is authenticated.
- int `onlineSeconds` = 120
*The max seconds from now to a user's **lastSeen** to consider the online state.*
- int `playingSeconds` = 120
*The max seconds from now to a user's **lastSeen** to consider the playing state.*
- string `language` = "en"
The language to localize the error messages from web services.
- GameObject `achievementUIObject`
The achievement user interface object for `CombuPlatform.ShowAchievementsUI()`.
- string `achievementUIFunction`
The achievement user interface function for `CombuPlatform.ShowAchievementsUI()`.
- GameObject `leaderboardUIObject`
The leaderboard user interface object for `CombuPlatform.ShowLeaderboardUI()`.
- string `leaderboardUIFunction`
The leaderboard user interface function for `CombuPlatform.ShowLeaderboardUI()`.

Protected Member Functions

- virtual void `Awake` ()
- virtual void `Start` ()
- virtual void `OnEnable` ()
- virtual void `OnDisable` ()
- virtual void `AutoPing` ()
This is InvokeRepeated on enable and automatically pings the server. If the user was authenticated and the ping fails, the local user is automatically disconnected.
- IEnumerator `DownloadUrl` (string url, WWWForm form, Action< string, string > onComplete)
Downloads the content of an URL with the specified form.

Protected Attributes

- ISocialPlatform **_defaultSocialPlatform**
- [CombuServerInfo](#) **_serverInfo**
- bool **downloading**

Static Protected Attributes

- static [CombuManager](#) **_instance**
The singleton instance of [CombuManager](#)
- static [CombuPlatform](#) **_platform**
The singleton instance of [CombuPlatform](#).
- static [CombuEncryption](#) **encryption** = new [CombuEncryption](#)()
The instance of data encryption.

Properties

- static string [sessionToken](#) [get]
Gets the session token.
- static [CombuManager](#) **instance** [get]
Gets the current singleton instance.
- static [CombuPlatform](#) **platform** [get]
Gets the [Combu](#) ISocialPlatform implementation.
- static [User](#) **localUser** [get]
Gets the local user.
- static bool **isInitialized** [get]
Gets a value indicating whether the Singleton instance of [Combu.CombuManager](#) is initialized.
- ISocialPlatform **defaultSocialPlatform** [get]
Gets the default social platform defined (this is set before [Combu](#) is set as activate, eventually).
- bool **isDownloading** [get]
Gets a value indicating whether this [Combu.CombuManager](#) is downloading from a webservice.
- bool **isCancelling** [get]
Gets a value indicating whether this [Combu.CombuManager](#) is cancelling a webservice request.
- bool **isAuthenticated** [get]
Gets a value indicating whether [Combu.CombuManager.localUser](#) is authenticated.
- [CombuServerInfo](#) **serverInfo** [get]
Gets the server info.

27.4.1 Detailed Description

[Combu](#) Manager class, it follows the Singleton pattern design (accessible through the 'instance' static property), it means that you shouldn't have more than one instance of this component in your scene.

27.4.2 Member Function Documentation

27.4.2.1 AutoPing()

```
virtual void Combu.CombuManager.AutoPing ( ) [protected], [virtual]
```

This is InvokeRepeated on enable and automatically pings the server. If the user was authenticated and the ping fails, the local user is automatically disconnected.

27.4.2.2 CallWebservice()

```
void Combu.CombuManager.CallWebservice (
    string url,
    WWWForm form,
    System.Action< string, string > onComplete )
```

Calls a webservice.

Parameters

<i>url</i>	URL.
<i>form</i>	Form.
<i>onComplete</i>	On complete callback.

27.4.2.3 CancelRequest()

```
void Combu.CombuManager.CancelRequest ( )
```

Cancels the current request (next frame).

27.4.2.4 CaptureScreenshot()

```
static byte [ ] Combu.CombuManager.CaptureScreenshot (
    int thumbnailHeight = 720,
    List< Camera > excludeCams = null ) [static]
```

Captures the screen shot.

Returns

The screen shot.

Parameters

<i>thumbnailHeight</i>	Thumbnail height.
<i>excludeCams</i>	Exclude cams.

27.4.2.5 CreateForm()

`CombuForm` `Combu.CombuManager.CreateForm ()`

Creates a new form to be passed to a webservice.

Returns

The form.

27.4.2.6 DecryptAES()

```
static string Combu.CombuManager.DecryptAES (
    string inputString ) [static]
```

Decrypts a string in AES.

Returns

The decrypted string.

Parameters

<i>inputString</i>	Input string.
--------------------	---------------

27.4.2.7 DownloadUrl()

```
IEnumerator Combu.CombuManager.DownloadUrl (
    string url,
    WWWForm form,
    Action< string, string > onComplete ) [protected]
```

Downloads the content of an URL with the specified form.

Returns

The URL.

Parameters

<i>url</i>	URL.
<i>form</i>	Form.
<i>onComplete</i>	On complete.

27.4.2.8 EncryptAES()

```
static string Combu.CombuManager.EncryptAES (  
    string inputString ) [static]
```

Encrypts a string in AES.

Returns

The encrypted string.

Parameters

<i>inputString</i>	Input string.
--------------------	---------------

27.4.2.9 EncryptMD5()

```
static string Combu.CombuManager.EncryptMD5 (  
    string inputString ) [static]
```

Encrypts a string in MD5.

Returns

The encrypted string.

Parameters

<i>inputString</i>	Input string.
--------------------	---------------

27.4.2.10 EncryptSHA1()

```
static string Combu.CombuManager.EncryptSHA1 (  
    string inputString ) [static]
```

Encrypts a string in SHA1.

Returns

The encrypted string.

Parameters

<i>inputString</i>	Input string.
--------------------	---------------

27.4.2.11 GetServerInfo()

```
virtual void Combu.CombuManager.GetServerInfo (
    Action< bool, CombuServerInfo > callback ) [virtual]
```

Gets the server info.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.4.2.12 GetUrl()

```
string Combu.CombuManager.GetUrl (
    string relativeUrl )
```

Gets the absolute URL from a relative.

Returns

The URL.

Parameters

<i>relativeUrl</i>	Relative URL.
--------------------	---------------

27.4.2.13 Ping()

```
virtual void Combu.CombuManager.Ping (
    bool onlyIfAuthenticated = true,
    Action< bool > callback = null ) [virtual]
```

Ping the server.

Parameters

<i>onlyIfAuthenticated</i>	If set to <code>true</code> then it runs only if local user is authenticated.
<i>callback</i>	Callback.

27.4.2.14 SetLocalUser()

```
virtual void Combu.CombuManager.SetLocalUser (  
    User user ) [virtual]
```

Sets the local user. For internal use only (e.g. [User.Authenticate](#)), it's not recommended to call this method directly.

Parameters

<i>user</i>	User.
-------------	-----------------------

27.4.3 Member Data Documentation

27.4.3.1 _instance

```
CombuManager Combu.CombuManager._instance [static], [protected]
```

The singleton instance of [CombuManager](#)

27.4.3.2 _platform

```
CombuPlatform Combu.CombuManager._platform [static], [protected]
```

The singleton instance of [CombuPlatform](#).

27.4.3.3 achievementUIFunction

```
string Combu.CombuManager.achievementUIFunction
```

The achievement user interface function for [CombuPlatform.ShowAchievementsUI\(\)](#).

27.4.3.4 achievementUIObject

```
GameObject Combu.CombuManager.achievementUIObject
```

The achievement user interface object for [CombuPlatform.ShowAchievementsUI\(\)](#).

27.4.3.5 appId

```
string Combu.CombuManager.appId
```

The app identifier.

27.4.3.6 appSecret

```
string Combu.CombuManager.appSecret
```

The app secret key.

27.4.3.7 COMBU_VERSION

```
const string Combu.CombuManager.COMBU_VERSION = "3.0.16"
```

Current API version.

27.4.3.8 dontDestroyOnLoad

```
bool Combu.CombuManager.dontDestroyOnLoad = true
```

Should call DontDestroyOnLoad on the [CombuManager](#) gameObject? Recommended: set to true

27.4.3.9 encryption

```
CombuEncryption Combu.CombuManager.encryption = new CombuEncryption() [static], [protected]
```

The instance of data encryption.

27.4.3.10 language

```
string Combu.CombuManager.language = "en"
```

The language to localize the error messages from web services.

27.4.3.11 leaderboardUIFunction

```
string Combu.CombuManager.leaderboardUIFunction
```

The leaderboard user interface function for [CombuPlatform.ShowLeaderboardUI\(\)](#).

27.4.3.12 leaderboardUIObject

```
GameObject Combu.CombuManager.leaderboardUIObject
```

The leaderboard user interface object for [CombuPlatform.ShowLeaderboardUI\(\)](#).

27.4.3.13 logDebugInfo

```
bool Combu.CombuManager.logDebugInfo
```

Print debug info in the console log.

27.4.3.14 onlineSeconds

```
int Combu.CombuManager.onlineSeconds = 120
```

The max seconds from now to a user's **lastSeen** to consider the online state.

27.4.3.15 pingIntervalSeconds

```
float Combu.CombuManager.pingIntervalSeconds = 30f
```

The ping interval in seconds (set 0 to disable automatic pings). Ping is currently used to maintain the online state of the local user and is automatically called only if the local user is authenticated.

27.4.3.16 playingSeconds

```
int Combu.CombuManager.playingSeconds = 120
```

The max seconds from now to a user's **lastSeen** to consider the playing state.

27.4.3.17 rememberCredentials

```
bool Combu.CombuManager.rememberCredentials
```

Store the user credentials in PlayerPrefs to auto-login on next connection.

27.4.3.18 setAsDefaultSocialPlatform

```
bool Combu.CombuManager.setAsDefaultSocialPlatform
```

Should set [Combu](#) as the active social platform? The previous platform is accessible from defaultSocialPlatform

27.4.3.19 urlRootProduction

```
string Combu.CombuManager.urlRootProduction
```

The URL root for the production environment.

27.4.3.20 urlRootStage

```
string Combu.CombuManager.urlRootStage
```

The URL root for the stage environment.

27.4.3.21 useStage

```
bool Combu.CombuManager.useStage
```

If *true* sets the stage as current environment (default: false for production).

27.4.4 Property Documentation

27.4.4.1 defaultSocialPlatform

```
ISocialPlatform Combu.CombuManager.defaultSocialPlatform [get]
```

Gets the default social platform defined (this is set before [Combu](#) is set as activate, eventually).

The default social platform.

27.4.4.2 instance

`CombuManager` `Combu.CombuManager.instance` [static], [get]

Gets the current singleton instance.

The instance.

27.4.4.3 isAuthenticated

`bool` `Combu.CombuManager.isAuthenticated` [get]

Gets a value indicating whether `Combu.CombuManager.localUser` is authenticated.

`true` if is authenticated; otherwise, `false`.

27.4.4.4 isCancelling

`bool` `Combu.CombuManager.isCancelling` [get]

Gets a value indicating whether this `Combu.CombuManager` is cancelling a webservice request.

`true` if is cancelling; otherwise, `false`.

27.4.4.5 isDownloading

`bool` `Combu.CombuManager.isDownloading` [get]

Gets a value indicating whether this `Combu.CombuManager` is downloading from a webservice.

`true` if is downloading; otherwise, `false`.

27.4.4.6 isInitialized

`bool` `Combu.CombuManager.isInitialized` [static], [get]

Gets a value indicating whether the Singleton instance of `Combu.CombuManager` is initialized.

`true` if is initialized; otherwise, `false`.

27.4.4.7 localUser

`User` `Combu.CombuManager.localUser` [static], [get]

Gets the local user.

The local user.

27.4.4.8 platform

`CombuPlatform` `Combu.CombuManager.platform` [static], [get]

Gets the [Combu](#) `ISocialPlatform` implementation.

The platform.

27.4.4.9 serverInfo

`CombuServerInfo` `Combu.CombuManager.serverInfo` [get]

Gets the server info.

The server info.

27.4.4.10 sessionToken

`string` `Combu.CombuManager.sessionToken` [static], [get]

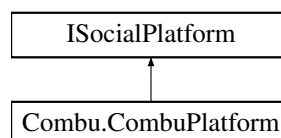
Gets the session token.

The session token.

27.5 Combu.CombuPlatform Class Reference

[Combu](#) Platform implementation of Unity built-in Social interfaces (`ISocialPlatform`).

Inheritance diagram for `Combu.CombuPlatform`:



Public Member Functions

- virtual void [Authenticate](#) (ILocalUser user, System.Action< bool > callback)
Authenticates the user.
- virtual void [Authenticate](#) (string username, string password, System.Action< bool, string > callback)
Authenticates the user with specified username and password.
- virtual void [Authenticate< T >](#) (string username, string password, System.Action< bool, string > callback)
Authenticates the user with specified username and password using the specified profile class.
- virtual void [Authenticate](#) (ILocalUser user, System.Action< bool, string > callback)
Authenticate the specified user.
- virtual void [LoadUsers](#) (string[] userIDs, System.Action< IUserProfile[]> callback)
Loads the users by Id.
- virtual void [ReportProgress](#) (string achievementId, double progress, System.Action< bool > callback)
Reports the progress of an [Achievement](#) expressed as percentage. The progress will be multiplied by 100.0 and finally rounded to int.
- virtual void [ReportProgress](#) (string achievementId, int progress, System.Action< bool > callback)
Reports the progress of an [Achievement](#).
- virtual void [LoadAchievementDescriptions](#) (System.Action< IAchievementDescription[]> callback)
Loads the achievement descriptions.
- virtual void [LoadAchievements](#) (System.Action< IAchievement[]> callback)
Loads the achievements.
- virtual void [LoadAchievements< T >](#) (System.Action< T[]> callback)
Loads the achievements.
- virtual IAchievement [CreateAchievement](#) ()
Creates the achievement.
- virtual void [ReportScore](#) (long score, string board, System.Action< bool > callback)
Reports the score of a [Leaderboard](#).
- virtual void [ReportScore](#) (string score, string board, System.Action< bool > callback)
Reports the score of a [Leaderboard](#).
- virtual void [ReportScore](#) (string score, string board, string username, System.Action< bool > callback)
Reports the score of a [Leaderboard](#).
- virtual void [LoadScores](#) (string leaderboardID, System.Action< IScore[]> callback)
Loads the scores of a [Leaderboard](#).
- virtual void [LoadScores](#) (string leaderboardID, int page, int countPerPage, System.Action< IScore[]> callback)
Loads the scores of a [Leaderboard](#).
- void [LoadScores](#) (string leaderboardID, TimeScope timeScope, int page, int countPerPage, System.Action< IScore[]> callback)
Loads the scores of a [Leaderboard](#).
- virtual ILeaderboard [CreateLeaderboard](#) ()
Creates the leaderboard.
- virtual void [ShowAchievementsUI](#) ()
Shows the achievements UI. Requires achievementUIObject and eventually achievementUIFunction set in order to work.
- virtual void [ShowLeaderboardUI](#) ()
Shows the leaderboard UI. Requires leaderboardUIObject and eventually leaderboardUIFunction set in order to work.
- virtual void [LoadFriends](#) (ILocalUser user, System.Action< bool > callback)
Loads the friends of localUser.
- virtual void [LoadScores](#) (ILeaderboard board, System.Action< bool > callback)
Loads the scores of a [Leaderboard](#).
- virtual bool [GetLoading](#) (ILeaderboard board)
Gets the loading state of a [Leaderboard](#).

- virtual void [SetLocalUser](#) ([User](#) user)
Sets the local user. For internal use only (e.g. [User.Authenticate](#)), it's not recommended to call this method directly.
- virtual void [Logout](#) (System.Action callback)
Logout localUser.
- virtual void [LoadScoresByUser](#) (string leaderboardId, [User](#) user, [eLeaderboardInterval](#) interval, int limit, System.Action< [Score](#), int, string > callback)
Loads the scores of a [Leaderboard](#) by user.

Static Public Member Functions

- static void [ResetAllAchievements](#) (System.Action< bool > callback)
Resets all achievements of localUser.

Properties

- ILocalUser **localUser** [get]

27.5.1 Detailed Description

[Combu](#) Platform implementation of Unity built-in Social interfaces (ISocialPlatform).

27.5.2 Member Function Documentation

27.5.2.1 Authenticate() [1/3]

```
virtual void Combu.CombuPlatform.Authenticate (
    ILocalUser user,
    System.Action< bool > callback ) [virtual]
```

Authenticates the user.

Parameters

<i>user</i>	User .
<i>callback</i>	Callback.

Example of usage:

```
CombuManager.platform.Authenticate (user, (bool success) => {
    Debug.Log("Authenticate: " + success);
});
```

27.5.2.2 Authenticate() [2/3]

```
virtual void Combu.CombuPlatform.Authenticate (
    string username,
    string password,
    System.Action< bool, string > callback ) [virtual]
```

Authenticates the user with specified username and password.

Parameters

<i>username</i>	Username.
<i>password</i>	Password.
<i>callback</i>	Callback.

Example of usage:

```
CombuManager.platform.Authenticate ("username", "password", (bool success, string error) => {
    Debug.Log("Authenticate: " + success + " -- " + error);
});
```

27.5.2.3 Authenticate() [3/3]

```
virtual void Combu.CombuPlatform.Authenticate (
    ILocalUser user,
    System.Action< bool, string > callback ) [virtual]
```

Authenticate the specified user.

Parameters

<i>user</i>	User.
<i>callback</i>	Callback.

Example of usage:

```
CombuManager.platform.Authenticate (user, (bool success, string error) => {
    Debug.Log("Authenticate: " + success + " -- " + error);
});
```

27.5.2.4 Authenticate< T >()

```
virtual void Combu.CombuPlatform.Authenticate< T > (
    string username,
    string password,
    System.Action< bool, string > callback ) [virtual]
```

Authenticates the user with specified username and password using the specified profile class.

Parameters

<i>username</i>	Username.
<i>password</i>	Password.
<i>callback</i>	Callback.

Template Parameters

<i>T</i>	Type of the returned profiles.
----------	--------------------------------

Example of usage:

```
CombuManager.platform.Authenticate<MyUserClass> ("username", "password", (bool success, string error) => {
    Debug.Log("Authenticate: " + success + " -- " + error);
});
```

Type Constraints

T* : *User

T* : *new()

27.5.2.5 CreateAchievement()

```
virtual IAchievement Combu.CombuPlatform.CreateAchievement ( ) [virtual]
```

Creates the achievement.

Returns

The achievement.

Example of usage:

```
IAchievement achievement = CombuManager.platform.CreateAchievement();
```

27.5.2.6 CreateLeaderboard()

```
virtual ILeaderboard Combu.CombuPlatform.CreateLeaderboard ( ) [virtual]
```

Creates the leaderboard.

Returns

The leaderboard.

Example of usage:

```
ILeaderboard leaderboard = CombuManager.platform.CreateLeaderboard();
```


27.5.2.7 GetLoading()

```
virtual bool Combu.CombuPlatform.GetLoading (
    ILeaderboard board ) [virtual]
```

Gets the loading state of a [Leaderboard](#).

Returns

true, if loading was gotten, false otherwise.

Parameters

<i>board</i>	Board.
--------------	--------

Example of usage:

```
Debug.Log("LoadFriends: " + CombuManager.platform.GetLoading(myLeaderboard));
```

27.5.2.8 LoadAchievementDescriptions()

```
virtual void Combu.CombuPlatform.LoadAchievementDescriptions (
    System.Action< IAchievementDescription[]> callback ) [virtual]
```

Loads the achievement descriptions.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

Example of usage:

```
CombuManager.platform.LoadAchievementDescriptions ((IAchievementDescription[] descriptions) => {
    Debug.Log("Achievement descriptions: " + descriptions.Length);
});
```

27.5.2.9 LoadAchievements()

```
virtual void Combu.CombuPlatform.LoadAchievements (
    System.Action< IAchievement[]> callback ) [virtual]
```

Loads the achievements.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

Example of usage:

```
CombuManager.platform.LoadAchievements ((IAchievement[] achievements) => {
    Debug.Log("Achievements: " + achievements.Length);
});
```

27.5.2.10 LoadAchievements< T >()

```
virtual void Combu.CombuPlatform.LoadAchievements< T > (
    System.Action< T[]> callback ) [virtual]
```

Loads the achievements.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

Template Parameters

<i>T</i>	The 1st type parameter.
----------	-------------------------

Example of usage:

```
CombuManager.platform.LoadAchievements<MyAchievementClass> ((MyAchievementClass[] achievements) => {
    Debug.Log("Achievements: " + achievements.Length);
});
```

Type Constraints

***T*: Achievement**

***T*: new()**

27.5.2.11 LoadFriends()

```
virtual void Combu.CombuPlatform.LoadFriends (
    ILocalUser user,
    System.Action< bool > callback ) [virtual]
```

Loads the friends of localUser.

Parameters

<i>user</i>	User.
<i>callback</i>	Callback.

Example of usage:

```
CombuManager.platform.LoadFriends (CombuManager.localUser, (bool success) => {
    Debug.Log("LoadFriends: " + success);
});
```

27.5.2.12 LoadScores() [1/4]

```
virtual void Combu.CombuPlatform.LoadScores (
    string leaderboardID,
    System.Action< IScore[]> callback ) [virtual]
```

Loads the scores of a [Leaderboard](#).

Parameters

<i>leaderboardID</i>	Leaderboard I.
<i>callback</i>	Callback.

Example of usage:

```
CombuManager.platform.LoadScores ("1234", (IScore[] scores) => {
    Debug.Log("Scores: " + scores.Length);
});
```

27.5.2.13 LoadScores() [2/4]

```
virtual void Combu.CombuPlatform.LoadScores (
    string leaderboardID,
    int page,
    int countPerPage,
    System.Action< IScore[]> callback ) [virtual]
```

Loads the scores of a [Leaderboard](#).

Parameters

<i>leaderboardID</i>	Leaderboard I.
<i>page</i>	Page.
<i>countPerPage</i>	Count per page.
<i>callback</i>	Callback.

Example of usage:

```
CombuManager.platform.LoadScores ("1234", 1, 10, (IScore[] scores) => {
    Debug.Log("Scores: " + scores.Length);
});
```

27.5.2.14 LoadScores() [3/4]

```
void Combu.CombuPlatform.LoadScores (
    string leaderboardID,
    TimeScope timeScope,
    int page,
    int countPerPage,
    System.Action< IScore[]> callback )
```

Loads the scores of a [Leaderboard](#).

Parameters

<i>leaderboardID</i>	Leaderboard identifier.
<i>timeScope</i>	Time scope.
<i>page</i>	Page.
<i>countPerPage</i>	Count per page.
<i>callback</i>	Callback.

27.5.2.15 LoadScores() [4/4]

```
virtual void Combu.CombuPlatform.LoadScores (
    ILeaderboard board,
    System.Action< bool > callback ) [virtual]
```

Loads the scores of a [Leaderboard](#).

Parameters

<i>board</i>	Board.
<i>callback</i>	Callback.

Example of usage:

```
Leaderboard myLeaderboard = CombuManager.platform.CreateLeaderboard();
myLeaderboard.id = "123";
myLeaderboard.timeScope = UnityEngine.SocialPlatforms.TimeScope.AllTime;
myLeaderboard.range = new UnityEngine.SocialPlatforms.Range(1, 10);
CombuManager.platform.LoadScores (myLeaderboard, (bool success) => {
    Debug.Log("LoadScores: " + success);
});
```

27.5.2.16 LoadScoresByUser()

```
virtual void Combu.CombuPlatform.LoadScoresByUser (
    string leaderboardId,
    User user,
    eLeaderboardInterval interval,
    int limit,
    System.Action< Score, int, string > callback ) [virtual]
```

Loads the scores of a [Leaderboard](#) by user.

Parameters

<i>leaderboardId</i>	Leaderboard identifier.
<i>user</i>	User .
<i>interval</i>	Interval.
<i>callback</i>	Callback.

Example of usage:

```
CombuManager.platform.LoadScores ("1234", CombuManager.localUser,
    eLeaderboardInterval.Week, 10, (Score score, int page, string error) => {
    Debug.Log("Score: " + score.rank);
});
```

27.5.2.17 LoadUsers()

```
virtual void Combu.CombuPlatform.LoadUsers (
    string [] userIDs,
    System.Action< IUserProfile[]> callback ) [virtual]
```

Loads the users by Id.

Parameters

<i>userIDs</i>	User Ids.
<i>callback</i>	Callback.

Example of usage:

```
CombuManager.platform.LoadUsers (new string[] {1234, 5678, 9012}, (IUserProfile[] users) => {
    Debug.Log("Users: " + users.Length);
});
```

27.5.2.18 Logout()

```
virtual void Combu.CombuPlatform.Logout (
    System.Action callback ) [virtual]
```

Logout localUser.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

Example of usage:

```
CombuManager.platform.Logout();
```

27.5.2.19 ReportProgress() [1/2]

```
virtual void Combu.CombuPlatform.ReportProgress (
    string achievementId,
    double progress,
    System.Action< bool > callback ) [virtual]
```

Reports the progress of an [Achievement](#) expressed as percentage. The progress will be multiplied by 100.0 and finally rounded to int.

Parameters

<i>achievementID</i>	Achievement identifier.
<i>progress</i>	Progress.
<i>callback</i>	Callback.

Example of usage:

```
CombuManager.platform.ReportProgress ("1234", 1.0, (bool success) => {
    Debug.Log("ReportProgress: " + success);
});
```

27.5.2.20 ReportProgress() [2/2]

```
virtual void Combu.CombuPlatform.ReportProgress (
    string achievementId,
    int progress,
    System.Action< bool > callback ) [virtual]
```

Reports the progress of an [Achievement](#).

Parameters

<i>achievementId</i>	Achievement identifier.
<i>progress</i>	Progress.
<i>callback</i>	Callback.

Example of usage:

```
CombuManager.platform.ReportProgress ("1234", 100, (bool success) => {
    Debug.Log("ReportProgress: " + success);
});
```

27.5.2.21 ReportScore() [1/3]

```
virtual void Combu.CombuPlatform.ReportScore (
    long score,
    string board,
    System.Action< bool > callback ) [virtual]
```

Reports the score of a [Leaderboard](#).

Parameters

<i>score</i>	Score .
<i>board</i>	Board.
<i>callback</i>	Callback.

Example of usage:

```
CombuManager.platform.ReportScore (1000, "1234", (bool success) => {
    Debug.Log("ReportScore: " + success);
});
```

27.5.2.22 ReportScore() [2/3]

```
virtual void Combu.CombuPlatform.ReportScore (
    string score,
    string board,
    System.Action< bool > callback ) [virtual]
```

Reports the score of a [Leaderboard](#).

Parameters

<i>score</i>	Score .
<i>board</i>	Board.
<i>callback</i>	Callback.

Example of usage:

```
CombuManager.platform.ReportScore ("1000", "1234", (bool success) => {
    Debug.Log("ReportScore: " + success);
});
```

27.5.2.23 ReportScore() [3/3]

```
virtual void Combu.CombuPlatform.ReportScore (
    string score,
    string board,
    string username,
    System.Action< bool > callback ) [virtual]
```

Reports the score of a [Leaderboard](#).

Parameters

<i>score</i>	Score.
<i>board</i>	Board.
<i>username</i>	Username.
<i>callback</i>	Callback.

Example of usage:

```
CombuManager.platform.ReportScore ("1000", "1234", "username", (bool success) => {
    Debug.Log("ReportScore: " + success);
});
```

27.5.2.24 ResetAllAchievements()

```
static void Combu.CombuPlatform.ResetAllAchievements (
    System.Action< bool > callback ) [static]
```

Resets all achievements of localUser.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.5.2.25 SetLocalUser()

```
virtual void Combu.CombuPlatform.SetLocalUser (
    User user ) [virtual]
```


Sets the local user. For internal use only (e.g. [User.Authenticate](#)), it's not recommended to call this method directly.

Parameters

<i>user</i>	User.
-------------	-----------------------

27.5.2.26 ShowAchievementsUI()

```
virtual void Combu.CombuPlatform.ShowAchievementsUI ( ) [virtual]
```

Shows the achievements UI. Requires achievementUIObject and eventually achievementUIFunction set in order to work.

Example of usage:

```
CombuManager.platform.ShowAchievementsUI();
```

27.5.2.27 ShowLeaderboardUI()

```
virtual void Combu.CombuPlatform.ShowLeaderboardUI ( ) [virtual]
```

Shows the leaderboard UI. Requires leaderboardUIObject and eventually leaderboardUIFunction set in order to work.

Example of usage:

```
CombuManager.platform.ShowAchievementsUI();
```

27.6 Combu.CombuServerInfo Class Reference

Class to handle [Combu](#) server informations.

Public Member Functions

- **CombuServerInfo** (Hashtable data)
- override string [ToString](#) ()
Returns a T:System.String that represents the current T:Combu.CombuServerInfo.

Public Attributes

- string **version** = string.Empty
- bool **requireUpdate**
- DateTime **time** = DateTime.MinValue
- Hashtable **settings** = new Hashtable()
- bool **responseEncrypted**

27.6.1 Detailed Description

Class to handle [Combu](#) server informations.

27.6.2 Member Function Documentation

27.6.2.1 ToString()

```
override string Combu.CombuServerInfo.ToString ( )
```

Returns a `T:System.String` that represents the current `T:Combu.CombuServerInfo`.

Returns

A `T:System.String` that represents the current `T:Combu.CombuServerInfo`.

27.7 Combu.Inventory Class Reference

Public Member Functions

- [Inventory](#) ()
Initializes a new instance of the [Inventory](#) class.
- [Inventory](#) (string jsonString)
Initializes a new instance of the [Inventory](#) class from a JSON formatted string.
- [Inventory](#) (Hashtable hash)
Initializes a new instance of the [Inventory](#) class from a Hashtable.
- virtual void [FromJson](#) (string jsonString)
Initialize the object from a JSON formatted string.
- virtual void [FromHashtable](#) (Hashtable hash)
Initialize the object from a hashtable.
- virtual void [Update](#) (System.Action< bool, string > callback)
Update this inventory item to server.
- virtual void [Delete](#) (System.Action< bool, string > callback)
Delete this inventory item from server.

Static Public Member Functions

- static void [Load](#) (string userId, System.Action< [Inventory](#)[], string > callback)
Load the inventory items of a [User](#).
- static void [Load](#)< T > (string userId, System.Action< T[], string > callback)
Load the inventory of a [User](#).
- static void [Delete](#) (long idInventory, System.Action< bool, string > callback)
Delete the specified inventory item from server.

Public Attributes

- string **name** = ""
- int **quantity** = 0
- Hashtable **customData** = new Hashtable()

Properties

- long **id** [get]

27.7.1 Constructor & Destructor Documentation

27.7.1.1 Inventory() [1/3]

```
Combu.Inventory.Inventory ( )
```

Initializes a new instance of the [Inventory](#) class.

27.7.1.2 Inventory() [2/3]

```
Combu.Inventory.Inventory (
    string jsonString )
```

Initializes a new instance of the [Inventory](#) class from a JSON formatted string.

Parameters

<i>jsonString</i>	JSON formatted string.
-------------------	------------------------

27.7.1.3 Inventory() [3/3]

```
Combu.Inventory.Inventory (
    Hashtable hash )
```

Initializes a new instance of the [Inventory](#) class from a Hashtable.

Parameters

<i>hash</i>	Hash.
-------------	-------

27.7.2 Member Function Documentation

27.7.2.1 Delete() [1/2]

```
virtual void Combu.Inventory.Delete (
    System.Action< bool, string > callback ) [virtual]
```

Delete this inventory item from server.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.7.2.2 Delete() [2/2]

```
static void Combu.Inventory.Delete (
    long idInventory,
    System.Action< bool, string > callback ) [static]
```

Delete the specified inventory item from server.

Parameters

<i>idInventory</i>	Identifier inventory.
<i>callback</i>	Callback.

27.7.2.3 FromHashtable()

```
virtual void Combu.Inventory.FromHashtable (
    Hashtable hash ) [virtual]
```

Initialize the object from a hashtable.

Parameters

<i>hash</i>	Hash.
-------------	-------

27.7.2.4 FromJson()

```
virtual void Combu.Inventory.FromJson (
```

```
string jsonString ) [virtual]
```

Initialize the object from a JSON formatted string.

Parameters

<i>jsonString</i>	Json string.
-------------------	--------------

27.7.2.5 Load()

```
static void Combu.Inventory.Load (
    string userId,
    System.Action< Inventory[], string > callback ) [static]
```

Load the inventory items of a [User](#).

Parameters

<i>userId</i>	User identifier.
<i>callback</i>	Callback.

27.7.2.6 Load< T >()

```
static void Combu.Inventory.Load< T > (
    string userId,
    System.Action< T[], string > callback ) [static]
```

Load the inventory of a [User](#).

Parameters

<i>userId</i>	User identifier.
<i>callback</i>	Callback.

Template Parameters

<i>T</i>	The 1st type parameter.
----------	-------------------------

Type Constraints

***T* : [Inventory](#)**

***T* : [new\(\)](#)**

27.7.2.7 Update()

```
virtual void Combu.Inventory.Update (
    System.Action< bool, string > callback ) [virtual]
```

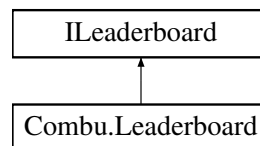
Update this inventory item to server.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.8 Combu.Leaderboard Class Reference

Inheritance diagram for Combu.Leaderboard:



Public Member Functions

- **Leaderboard** (string jsonString)
- virtual void **FromJson** (string jsonString)
Initialize the object from a JSON formatted string.
- void **SetUserFilter** (string[] userIDs)
Sets the user filter.
- void **LoadScores** (System.Action< bool > callback)
Loads the scores.
- virtual void **LoadScoresByUser** (User user, eLeaderboardInterval interval, int limit, System.Action< Score, int, string > callback)
Loads the scores of a user on this leaderboard.
- virtual void **LoadScoresByUser** (long userId, eLeaderboardInterval interval, int limit, System.Action< Score, int, string > callback)
Loads the scores by user id.
- virtual void **LoadScoresByUser** (string userName, eLeaderboardInterval interval, int limit, System.Action< Score, int, string > callback)
Loads the scores by userName.

Static Public Member Functions

- static void **Load** (string leaderboardId, System.Action< Leaderboard, string > callback)
Load the specified leaderboardId.
- static void **Load< T >** (string leaderboardId, System.Action< Leaderboard, string > callback)
Load the specified leaderboardId.
- static void **LoadScoresByUser** (User user, eLeaderboardInterval interval, System.Action< Score[], string > callback)
Loads the scores of a user.

Public Attributes

- bool **highestScorePerPlayer**
- bool **sumScoresPerPlayer**

Protected Member Functions

- void [LoadScoresByUser](#) (long userId, string userName, [eLeaderboardInterval](#) interval, int limit, System.Action< [Score](#), int, string > callback)
Loads the scores by user.

Properties

- bool **loading** [get]
- string **id** [get, set]
- string **code** [get, set]
- UserScope **userScope** [get, set]
- Range **range** [get, set]
- TimeScope **timeScope** [get, set]
- [eLeaderboardTimeScope](#) **customTimescope** [get, set]
- IScore **localUserScore** [get]
- uint **maxRange** [get]
- IScore [] **scores** [get]
- string **title** [get]
- string **description** [get]

27.8.1 Member Function Documentation

27.8.1.1 FromJson()

```
virtual void Combu.Leaderboard.FromJson (
    string jsonString ) [virtual]
```

Initialize the object from a JSON formatted string.

Parameters

<i>jsonString</i>	Json string.
-------------------	--------------

27.8.1.2 Load()

```
static void Combu.Leaderboard.Load (
    string leaderboardId,
    System.Action< Leaderboard, string > callback ) [static]
```

Load the specified leaderboardId.

Parameters

<i>leaderboardId</i>	Leaderboard identifier.
<i>callback</i>	Callback.

27.8.1.3 Load< T >()

```
static void Combu.Leaderboard.Load< T > (
    string leaderboardId,
    System.Action< Leaderboard, string > callback ) [static]
```

Load the specified leaderboardId.

Parameters

<i>leaderboardId</i>	Leaderboard identifier.
<i>callback</i>	Callback.

Template Parameters

<i>T</i>	The 1st type parameter.
----------	-------------------------

Type Constraints

T* : *Leaderboard

T* : *new()

27.8.1.4 LoadScores()

```
void Combu.Leaderboard.LoadScores (
    System.Action< bool > callback )
```

Loads the scores.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.8.1.5 LoadScoresByUser() [1/5]

```
virtual void Combu.Leaderboard.LoadScoresByUser (
    User user,
    eLeaderboardInterval interval,
    int limit,
    System.Action< Score, int, string > callback ) [virtual]
```

Loads the scores of a user on this leaderboard.

Parameters

<i>user</i>	User.
<i>interval</i>	Interval.
<i>callback</i>	Callback.

27.8.1.6 LoadScoresByUser() [2/5]

```
virtual void Combu.Leaderboard.LoadScoresByUser (
    long userId,
    eLeaderboardInterval interval,
    int limit,
    System.Action< Score, int, string > callback ) [virtual]
```

Loads the scores by user id.

Parameters

<i>userId</i>	User identifier.
<i>interval</i>	Interval.
<i>limit</i>	Limit.
<i>callback</i>	Callback.

27.8.1.7 LoadScoresByUser() [3/5]

```
virtual void Combu.Leaderboard.LoadScoresByUser (
    string userName,
    eLeaderboardInterval interval,
    int limit,
    System.Action< Score, int, string > callback ) [virtual]
```

Loads the scores by userName.

Parameters

<i>userName</i>	User name.
<i>interval</i>	Interval.
<i>limit</i>	Limit.
<i>callback</i>	Callback.

27.8.1.8 LoadScoresByUser() [4/5]

```
void Combu.Leadersboard.LoadScoresByUser (
    long userId,
    string userName,
    eLeadersboardInterval interval,
    int limit,
    System.Action< Score, int, string > callback ) [protected]
```

Loads the scores by user.

Parameters

<i>userId</i>	User identifier.
<i>userName</i>	User name.
<i>interval</i>	Interval.
<i>limit</i>	Limit.
<i>callback</i>	Callback.

27.8.1.9 LoadScoresByUser() [5/5]

```
static void Combu.Leadersboard.LoadScoresByUser (
    User user,
    eLeadersboardInterval interval,
    System.Action< Score[], string > callback ) [static]
```

Loads the scores of a user.

Parameters

<i>user</i>	User.
<i>interval</i>	Interval.
<i>callback</i>	Callback.

27.8.1.10 SetUserFilter()

```
void Combu.Leadersboard.SetUserFilter (
    string [] userIDs )
```

Sets the user filter.

Parameters

<i>userIDs</i>	User IDs.
----------------	-----------

27.9 Combu.Mail Class Reference

Public Member Functions

- virtual void [FromJson](#) (string jsonString)
Initialize the object from a JSON formatted string.
- virtual void [FromHashtable](#) (Hashtable hash)
Initialize the object from a hashtable.
- void [Read](#) (Action< bool, string > callback)
Mark this mail as read.
- void [Unread](#) (Action< bool, string > callback)
Mark this mail as read.
- virtual void [Delete](#) (Action< bool, string > callback)
Delete this instance.

Static Public Member Functions

- static void [Load](#) (eMailList listType, int pageNumber, int limit, System.Action< [Mail](#)[], int, int, string > callback)
- static void [Load](#) (eMailList listType, long idRecipient, long idSender, long idGroup, int pageNumber, int limit, System.Action< [Mail](#)[], int, int, string > callback)
- static void [Load](#)< T > (eMailList listType, long idRecipient, long idSender, long idGroup, int pageNumber, int limit, System.Action< T[], int, int, string > callback)
Load mails list from specified page and number of records.
- static void [Send](#) (long recipientId, string subject, string message, bool isPublic, System.Action< bool, string > callback)
Sends a mail to a user.
- static void [Send](#) (long[] recipientsId, string subject, string message, bool isPublic, System.Action< bool, string > callback)
Sends the mail to multiple users.
- static void [Send](#) (string recipientUsername, string subject, string message, bool isPublic, System.Action< bool, string > callback)
Sends a mail to a user.
- static void [Send](#) (string[] recipientsUsername, string subject, string message, bool isPublic, System.Action< bool, string > callback)
Sends the mail to multiple users.
- static void [SendMailToGroup](#) (long groupId, string subject, string message, bool isPublic, System.Action< bool, string > callback)
Sends the mail to group.
- static void [Read](#) (long idMail, Action< bool, string > callback)
Mark a single mail as read.
- static void [Read](#) (long[] idSenders, long[] idGroups, Action< bool, string > callback)
Mark a set of mails as read.
- static void [Unread](#) (long idMail, Action< bool, string > callback)
Mark a single mail as read.
- static void [Delete](#) (long idMail, Action< bool, string > callback)
Delete the specified Mail.
- static void [LoadConversations](#) (Action< ArrayList, int, string > callback)
Loads the conversations (list of senders as both users and groups).
- static void [Count](#) (long[] idUsers, long[] idGroups, Action< [MailCount](#)[], string > callback)
Loads the read/unread messages from list of senders and groups.

Public Attributes

- long **id** = 0
- DateTime **sendDate** = DateTime.MinValue
- DateTime **readDate** = DateTime.MinValue
- string **subject** = ""
- string **message** = ""
- bool **isPublic** = false
- [User](#) **fromUser**
- [User](#) **toUser**
- long **idGroup** = 0
- [UserGroup](#) **toGroup**

Static Protected Member Functions

- static void [Send](#) (long[] recipientsId, string[] recipientsUsername, long recipientGroupId, string subject, string message, bool isPublic, System.Action< bool, string > callback)
Sends the mail.
- static void [Read](#) (long idMail, long[] idSenders, long[] idGroups, Action< bool, string > callback)
Mark a single mail or a set of mails as read.

Properties

- bool **isRead** [get]

27.9.1 Member Function Documentation

27.9.1.1 Count()

```
static void Combu.Mail.Count (
    long [] idUsers,
    long [] idGroups,
    Action< MailCount[], string > callback ) [static]
```

Loads the read/unread messages from list of senders and groups.

27.9.1.2 Delete() [1/2]

```
virtual void Combu.Mail.Delete (
    Action< bool, string > callback ) [virtual]
```

Delete this instance.

27.9.1.3 Delete() [2/2]

```
static void Combu.Mail.Delete (
    long idMail,
    Action< bool, string > callback ) [static]
```

Delete the specified [Mail](#).

Parameters

<i>idMail</i>	Identifier mail.
<i>callback</i>	Callback.

27.9.1.4 FromHashtable()

```
virtual void Combu.Mail.FromHashtable (
    Hashtable hash ) [virtual]
```

Initialize the object from a hashtable.

Parameters

<i>hash</i>	Hash.
-------------	-------

27.9.1.5 FromJson()

```
virtual void Combu.Mail.FromJson (
    string jsonString ) [virtual]
```

Initialize the object from a JSON formatted string.

Parameters

<i>jsonString</i>	Json string.
-------------------	--------------

27.9.1.6 Load< T >()

```
static void Combu.Mail.Load< T > (
    eMailList listType,
    long idRecipient,
    long idSender,
    long idGroup,
    int pageNumber,
    int limit,
    System.Action< T[], int, int, string > callback ) [static]
```

Load mails list from specified page and number of records.

Parameters

<i>pageNumber</i>	Page number.
<i>limit</i>	Limit.

Template Parameters

<i>T</i>	Type of returned objects.
----------	---------------------------

Type Constraints

T* : Mail**T* : new()**

27.9.1.7 LoadConversations()

```
static void Combu.Mail.LoadConversations (
    Action< ArrayList, int, string > callback ) [static]
```

Loads the conversations (list of senders as both users and groups).

27.9.1.8 Read() [1/4]

```
void Combu.Mail.Read (
    Action< bool, string > callback )
```

Mark this mail as read.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.9.1.9 Read() [2/4]

```
static void Combu.Mail.Read (
    long idMail,
    Action< bool, string > callback ) [static]
```

Mark a single mail as read.

Parameters

<i>idMail</i>	Identifier mail.
<i>callback</i>	Callback.

27.9.1.10 Read() [3/4]

```
static void Combu.Mail.Read (
    long [] idSenders,
    long [] idGroups,
    Action< bool, string > callback ) [static]
```

Mark a set of mails as read.

Parameters

<i>idSenders</i>	Identifier senders.
<i>idGroups</i>	Identifier groups.

27.9.1.11 Read() [4/4]

```
static void Combu.Mail.Read (
    long idMail,
    long [] idSenders,
    long [] idGroups,
    Action< bool, string > callback ) [static], [protected]
```

Mark a single mail or a set of mails as read.

Parameters

<i>idMail</i>	Identifier mail.
<i>idSenders</i>	Identifier senders.
<i>idGroups</i>	Identifier groups.
<i>callback</i>	Callback.

27.9.1.12 Send() [1/5]

```
static void Combu.Mail.Send (
    long recipientId,
    string subject,
    string message,
    bool isPublic,
    System.Action< bool, string > callback ) [static]
```

Sends a mail to a user.

Parameters

<i>idDestination</i>	Identifier Id of the destination user.
<i>subject</i>	Subject.
<i>message</i>	Message.
<i>isPublic</i>	If set to true is public.

27.9.1.13 Send() [2/5]

```
static void Combu.Mail.Send (
    long [] recipientsId,
    string subject,
    string message,
    bool isPublic,
    System.Action< bool, string > callback ) [static]
```

Sends the mail to multiple users.

Parameters

<i>recipientsId</i>	Recipients identifier.
<i>subject</i>	Subject.
<i>message</i>	Message.
<i>isPublic</i>	If set to true is public.

27.9.1.14 Send() [3/5]

```
static void Combu.Mail.Send (
    string recipientUsername,
    string subject,
    string message,
    bool isPublic,
    System.Action< bool, string > callback ) [static]
```

Sends a mail to a user.

Parameters

<i>usernameDestination</i>	Username of the destination user.
<i>subject</i>	Subject.
<i>message</i>	Message.
<i>isPublic</i>	If set to true is public.

27.9.1.15 Send() [4/5]

```
static void Combu.Mail.Send (
    string [] recipientsUsername,
    string subject,
    string message,
```



```
bool isPublic,
System.Action< bool, string > callback ) [static]
```

Sends the mail to multiple users.

Parameters

<i>recipientsUsername</i>	Recipients username.
<i>subject</i>	Subject.
<i>message</i>	Message.
<i>isPublic</i>	If set to true is public.

27.9.1.16 Send() [5/5]

```
static void Combu.Mail.Send (
    long [] recipientsId,
    string [] recipientsUsername,
    long recipientGroupId,
    string subject,
    string message,
    bool isPublic,
    System.Action< bool, string > callback ) [static], [protected]
```

Sends the mail.

Parameters

<i>recipientsId</i>	Recipients identifier.
<i>recipientsUsername</i>	Recipients username.
<i>recipientGroupId</i>	Recipient group identifier.
<i>subject</i>	Subject.
<i>message</i>	Message.
<i>isPublic</i>	If set to true is public.

27.9.1.17 SendMailToGroup()

```
static void Combu.Mail.SendMailToGroup (
    long groupId,
    string subject,
    string message,
    bool isPublic,
    System.Action< bool, string > callback ) [static]
```

Sends the mail to group.

Parameters

<i>groupid</i>	Group identifier.
<i>subject</i>	Subject.
<i>message</i>	Message.
<i>isPublic</i>	If set to <code>true</code> is public.

27.9.1.18 Unread() [1/2]

```
void Combu.Mail.Unread (
    Action< bool, string > callback )
```

Mark this mail as read.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.9.1.19 Unread() [2/2]

```
static void Combu.Mail.Unread (
    long idMail,
    Action< bool, string > callback ) [static]
```

Mark a single mail as read.

Parameters

<i>idMail</i>	Identifier mail.
<i>callback</i>	Callback.

27.10 Combu.MailCount Class Reference**Public Member Functions**

- **MailCount** (Hashtable hash)

Public Attributes

- long **idSender** = 0
- long **idGroup** = 0
- int **read** = 0
- int **unread** = 0

27.11 Combu.Match Class Reference

Classes

- class [MatchRoundData](#)

Public Member Functions

- [Match](#) (string jsonString)
Initializes a new instance of the [Combu.Match](#) class.
- [Match](#) (Hashtable data)
Initializes a new instance of the [Combu.Match](#) class.
- virtual void [FromJson](#) (string jsonString)
Initialize the object from a JSON formatted string.
- virtual void [FromHashtable](#) (Hashtable hash)
Initialize the object from a hashtable.
- virtual void [AddUser](#) ([Profile](#) user)
Adds the user to this match.
- virtual void [RemoveUser](#) ([Profile](#) user)
Removes the user.
- virtual void [RemoveUser](#) (long idUser)
Removes the user.
- virtual void [RemoveUser](#) (string username)
Removes the user.
- void [Score](#) (float score, Action< bool, string > callback)
Send the specified score.
- void [Save](#) (Action< bool, string > callback)
Save the this instance in the server.
- virtual void [Delete](#) (Action< bool, string > callback)
Delete this instance from the database.

Static Public Member Functions

- static void [Delete](#) (long idMatch, Action< bool, string > callback)
Delete the specified [Match](#).
- static void [QuickMatch](#) (bool friendsOnly, [SearchCustomData](#)[] customData, int rounds, Action< [Match](#) > callback)
Creates a quick match.
- static void [Load](#) (long idTournament, bool activeOnly, string title, Action< [Match](#)[]> callback)
Load the list of [Matches](#) by specified filters.
- static void [Load](#) (long idMatch, Action< [Match](#) > callback)
Load the specified [Match](#).

Public Attributes

- long **id** = 0
- long **idTournament** = 0
- string **title** = ""
- int **roundsCount** = 1
- DateTime **dateCreation** = DateTime.Now
- DateTime **dateExpire** = null
- Hashtable **customData** = new Hashtable()

Protected Member Functions

- virtual void [RemoveUser](#) (long idUser, string username)
Removes the user.

Properties

- List< [MatchAccount](#) > **users** [get]
- List< [MatchRoundData](#) > **rounds** [get]
- bool **finished** [get]
- bool **searchingQuickMatch** [get]

27.11.1 Constructor & Destructor Documentation

27.11.1.1 Match() [1/2]

```
Combu.Match.Match (
    string jsonString )
```

Initializes a new instance of the [Combu.Match](#) class.

Parameters

<i>jsonString</i>	JSON string to initialize the instance.
-------------------	---

27.11.1.2 Match() [2/2]

```
Combu.Match.Match (
    Hashtable data )
```

Initializes a new instance of the [Combu.Match](#) class.

Parameters

<i>data</i>	Data to initialize the instance.
-------------	----------------------------------

27.11.2 Member Function Documentation

27.11.2.1 AddUser()

```
virtual void Combu.Match.AddUser (
    Profile user ) [virtual]
```

Adds the user to this match.

Parameters

<i>user</i>	User.
-------------	-------

27.11.2.2 Delete() [1/2]

```
virtual void Combu.Match.Delete (
    Action< bool, string > callback ) [virtual]
```

Delete this instance from the database.

27.11.2.3 Delete() [2/2]

```
static void Combu.Match.Delete (
    long idMatch,
    Action< bool, string > callback ) [static]
```

Delete the specified Match.

Parameters

<i>idMatch</i>	Identifier match.
<i>callback</i>	Callback.

27.11.2.4 FromHashtable()

```
virtual void Combu.Match.FromHashtable (
    Hashtable hash ) [virtual]
```

Initialize the object from a hashtable.

Parameters

<i>hash</i>	Hash.
-------------	-------

27.11.2.5 FromJson()

```
virtual void Combu.Match.FromJson (
    string jsonString ) [virtual]
```

Initialize the object from a JSON formatted string.

Parameters

<i>jsonString</i>	Json string.
-------------------	--------------

27.11.2.6 Load() [1/2]

```
static void Combu.Match.Load (
    long idTournament,
    bool activeOnly,
    string title,
    Action< Match[]> callback ) [static]
```

Load the list of Matches by specified filters.

Parameters

<i>idTournament</i>	Identifier tournament.
<i>activeOnly</i>	If set to <code>true</code> then displays active matches only, else archived matches.
<i>title</i>	Title.
<i>callback</i>	Callback.

27.11.2.7 Load() [2/2]

```
static void Combu.Match.Load (
    long idMatch,
    Action< Match > callback ) [static]
```

Load the specified [Match](#).

Parameters

<i>idMatch</i>	Identifier match.
<i>callback</i>	Callback.

27.11.2.8 QuickMatch()

```
static void Combu.Match.QuickMatch (
    bool friendsOnly,
    SearchCustomData [] customData,
    int rounds,
    Action< Match > callback ) [static]
```

Creates a quick match.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.11.2.9 RemoveUser() [1/4]

```
virtual void Combu.Match.RemoveUser (
    Profile user ) [virtual]
```

Removes the user.

Parameters

<i>user</i>	User.
-------------	-------

27.11.2.10 RemoveUser() [2/4]

```
virtual void Combu.Match.RemoveUser (
    long idUser ) [virtual]
```

Removes the user.

Parameters

<i>idUser</i>	Identifier user.
---------------	------------------

27.11.2.11 RemoveUser() [3/4]

```
virtual void Combu.Match.RemoveUser (
    string username ) [virtual]
```

Removes the user.

Parameters

<i>username</i>	Username.
-----------------	-----------

27.11.2.12 RemoveUser() [4/4]

```
virtual void Combu.Match.RemoveUser (
    long idUser,
    string username ) [protected], [virtual]
```

Removes the user.

Parameters

<i>idUser</i>	Identifier user.
<i>username</i>	Username.

27.11.2.13 Save()

```
void Combu.Match.Save (
    Action< bool, string > callback )
```

Save the this instance in the server.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.11.2.14 Score()

```
void Combu.Match.Score (
    float score,
    Action< bool, string > callback )
```

Send the specified score.

Parameters

<i>score</i>	Score.
<i>callback</i>	Callback.

27.12 Combu.MatchAccount Class Reference

Public Member Functions

- **MatchAccount** (string jsonString)
- **MatchAccount** (Hashtable data)
- virtual void **FromJson** (string jsonString)
Initialize the object from a JSON formatted string.
- virtual void **FromHashtable** (Hashtable hash)
Initialize the object from a hashtable.

Public Attributes

- long **id** = 0
- long **idMatch** = 0
- long **idAccount** = 0
- Hashtable **customData** = new Hashtable()
- float **score** = 0
- DateTime **dateScore** = null
- Profile **user** = null

Properties

- List< **MatchRound** > **rounds** [get]

27.12.1 Member Function Documentation

27.12.1.1 FromHashtable()

```
virtual void Combu.MatchAccount.FromHashtable (  
    Hashtable hash ) [virtual]
```

Initialize the object from a hashtable.

Parameters

<i>hash</i>	Hash.
-------------	-------

27.12.1.2 FromJson()

```
virtual void Combu.MatchAccount.FromJson (  
    string jsonString ) [virtual]
```

Initialize the object from a JSON formatted string.

Parameters

<code>jsonString</code>	Json string.
-------------------------	--------------

27.13 Combu.MatchRound Class Reference

Public Member Functions

- **MatchRound** (string jsonString)
- **MatchRound** (Hashtable data)
- virtual void **FromJson** (string jsonString)
Initialize the object from a JSON formatted string.
- virtual void **FromHashtable** (Hashtable hash)
Initialize the object from a hashtable.

Public Attributes

- long **id** = 0
- long **idMatchAccount** = 0
- float **score** = 0
- DateTime **dateScore** = null

Properties

- bool **hasScore** [get]

27.13.1 Member Function Documentation

27.13.1.1 FromHashtable()

```
virtual void Combu.MatchRound.FromHashtable (  
    Hashtable hash ) [virtual]
```

Initialize the object from a hashtable.

Parameters

<code>hash</code>	Hash.
-------------------	-------

27.13.1.2 FromJson()

```
virtual void Combu.MatchRound.FromJson (
    string jsonString ) [virtual]
```

Initialize the object from a JSON formatted string.

Parameters

<code>jsonString</code>	Json string.
-------------------------	--------------

27.14 Combu.Match.MatchRoundData Class Reference

Public Attributes

- List< [MatchAccount](#) > **users** = new List<[MatchAccount](#)>()
- List< [MatchRound](#) > **scores** = new List<[MatchRound](#)>()

27.15 Combu.MiniJSON Class Reference

Static Public Member Functions

- static object [jsonDecode](#) (string json)
 - Parses the string json into a value*
- static string [jsonEncode](#) (object json)
 - Converts a Hashtable / ArrayList / Dictionary(string,string) object into a JSON string*
- static bool [lastDecodeSuccessful](#) ()
 - On decoding, this function returns the position at which the parse failed (-1 = no error).*
- static int [getLastErrorIndex](#) ()
 - On decoding, this function returns the position at which the parse failed (-1 = no error).*
- static string [getLastErrorSnippet](#) ()
 - If a decoding error occurred, this function returns a piece of the JSON string at which the error took place. To ease debugging.*

Static Protected Member Functions

- static Hashtable **parseObject** (char[] json, ref int index)
- static ArrayList **parseArray** (char[] json, ref int index)
- static object **parseValue** (char[] json, ref int index, ref bool success)
- static string **parseString** (char[] json, ref int index)
- static double **parseNumber** (char[] json, ref int index)
- static int **getLastIndexOfNumber** (char[] json, int index)
- static void **eatWhitespace** (char[] json, ref int index)
- static int **lookAhead** (char[] json, int index)
- static int **nextToken** (char[] json, ref int index)
- static bool **serializeObjectOrArray** (object objectOrArray, StringBuilder builder)
- static bool **serializeObject** (Hashtable anObject, StringBuilder builder)
- static bool **serializeDictionary** (Dictionary< string, string > dict, StringBuilder builder)
- static bool **serializeArray** (ArrayList anArray, StringBuilder builder)
- static bool **serializeValue** (object value, StringBuilder builder)
- static void **serializeString** (string aString, StringBuilder builder)
- static void **serializeNumber** (double number, StringBuilder builder)

Static Protected Attributes

- static int `lastErrorIndex` = -1
On decoding, this value holds the position at which the parse failed (-1 = no error).
- static string `lastDecode` = ""

27.15.1 Member Function Documentation

27.15.1.1 getLastErrorIndex()

```
static int Combu.MinijSON.getLastErrorIndex ( ) [static]
```

On decoding, this function returns the position at which the parse failed (-1 = no error).

Returns

27.15.1.2 getLastErrorSnippet()

```
static string Combu.MinijSON.getLastErrorSnippet ( ) [static]
```

If a decoding error occurred, this function returns a piece of the JSON string at which the error took place. To ease debugging.

Returns

27.15.1.3 jsonDecode()

```
static object Combu.MinijSON.jsonDecode (
    string json ) [static]
```

Parses the string json into a value

Parameters

<i>json</i>	A JSON string.
-------------	----------------

Returns

An ArrayList, a Hashtable, a double, a string, null, true, or false

27.15.1.4 jsonEncode()

```
static string Combu.MinijJSON.jsonEncode (  
    object json ) [static]
```

Converts a Hashtable / ArrayList / Dictionary(string,string) object into a JSON string

Parameters

<i>json</i>	A Hashtable / ArrayList
-------------	-------------------------

Returns

A JSON encoded string, or null if object 'json' is not serializable

27.15.1.5 lastDecodeSuccessful()

```
static bool Combu.MinijJSON.lastDecodeSuccessful ( ) [static]
```

On decoding, this function returns the position at which the parse failed (-1 = no error).

Returns**27.15.2 Member Data Documentation****27.15.2.1 lastErrorIndex**

```
int Combu.MinijJSON.lastErrorIndex = -1 [static], [protected]
```

On decoding, this value holds the position at which the parse failed (-1 = no error).

27.16 Combu.News Class Reference

Public Member Functions

- virtual void [FromJson](#) (string jsonString)
Initialize the object from a JSON formatted string.
- virtual void [FromHashtable](#) (Hashtable hash)
Initialize the object from a hashtable.

Static Public Member Functions

- static void [Load](#) (int pageNumber, int limit, Action< [News](#)[], int, int, string > callback)
Load the specified pageNumber and limit of news.
- static void [Load< T >](#) (int pageNumber, int limit, Action< [News](#)[], int, int, string > callback)
Load the specified pageNumber and limit of news.

Public Attributes

- long **id** = 0
- DateTime **date** = DateTime.MinValue
- string **subject** = ""
- string **message** = ""
- string **url** = ""

27.16.1 Member Function Documentation

27.16.1.1 FromHashtable()

```
virtual void Combu.News.FromHashtable (  
    Hashtable hash ) [virtual]
```

Initialize the object from a hashtable.

Parameters

<i>hash</i>	Hash.
-------------	-------

27.16.1.2 FromJson()

```
virtual void Combu.News.FromJson (  
    string jsonString ) [virtual]
```

Initialize the object from a JSON formatted string.

Parameters

<i>jsonString</i>	Json string.
-------------------	--------------

27.16.1.3 Load()

```
static void Combu.News.Load (
    int pageNumber,
    int limit,
    Action< News[], int, int, string > callback ) [static]
```

Load the specified pageNumber and limit of news.

Parameters

<i>pageNumber</i>	Page number.
<i>limit</i>	Limit.
<i>callback</i>	Callback.

27.16.1.4 Load< T >()

```
static void Combu.News.Load< T > (
    int pageNumber,
    int limit,
    Action< News[], int, int, string > callback ) [static]
```

Load the specified pageNumber and limit of news.

Parameters

<i>pageNumber</i>	Page number.
<i>limit</i>	Limit.
<i>callback</i>	Callback.

Template Parameters

<i>T</i>	The 1st type parameter.
----------	-------------------------

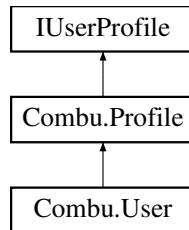
Type Constraints

***T* : News**

***T* : new()**

27.17 Combu.Profile Class Reference

Inheritance diagram for Combu.Profile:



Public Member Functions

- [Profile](#) (string jsonString)
Initializes a new instance of the CBUUser class from a JSON formatted string.
- [Profile](#) (Hashtable hash)
Initializes a new instance of the CBUUser class from a Hashtable.
- virtual void [FromJson](#) (string jsonString)
Initialize the object from a JSON formatted string.
- virtual void [FromHashtable](#) (Hashtable hash)
Initialize the object from a hashtable.

Public Attributes

- string [email](#)
The email address.
- Hashtable [customData](#) = new Hashtable()
The global custom data shared between apps.
- Hashtable [appCustomData](#) = new Hashtable()
The custom data for the current app scope.

Protected Attributes

- long [_id](#) = 0
- string [_userName](#) = ""
- Texture2D [_image](#)
- string [_sessionToken](#) = ""
- System.DateTime [_lastSeen](#)

Properties

- List< [ProfilePlatform](#) > **platforms** [get]
- string **id** [get]

Gets the identifier value as string.
- long **idLong** [get]

Gets the identifier value as long. id is just a ToString() of idLong, since ids are stored as long in the database.
- string **userName** [get, set]

Gets or sets the name of the user.
- bool **isFriend** [get]

Gets a value indicating whether this [Combu.Profile](#) is a friend of the local user.
- virtual UserState **state** [get]

Gets the online state.
- Texture2D **image** [get, set]

Gets or sets the image.
- string **sessionToken** [get]

Gets the session token.
- System.DateTime **lastSeen** [get]

Gets the last seen date/time.

27.17.1 Constructor & Destructor Documentation

27.17.1.1 Profile() [1/2]

```
Combu.Profile.Profile (
    string jsonString )
```

Initializes a new instance of the CBUUser class from a JSON formatted string.

Parameters

<i>jsonString</i>	JSON formatted string.
-------------------	------------------------

27.17.1.2 Profile() [2/2]

```
Combu.Profile.Profile (
    Hashtable hash )
```

Initializes a new instance of the CBUUser class from a Hashtable.

Parameters

<i>hash</i>	Hash.
-------------	-------

27.17.2 Member Function Documentation

27.17.2.1 FromHashtable()

```
virtual void Combu.Profile.FromHashtable (
    Hashtable hash ) [virtual]
```

Initialize the object from a hashtable.

Parameters

<i>hash</i>	Hash.
-------------	-------

27.17.2.2 FromJson()

```
virtual void Combu.Profile.FromJson (
    string jsonString ) [virtual]
```

Initialize the object from a JSON formatted string.

Parameters

<i>jsonString</i>	Json string.
-------------------	--------------

27.17.3 Member Data Documentation

27.17.3.1 appCustomData

```
Hashtable Combu.Profile.appCustomData = new Hashtable()
```

The custom data for the current app scope.

27.17.3.2 customData

```
Hashtable Combu.Profile.customData = new Hashtable()
```

The global custom data shared between apps.

27.17.3.3 email

```
string Combu.Profile.email
```

The email address.

27.17.4 Property Documentation

27.17.4.1 id

```
string Combu.Profile.id [get]
```

Gets the identifier value as string.

The identifier.

27.17.4.2 idLong

```
long Combu.Profile.idLong [get]
```

Gets the identifier value as long. id is just a ToString() of idLong, since Ids are stored as long in the database.

The identifier long.

27.17.4.3 image

```
Texture2D Combu.Profile.image [get], [set]
```

Gets or sets the image.

The image.

27.17.4.4 isFriend

```
bool Combu.Profile.isFriend [get]
```

Gets a value indicating whether this [Combu.Profile](#) is a friend of the local user.

true if is friend; otherwise, false.

27.17.4.5 lastSeen

```
System.DateTime Combu.Profile.lastSeen [get]
```

Gets the last seen date/time.

The last seen.

27.17.4.6 sessionToken

```
string Combu.Profile.sessionToken [get]
```

Gets the session token.

The session token.

27.17.4.7 state

```
virtual UserState Combu.Profile.state [get]
```

Gets the online state.

The state.

27.17.4.8 userName

```
string Combu.Profile.userName [get], [set]
```

Gets or sets the name of the user.

The name of the user.

27.18 Combu.ProfilePlatform Class Reference

Public Member Functions

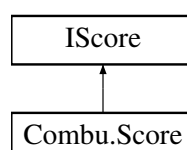
- **ProfilePlatform** (Hashtable data)

Public Attributes

- string **platformKey** = ""
- string **platformId** = ""

27.19 Combu.Score Class Reference

Inheritance diagram for Combu.Score:



Public Member Functions

- **Score** (string idLeaderboard, int rank, [User](#) user, double score)
- **Score** (string idLeaderboard, string codeLeaderboard, int rank, [User](#) user, double score)
- void **Initialize** (string idLeaderboard, int rank, [Profile](#) user, double score)
Initialize this [Score](#) with the specified idLeaderboard, rank, user and score.
- void **Initialize** (string idLeaderboard, string codeLeaderboard, int rank, [Profile](#) user, double score)
Initialize this [Score](#) with the specified idLeaderboard, rank, user and score.
- void **ReportScore** (System.Action< bool > callback)
Reports the score.

Properties

- string **leaderboardID** [get, set]
- string **leaderboardCode** [get, set]
- System.DateTime **date** [get]
- string **formattedValue** [get]
- string **userID** [get]
- [Profile](#) **user** [get]
- int **rank** [get]
- long **value** [get, set]
- float **valueFloat** [get, set]
- double **valueDouble** [get, set]

27.19.1 Member Function Documentation

27.19.1.1 Initialize() [1/2]

```
void Combu.Score.Initialize (
    string idLeaderboard,
    int rank,
    Profile user,
    double score )
```

Initialize this [Score](#) with the specified idLeaderboard, rank, user and score.

Parameters

<i>idLeaderboard</i>	Identifier leaderboard.
<i>rank</i>	Rank.
<i>user</i>	User .
<i>score</i>	Score .

27.19.1.2 Initialize() [2/2]

```
void Combu.Score.Initialize (
    string idLeaderboard,
    string codeLeaderboard,
    int rank,
    Profile user,
    double score )
```

Initialize this [Score](#) with the specified idLeaderboard, rank, user and score.

Parameters

<i>idLeaderboard</i>	Identifier of the leaderboard.
<i>codeLeaderboard</i>	Code of the leaderboard.
<i>rank</i>	Rank.
<i>user</i>	User .
<i>score</i>	Score .

27.19.1.3 ReportScore()

```
void Combu.Score.ReportScore (
    System.Action< bool > callback )
```

Reports the score.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.20 Combu.SearchCustomData Class Reference

Class to handle generic a generic search filter.

Public Member Functions

- **SearchCustomData** (string key, [eSearchOperator](#) op, string value)

Public Attributes

- string **key**
- [eSearchOperator](#) **op**
- string **value**

27.20.1 Detailed Description

Class to handle generic a generic search filter.

27.21 Combu.Tournament Class Reference

Tournaments class.

Public Member Functions

- [Tournament](#) (string jsonString)
Initializes a new instance of the [Combu.Tournament](#) class.
- [Tournament](#) (Hashtable data)
Initializes a new instance of the [Combu.Tournament](#) class.
- virtual void [FromJson](#) (string jsonString)
Initialize the object from a JSON formatted string.
- virtual void [FromHashtable](#) (Hashtable hash)
Initialize the object from a hashtable.
- virtual void [Save](#) (Action< bool, string > callback)
Save this instance in the database.
- virtual void [Delete](#) (Action< bool, string > callback)
Delete this instance from the database.
- void [Leave](#) (Action< bool, string > callback)

Static Public Member Functions

- static void [Delete](#) (long idTournament, Action< bool, string > callback)
Delete the specified [Tournament](#).
- static void [Load](#) (bool finished, [SearchCustomData](#)[] customData, System.Action< [Tournament](#)[]> callback)
Load the list of tournaments with the specified filters.
- static void [Load](#) (long id, System.Action< [Tournament](#) > callback)
Load the tournament with the specified id.
- static void [Leave](#) (long idTournament, long idUser, Action< bool, string > callback)
- static [Tournament QuickTournament](#) ([Profile](#)[] users)
Creates a quick tournament. If the number of other users is 1 then it creates 3 rounds, else 2 rounds for each enemy.

Public Attributes

- long **id** = 0
- long **idOwner** = 0
- string **title** = ""
- DateTime **dateCreation** = DateTime.Now
- DateTime **dateFinished** = null
- Hashtable **customData** = new Hashtable()

Properties

- [Profile owner](#) [get]
- List< [Match](#) > **matches** [get]
- bool **finished** [get]

27.21.1 Detailed Description

Tournaments class.

27.21.2 Constructor & Destructor Documentation

27.21.2.1 Tournament() [1/2]

```
Combu.Tournament.Tournament (
    string jsonString )
```

Initializes a new instance of the [Combu.Tournament](#) class.

Parameters

<i>jsonString</i>	JSON string to initialize the instance.
-------------------	---

27.21.2.2 Tournament() [2/2]

```
Combu.Tournament.Tournament (
    Hashtable data )
```

Initializes a new instance of the [Combu.Tournament](#) class.

Parameters

<i>data</i>	Data to initialize the instance.
-------------	----------------------------------

27.21.3 Member Function Documentation

27.21.3.1 Delete() [1/2]

```
virtual void Combu.Tournament.Delete (
    Action< bool, string > callback ) [virtual]
```

Delete this instance from the database.

27.21.3.2 Delete() [2/2]

```
static void Combu.Tournament.Delete (
    long idTournament,
    Action< bool, string > callback ) [static]
```

Delete the specified [Tournament](#).

Parameters

<i>idTournament</i>	Identifier tournament.
<i>callback</i>	Callback.

27.21.3.3 FromHashtable()

```
virtual void Combu.Tournament.FromHashtable (
    Hashtable hash ) [virtual]
```

Initialize the object from a hashtable.

Parameters

<i>hash</i>	Hash.
-------------	-------

27.21.3.4 FromJson()

```
virtual void Combu.Tournament.FromJson (
    string jsonString ) [virtual]
```

Initialize the object from a JSON formatted string.

Parameters

<i>jsonString</i>	Json string.
-------------------	--------------

27.21.3.5 Load() [1/2]

```
static void Combu.Tournament.Load (
```

```
bool finished,
SearchCustomData [] customData,
System.Action< Tournament []> callback ) [static]
```

Load the list of tournaments with the specified filters.

Parameters

<i>finished</i>	If set to <code>true</code> then it returns also the finished tournaments.
<i>customData</i>	Custom data.
<i>callback</i>	Callback.

27.21.3.6 Load() [2/2]

```
static void Combu.Tournament.Load (
    long id,
    System.Action< Tournament > callback ) [static]
```

Load the tournament with the specified id.

Parameters

<i>id</i>	Identifier.
<i>callback</i>	Callback.

27.21.3.7 QuickTournament()

```
static Tournament Combu.Tournament.QuickTournament (
    Profile [] users ) [static]
```

Creates a quick tournament. If the number of other users is 1 then it creates 3 rounds, else 2 rounds for each enemy.

Returns

The tournament.

Parameters

<i>users</i>	The list of other users (excluding <code>localUser</code>).
--------------	--

27.21.3.8 Save()

```
virtual void Combu.Tournament.Save (
    Action< bool, string > callback ) [virtual]
```

Save this instance in the database.

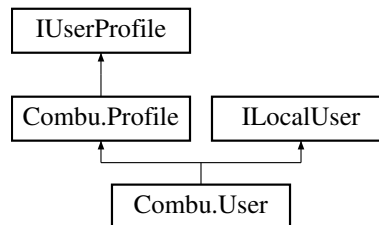
Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.22 Combu.User Class Reference

[User](#) class implementing the Unity built-in Social interfaces (specialized IUserProfile, ILocalUser).

Inheritance diagram for Combu.User:



Public Member Functions

- **User** (bool authenticated)
- **User** (string jsonString)
- **User** (Hashtable hash)
- virtual void [FromUser](#) ([User](#) source)
 - Sets the data from another user object.*
- virtual void [Authenticate](#) (System.Action< bool > callback)
 - Authenticate the user.*
- virtual void [Authenticate](#) (System.Action< bool, string > callback)
 - Authenticate the user.*
- virtual void [Authenticate](#) (string password, System.Action< bool, string > callback)
 - Authenticate the user with the specified password.*
- virtual void [CreateGuest](#) (System.Action< bool, string > callback)
 - Creates a guest account.*
- virtual void [LoadFriends](#) (System.Action< bool > callback)
 - Loads the friends of the current logged user.*
- virtual void [LoadFriends](#) ([eContactType](#) contactType, System.Action< bool > callback)
 - Loads the friends of the current logged user.*
- virtual void [LoadFriends< T >](#) ([eContactType](#) contactType, System.Action< bool > callback)
 - Loads the friends of the current logged user.*
- virtual void [Update](#) (System.Action< bool, string > callback)
 - Update or Create this user to server, whether id is positive and greater than zero.*

- virtual void [Delete](#) (System.Action< bool, string > callback)
Delete this instance from the server.
- void [Load](#) (System.Action< bool > callback)
Load of the current user from server.
- virtual void [ResetPassword](#) (System.Action< bool, string > callback)
Resets the password of this user.
- virtual void [ChangePassword](#) (string newPassword, System.Action< bool, string > callback)
Changes the password of this user.
- virtual void [AuthenticatePlatform](#) (string platformKey, string platformId, System.Action< bool, string > callback)
Authenticates the user from an external platform (like Facebook, Game Center, GooglePlay etc). Note you are the only responsible for the external authentication, [Combu](#) only stores the info that you send.
- virtual void [AuthenticatePlatform](#)< T > (string platformKey, string platformId, System.Action< bool, string > callback)
- virtual void [LinkAccount](#) (string username, string password, System.Action< bool, string > callback)
Links the currently logged account to another: all the platforms Ids of the current user will be transferred to the new account and the current account will be deleted.
- virtual void [LinkPlatform](#) (string platformKey, string platformId, System.Action< bool, string > callback)
Links a new platform Id to the logged account.
- void [GetContact](#) (string idOrUsername, Action< bool, string, [eContactType](#)?, [User](#) > callback)
Gets the contact information with loggedAccount if there's a pending request or is in the friends/ignored lists. If there's no relation found then the callback will return a failure.
- void [GetContact](#)< T > (string idOrUsername, Action< bool, string, [eContactType](#)?, [User](#) > callback)
Gets the contact information with loggedAccount if there's a pending request or is in the friends/ignored lists. If there's no relation found then the callback will return a failure.
- void [AddContact](#) (string otherUsername, [eContactType](#) contactType, System.Action< bool, string > callback)
Adds the contact.
- void [AddContact](#) (long otherId, [eContactType](#) contactType, System.Action< bool, string > callback)
Adds the contact.
- void [AddContact](#) ([Profile](#) otherUser, [eContactType](#) contactType, System.Action< bool, string > callback)
Adds the contact.
- void [RemoveContact](#) (string otherUsername, System.Action< bool, string > callback)
Removes the contact.
- void [RemoveContact](#) (long otherId, System.Action< bool, string > callback)
Removes the contact.
- void [RemoveContact](#) ([Profile](#) otherUser, System.Action< bool, string > callback)
Removes the contact.

Static Public Member Functions

- static bool [CanAutoLogin](#) (out string username, out string password)
Determines if can auto-login and output the stored username and password.
- static void [AutoLogin](#) (System.Action< bool, string > callback)
Automatically logins with the credentials stored in PlayerPrefs.
- static void [AutoLogin](#)< T > (System.Action< bool, string > callback)
- static void [Delete](#) (string username, string password, System.Action< bool, string > callback)
Delete a user from the server.
- static void [Load](#) ([User](#)[] updateUsers, System.Action< bool > callback)
Reload the specified users from server (by Id).
- static void [Load](#) (long userId, System.Action< [User](#) > callback)
Loads a user by Id.

- static void **Load** (string **userName**, System.Action< **User** > callback)
Loads a user by userName.
- static void **Load** (long[] **userIds**, System.Action< **User**[]> callback)
Loads the users by Id.
- static void **Load** (string[] **userNames**, System.Action< **User**[]> callback)
Loads the users by userName.
- static void **Load**< **T** > (string **username**, string **email**, **SearchCustomData**[] **customData**, bool **isOnline**, int **pageNumber**, int **limit**, System.Action< **T**[], int, int > callback)
Loads the users by searching for the specified parameters.
- static void **Random**< **T** > (**SearchCustomData**[] **customData**, int **count**, System.Action< **T**[]> callback)
Loads a specified count of random users.
- static void **Exists** (string **username**, string **email**, System.Action< bool, string > callback)
Verify if it exists an account with the specified username and email.
- static void **ResetPassword** (long **idUser**, System.Action< bool, string > callback)
Resets the password of a user by Id Account.
- static void **ResetPassword** (string **username**, System.Action< bool, string > callback)
Resets the password of a user by Username.
- static void **ChangePassword** (long **idUser**, string **username**, string **resetCode**, string **newPassword**, System.Action< bool, string > callback)
Changes the password of a user.
- static void **LoadPlatform** (IEnumerable< string > **platformKeys**, IEnumerable< string > **platformIds**, System.Action< **User**[]> callback)
- static void **LoadPlatform**< **T** > (IEnumerable< string > **platformKeys**, IEnumerable< string > **platformIds**, System.Action< **T**[]> callback)

Public Attributes

- string **password**

Properties

- IUserProfile [] **friends** [get]
- IUserProfile [] **ignored** [get]
- IUserProfile [] **requests** [get]
- IUserProfile [] **pendingRequests** [get]
- bool **authenticated** [get]
- virtual bool **underage** [get]

Additional Inherited Members

27.22.1 Detailed Description

User class implementing the Unity built-in Social interfaces (specialized IUserProfile, ILocalUser).

27.22.2 Member Function Documentation

27.22.2.1 AddContact() [1/3]

```
void Combu.User.AddContact (
    string otherUsername,
    eContactType contactType,
    System.Action< bool, string > callback )
```

Adds the contact.

Parameters

<i>otherUsername</i>	Other username.
<i>contactType</i>	Contact type.
<i>callback</i>	Callback.

27.22.2.2 AddContact() [2/3]

```
void Combu.User.AddContact (
    long otherId,
    eContactType contactType,
    System.Action< bool, string > callback )
```

Adds the contact.

Parameters

<i>otherId</i>	Other identifier.
<i>contactType</i>	Contact type.
<i>callback</i>	Callback.

27.22.2.3 AddContact() [3/3]

```
void Combu.User.AddContact (
    Profile otherUser,
    eContactType contactType,
    System.Action< bool, string > callback )
```

Adds the contact.

Parameters

<i>otherUser</i>	Other user.
<i>contactType</i>	Contact type.
<i>callback</i>	Callback.

27.22.2.4 Authenticate() [1/3]

```
virtual void Combu.User.Authenticate (
    System.Action< bool > callback ) [virtual]
```

Authenticate the user.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.22.2.5 Authenticate() [2/3]

```
virtual void Combu.User.Authenticate (
    System.Action< bool, string > callback ) [virtual]
```

Authenticate the user.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.22.2.6 Authenticate() [3/3]

```
virtual void Combu.User.Authenticate (
    string password,
    System.Action< bool, string > callback ) [virtual]
```

Authenticate the user with the specified password.

Parameters

<i>password</i>	Password.
<i>callback</i>	Callback.

Template Parameters

<i>T</i>	Type for User .
----------	---------------------------------

27.22.2.7 AuthenticatePlatform()

```
virtual void Combu.User.AuthenticatePlatform (
    string platformKey,
```



```
string platformId,
System.Action< bool, string > callback ) [virtual]
```

Authenticates the user from an external platform (like Facebook, Game Center, GooglePlay etc). Note you are the only responsible for the external authentication, [Combu](#) only stores the info that you send.

Parameters

<i>platformKey</i>	Platform key.
<i>platformId</i>	Platform identifier.
<i>callback</i>	Callback.

27.22.2.8 AutoLogin()

```
static void Combu.User.AutoLogin (
    System.Action< bool, string > callback ) [static]
```

Automatically logs in with the credentials stored in PlayerPrefs.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.22.2.9 CanAutoLogin()

```
static bool Combu.User.CanAutoLogin (
    out string username,
    out string password ) [static]
```

Determines if can auto-login and output the stored username and password.

Returns

true if can auto-login; otherwise, false.

Parameters

<i>username</i>	Username stored.
<i>password</i>	Password stored.

27.22.2.10 ChangePassword() [1/2]

```
virtual void Combu.User.ChangePassword (
```

```

    string newPassword,
    System.Action< bool, string > callback ) [virtual]

```

Changes the password of this user.

Parameters

<i>newPassword</i>	New password.
<i>callback</i>	Callback.

27.22.2.11 ChangePassword() [2/2]

```

static void Combu.User.ChangePassword (
    long idUser,
    string username,
    string resetCode,
    string newPassword,
    System.Action< bool, string > callback ) [static]

```

Changes the password of a user.

Parameters

<i>idUser</i>	Identifier user.
<i>username</i>	Username.
<i>resetCode</i>	Reset code.
<i>newPassword</i>	New password.
<i>callback</i>	Callback.

27.22.2.12 CreateGuest()

```

virtual void Combu.User.CreateGuest (
    System.Action< bool, string > callback ) [virtual]

```

Creates a guest account.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.22.2.13 Delete() [1/2]

```

virtual void Combu.User.Delete (
    System.Action< bool, string > callback ) [virtual]

```

Delete this instance from the server.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.22.2.14 Delete() [2/2]

```
static void Combu.User.Delete (
    string username,
    string password,
    System.Action< bool, string > callback ) [static]
```

Delete a user from the server.

Parameters

<i>username</i>	Username.
<i>password</i>	Password.
<i>callback</i>	Callback.

27.22.2.15 Exists()

```
static void Combu.User.Exists (
    string username,
    string email,
    System.Action< bool, string > callback ) [static]
```

Verify if it exists an account with the specified username and email.

Parameters

<i>username</i>	Username.
<i>email</i>	Email.
<i>callback</i>	Callback.

27.22.2.16 FromUser()

```
virtual void Combu.User.FromUser (
    User source ) [virtual]
```

Sets the data from another user object.

Parameters

<i>source</i>	Source.
---------------	---------

27.22.2.17 GetContact()

```
void Combu.User.GetContact (
    string idOrUsername,
    Action< bool, string, eContactType?, User > callback )
```

Gets the contact information with loggedAccount if there's a pending request or is in the friends/ignored lists. If there's no relation found then the callback will return a failure.

Parameters

<i>idOrUsername</i>	User id or username.
<i>callback</i>	Callback.

27.22.2.18 GetContact< T >()

```
void Combu.User.GetContact< T > (
    string idOrUsername,
    Action< bool, string, eContactType?, User > callback )
```

Gets the contact information with loggedAccount if there's a pending request or is in the friends/ignored lists. If there's no relation found then the callback will return a failure.

Parameters

<i>idOrUsername</i>	User id or username.
<i>callback</i>	Callback.

Type Constraints

***T* : User**

***T* : new()**

27.22.2.19 LinkAccount()

```
virtual void Combu.User.LinkAccount (
    string username,
```

```
string password,  
System.Action< bool, string > callback ) [virtual]
```

Links the currently logged account to another: all the platforms ids of the current user will be transferred to the new account and the current account will be deleted.

Parameters

<i>username</i>	Username.
<i>password</i>	Password.
<i>callback</i>	Callback.

27.22.2.20 LinkPlatform()

```
virtual void Combu.User.LinkPlatform (
    string platformKey,
    string platformId,
    System.Action< bool, string > callback ) [virtual]
```

Links a new platform Id to the logged account.

Parameters

<i>platformKey</i>	Platform key.
<i>platformId</i>	Platform identifier.
<i>callback</i>	Callback.

27.22.2.21 Load() [1/6]

```
static void Combu.User.Load (
    User [] updateUsers,
    System.Action< bool > callback ) [static]
```

Reload the specified users from server (by Id).

Parameters

<i>updateUsers</i>	List of users.
<i>callback</i>	Callback.

27.22.2.22 Load() [2/6]

```
void Combu.User.Load (
    System.Action< bool > callback )
```

Load of the current user from server.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.22.2.23 Load() [3/6]

```
static void Combu.User.Load (
    long userId,
    System.Action< User > callback ) [static]
```

Loads a user by Id.

Parameters

<i>userId</i>	User Id.
<i>callback</i>	Callback.

27.22.2.24 Load() [4/6]

```
static void Combu.User.Load (
    string userName,
    System.Action< User > callback ) [static]
```

Loads a user by userName.

Parameters

<i>userName</i>	User Name.
<i>callback</i>	Callback.

27.22.2.25 Load() [5/6]

```
static void Combu.User.Load (
    long [] userIds,
    System.Action< User[] > callback ) [static]
```

Loads the users by Id.

Parameters

<i>userIds</i>	User Ids.
<i>callback</i>	Callback.
<i>updateUser</i>	If passed its data will be replaced with the server result.

27.22.2.26 Load() [6/6]

```
static void Combu.User.Load (
    string [] usernames,
    System.Action< User[]> callback ) [static]
```

Loads the users by userName.

Parameters

<i>userNames</i>	User Names.
<i>callback</i>	Callback.

27.22.2.27 Load< T >()

```
static void Combu.User.Load< T > (
    string username,
    string email,
    SearchCustomData [] customData,
    bool isOnline,
    int pageNumber,
    int limit,
    System.Action< T[], int, int > callback ) [static]
```

Loads the users by searching for the specified parameters.

Parameters

<i>username</i>	Username.
<i>email</i>	Email.
<i>customData</i>	Custom data.
<i>pageNumber</i>	Page number.
<i>limit</i>	Limit.
<i>callback</i>	Callback.

Type Constraints

T: User

T: new()

27.22.2.28 LoadFriends() [1/2]

```
virtual void Combu.User.LoadFriends (
    System.Action< bool > callback ) [virtual]
```


Loads the friends of the current logged user.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.22.2.29 LoadFriends() [2/2]

```
virtual void Combu.User.LoadFriends (
    eContactType contactType,
    System.Action< bool > callback ) [virtual]
```

Loads the friends of the current logged user.

Parameters

<i>contactType</i>	Contact type.
<i>callback</i>	Callback.

27.22.2.30 LoadFriends< T >()

```
virtual void Combu.User.LoadFriends< T > (
    eContactType contactType,
    System.Action< bool > callback ) [virtual]
```

Loads the friends of the current logged user.

Parameters

<i>contactType</i>	Contact type.
<i>callback</i>	Callback.

Template Parameters

<i>T</i>	Type for User .
----------	---------------------------------

Type Constraints

***T* : [User](#)**

***T* : [new\(\)](#)**

27.22.2.31 Random< T >()

```
static void Combu.User.Random< T > (
    SearchCustomData [] customData,
    int count,
    System.Action< T[]> callback ) [static]
```

Loads a specified count of random users.

Parameters

<i>customData</i>	Custom data.
<i>count</i>	Count.
<i>callback</i>	Callback.

Type Constraints

***T* : User**

***T* : new()**

27.22.2.32 RemoveContact() [1/3]

```
void Combu.User.RemoveContact (
    string otherUsername,
    System.Action< bool, string > callback )
```

Removes the contact.

Parameters

<i>otherUsername</i>	Other username.
<i>callback</i>	Callback.

27.22.2.33 RemoveContact() [2/3]

```
void Combu.User.RemoveContact (
    long otherId,
    System.Action< bool, string > callback )
```

Removes the contact.

Parameters

<i>otherId</i>	Other identifier.
<i>callback</i>	Callback.

27.22.2.34 RemoveContact() [3/3]

```
void Combu.User.RemoveContact (
    Profile otherUser,
    System.Action< bool, string > callback )
```

Removes the contact.

Parameters

<i>otherUser</i>	Other user.
<i>callback</i>	Callback.

27.22.2.35 ResetPassword() [1/3]

```
virtual void Combu.User.ResetPassword (
    System.Action< bool, string > callback ) [virtual]
```

Resets the password of this user.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.22.2.36 ResetPassword() [2/3]

```
static void Combu.User.ResetPassword (
    long idUser,
    System.Action< bool, string > callback ) [static]
```

Resets the password of a user by Id Account.

Parameters

<i>idUser</i>	Identifier user.
<i>callback</i>	Callback.

27.22.2.37 ResetPassword() [3/3]

```
static void Combu.User.ResetPassword (
```

```
string username,
System.Action< bool, string > callback ) [static]
```

Resets the password of a user by Username.

Parameters

<i>username</i>	Username.
<i>callback</i>	Callback.

27.22.2.38 Update()

```
virtual void Combu.User.Update (
System.Action< bool, string > callback ) [virtual]
```

Update or Create this user to server, whether id is positive and greater than zero.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.23 Combu.UserFile Class Reference

Public Types

- enum **eShareType** { **Everybody**, **Nobody**, **Friends** }

Public Member Functions

- [UserFile](#) ()
Initializes a new instance of the [UserFile](#) class.
- [UserFile](#) (string jsonString)
Initializes a new instance of the [UserFile](#) class from a JSON formatted string.
- [UserFile](#) (Hashtable hash)
Initializes a new instance of the [UserFile](#) class from a Hashtable.
- virtual void [FromJson](#) (string jsonString)
Initialize the object from a JSON formatted string.
- virtual void [FromHashtable](#) (Hashtable hash)
Initialize the object from a hashtable.
- virtual void [Update](#) (byte[] contents, Action< bool, string > callback)
Update this file to server.
- virtual void [Delete](#) (Action< bool, string > callback)
Deletes this [UserFile](#) from server.
- virtual void [View](#) (Action< bool, string > callback)
Increase the View count of this [UserFile](#).
- virtual void [Like](#) (Action< bool, string > callback)
Increase the Like count of this [UserFile](#).
- void [Download](#) (Action< byte[]> callback)
Download the bytes from the url.

Static Public Member Functions

- static void [Load](#) (string `userId`, bool `includeShared`, int `pageNumber`, int `countPerPage`, Action< [UserFile](#)[]>, int, int, string > callback)
Load the UserFiles with the specified `userId`, `includeShared`, `pageNumber`, `countPerPage` and `callback`.
- static void [Load< T >](#) (string `userId`, bool `includeShared`, int `pageNumber`, int `countPerPage`, Action< [UserFile](#)[]>, int, int, string > callback)
Load the UserFiles with the specified `userId`, `includeShared`, `pageNumber`, `countPerPage` and `callback`.
- static void [Delete](#) (long `idFile`, Action< bool, string > callback)
Deletes the specified File from server.
- static void [View](#) (long `idFile`, Action< bool, string > callback)
Increase the View count of a [UserFile](#).
- static void [Like](#) (long `idFile`, Action< bool, string > callback)
Increase the Like count of a [UserFile](#).

Public Attributes

- string **name** = ""
- string **url** = ""
- eShareType **sharing** = eShareType.Nobody
- Hashtable **customData** = new Hashtable()

Static Protected Member Functions

- static void [View](#) ([UserFile](#) file, long `idFile`, Action< bool, string > callback)
Increase the View count of a [UserFile](#).
- static void [Like](#) ([UserFile](#) file, long `idFile`, Action< bool, string > callback)
Increase the Like count of a [UserFile](#).

Properties

- long **id** [get]
- long **idAccount** [get]
- int **views** [get]
- int **likes** [get]

27.23.1 Constructor & Destructor Documentation

27.23.1.1 [UserFile\(\)](#) [1/3]

```
Combu.UserFile.UserFile ( )
```

Initializes a new instance of the [UserFile](#) class.

27.23.1.2 [UserFile\(\)](#) [2/3]

```
Combu.UserFile.UserFile (
    string jsonString )
```

Initializes a new instance of the [UserFile](#) class from a JSON formatted string.

Parameters

<i>jsonString</i>	JSON formatted string.
-------------------	------------------------

27.23.1.3 UserFile() [3/3]

```
Combu.UserFile.UserFile (
    Hashtable hash )
```

Initializes a new instance of the [UserFile](#) class from a Hashtable.

Parameters

<i>hash</i>	Hash.
-------------	-------

27.23.2 Member Function Documentation

27.23.2.1 Delete() [1/2]

```
virtual void Combu.UserFile.Delete (
    Action< bool, string > callback ) [virtual]
```

Deletes this [UserFile](#) from server.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.23.2.2 Delete() [2/2]

```
static void Combu.UserFile.Delete (
    long idFile,
    Action< bool, string > callback ) [static]
```

Deletes the specified File from server.

Parameters

<i>idFile</i>	Identifier file.
<i>callback</i>	Callback.

27.23.2.3 Download()

```
void Combu.UserFile.Download (
    Action< byte[]> callback )
```

Download the bytes from the url.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.23.2.4 FromHashtable()

```
virtual void Combu.UserFile.FromHashtable (
    Hashtable hash ) [virtual]
```

Initialize the object from a hashtable.

Parameters

<i>hash</i>	Hash.
-------------	-------

27.23.2.5 FromJson()

```
virtual void Combu.UserFile.FromJson (
    string jsonString ) [virtual]
```

Initialize the object from a JSON formatted string.

Parameters

<i>jsonString</i>	Json string.
-------------------	--------------

27.23.2.6 Like() [1/3]

```
virtual void Combu.UserFile.Like (
    Action< bool, string > callback ) [virtual]
```

Increase the Like count of this [UserFile](#).

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.23.2.7 Like() [2/3]

```
static void Combu.UserFile.Like (
    long idFile,
    Action< bool, string > callback ) [static]
```

Increase the Like count of a [UserFile](#).

Parameters

<i>idFile</i>	Identifier file.
<i>callback</i>	Callback.

27.23.2.8 Like() [3/3]

```
static void Combu.UserFile.Like (
    UserFile file,
    long idFile,
    Action< bool, string > callback ) [static], [protected]
```

Increase the Like count of a [UserFile](#).

Parameters

<i>file</i>	File.
<i>idFile</i>	Identifier file (if File is null).
<i>callback</i>	Callback.

27.23.2.9 Load()

```
static void Combu.UserFile.Load (
    string userId,
    bool includeShared,
    int pageNumber,
    int countPerPage,
    Action< UserFile[], int, int, string > callback ) [static]
```

Load the UserFiles with the specified *userId*, *includeShared*, *pageNumber*, *countPerPage* and *callback*.

Parameters

<i>userId</i>	User identifier.
<i>includeShared</i>	If set to <code>true</code> include shared.
<i>pageNumber</i>	Page number.
<i>countPerPage</i>	Count per page.
<i>callback</i>	Callback.

27.23.2.10 Load< T >()

```
static void Combu.UserFile.Load< T > (
    string userId,
    bool includeShared,
    int pageNumber,
    int countPerPage,
    Action< UserFile[], int, int, string > callback ) [static]
```

Load the UserFiles with the specified `userId`, `includeShared`, `pageNumber`, `countPerPage` and `callback`.

Parameters

<i>userId</i>	User identifier.
<i>includeShared</i>	If set to <code>true</code> include shared.
<i>pageNumber</i>	Page number.
<i>countPerPage</i>	Count per page.
<i>callback</i>	Callback.

Template Parameters

<i>T</i>	Type for <code>UserFile</code> .
----------	----------------------------------

Type Constraints

***T* : `UserFile`**

***T* : `new()`**

27.23.2.11 Update()

```
virtual void Combu.UserFile.Update (
    byte [] contents,
    Action< bool, string > callback ) [virtual]
```

Update this file to server.

Parameters

<i>contents</i>	Contents of the file to send.
<i>callback</i>	Callback.

27.23.2.12 View() [1/3]

```
virtual void Combu.UserFile.View (
    Action< bool, string > callback ) [virtual]
```

Increase the View count of this [UserFile](#).

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.23.2.13 View() [2/3]

```
static void Combu.UserFile.View (
    long idFile,
    Action< bool, string > callback ) [static]
```

Increase the View count of a [UserFile](#).

Parameters

<i>idFile</i>	Identifier file.
<i>callback</i>	Callback.

27.23.2.14 View() [3/3]

```
static void Combu.UserFile.View (
    UserFile file,
    long idFile,
    Action< bool, string > callback ) [static], [protected]
```

Increase the View count of a [UserFile](#).

Parameters

<i>file</i>	File.
<i>idFile</i>	Identifier file (if File is null).
<i>callback</i>	Callback.

27.24 Combu.UserGroup Class Reference

Public Member Functions

- [UserGroup](#) ()
Initializes a new instance of the [UserGroup](#) class.
- [UserGroup](#) (string jsonString)
Initializes a new instance of the [UserGroup](#) class.
- [UserGroup](#) (Hashtable hash)
Initializes a new instance of the [UserGroup](#) class.
- virtual void [FromJson](#) (string jsonString)
Initialize the object from a JSON formatted string.
- virtual void [FromHashtable](#) (Hashtable hash)
Initialize the object from a hashtable.
- virtual void [Save](#) (System.Action< bool, string > callback)
Save this instance.
- virtual void [Delete](#) (System.Action< bool, string > callback)
Delete this instance.
- virtual void [Join](#) (System.Action< bool, string > callback)
Join the local user to this group.
- virtual void [Join](#) (long[] idUsers, System.Action< bool, string > callback)
Join the specified idUsers to this group.
- virtual void [Join](#) (string[] usernames, System.Action< bool, string > callback)
Join the specified usernames to this group.
- virtual void [Leave](#) (System.Action< bool, string > callback)
Leave the local user from this group.
- virtual void [Leave](#) (long[] idUsers, System.Action< bool, string > callback)
Leave the specified idUsers from this group.
- virtual void [Leave](#) (string[] usernames, System.Action< bool, string > callback)
Leave the specified usernames from this group.

Static Public Member Functions

- static void **Load** (long idOwner, System.Action< [UserGroup](#)[], string > callback)
- static void **Load** (string usernameOwner, System.Action< [UserGroup](#)[], string > callback)
- static void **LoadMembership** (long idMember, System.Action< [UserGroup](#)[], string > callback)
- static void **LoadMembership** (string usernameMember, System.Action< [UserGroup](#)[], string > callback)
- static void **Load** (string groupName, int pageNumber, int limit, System.Action< [UserGroup](#)[], int, int, string > callback)

Public Attributes

- long **id** = 0
- string **name**
- long **idOwner**
- [User](#) **owner**
- Hashtable **customData** = new Hashtable()
- [User](#) [] **users** = new [User](#)[0]

Protected Member Functions

- virtual void [Join](#)< T > (long[] idUsers, string[] usernames, System.Action< bool, string > callback)
Join the specified idUsers/usernames to this group.
- virtual void [Leave](#)< T > (long[] idUsers, string[] usernames, System.Action< bool, string > callback)
Leave the specified idUsers/usernames from this group.

27.24.1 Constructor & Destructor Documentation

27.24.1.1 [UserGroup\(\)](#) [1/3]

```
Combu.UserGroup.UserGroup ( )
```

Initializes a new instance of the [UserGroup](#) class.

27.24.1.2 [UserGroup\(\)](#) [2/3]

```
Combu.UserGroup.UserGroup (
    string jsonString )
```

Initializes a new instance of the [UserGroup](#) class.

Parameters

<i>jsonString</i>	Json string.
-------------------	--------------

27.24.1.3 [UserGroup\(\)](#) [3/3]

```
Combu.UserGroup.UserGroup (
    Hashtable hash )
```

Initializes a new instance of the [UserGroup](#) class.

Parameters

<i>hash</i>	Hash.
-------------	-------

27.24.2 Member Function Documentation

27.24.2.1 Delete()

```
virtual void Combu.UserGroup.Delete (
    System.Action< bool, string > callback ) [virtual]
```

Delete this instance.

27.24.2.2 FromHashtable()

```
virtual void Combu.UserGroup.FromHashtable (
    Hashtable hash ) [virtual]
```

Initialize the object from a hashtable.

Parameters

<i>hash</i>	Hash.
-------------	-------

27.24.2.3 FromJson()

```
virtual void Combu.UserGroup.FromJson (
    string jsonString ) [virtual]
```

Initialize the object from a JSON formatted string.

Parameters

<i>jsonString</i>	Json string.
-------------------	--------------

27.24.2.4 Join() [1/3]

```
virtual void Combu.UserGroup.Join (
    System.Action< bool, string > callback ) [virtual]
```

Join the local user to this group.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.24.2.5 Join() [2/3]

```
virtual void Combu.UserGroup.Join (
    long [] idUsers,
    System.Action< bool, string > callback ) [virtual]
```

Join the specified idUsers to this group.

Parameters

<i>idUsers</i>	Identifier users.
<i>callback</i>	Callback.

27.24.2.6 Join() [3/3]

```
virtual void Combu.UserGroup.Join (
    string [] usernames,
    System.Action< bool, string > callback ) [virtual]
```

Join the specified usernames to this group.

Parameters

<i>usernames</i>	Usernames.
<i>callback</i>	Callback.

27.24.2.7 Join< T >()

```
virtual void Combu.UserGroup.Join< T > (
    long [] idUsers,
    string [] usernames,
    System.Action< bool, string > callback ) [protected], [virtual]
```

Join the specified idUsers/usernames to this group.

Parameters

<i>idUsers</i>	Identifier users.
<i>usernames</i>	Usernames.
<i>callback</i>	Callback.

Template Parameters

<i>T</i>	Type for UserGroup records.
----------	---

Type Constraints

T* : *UserGroup***T* : *new()*****27.24.2.8 Leave()** [1/3]

```
virtual void Combu.UserGroup.Leave (
    System.Action< bool, string > callback ) [virtual]
```

Leave the local user from this group.

Parameters

<i>callback</i>	Callback.
-----------------	-----------

27.24.2.9 Leave() [2/3]

```
virtual void Combu.UserGroup.Leave (
    long [] idUsers,
    System.Action< bool, string > callback ) [virtual]
```

Leave the specified *idUsers* from this group.

Parameters

<i>idUsers</i>	Identifier users.
<i>callback</i>	Callback.

27.24.2.10 Leave() [3/3]

```
virtual void Combu.UserGroup.Leave (
    string [] usernames,
    System.Action< bool, string > callback ) [virtual]
```

Leave the specified *usernames* from this group.

Parameters

<i>usernames</i>	Usernames.
<i>callback</i>	Callback.

27.24.2.11 Leave< T >()

```
virtual void Combu.UserGroup.Leave< T > (
    long [] idUsers,
    string [] usernames,
    System.Action< bool, string > callback ) [protected], [virtual]
```

Leave the specified idUsers/usernames from this group.

Parameters

<i>idUsers</i>	Identifier users.
<i>usernames</i>	Usernames.
<i>callback</i>	Callback.

Template Parameters

<i>T</i>	Type for UserGroup records.
----------	---

Type Constraints

***T* : [UserGroup](#)**

***T* : [new\(\)](#)**

27.24.2.12 Save()

```
virtual void Combu.UserGroup.Save (
    System.Action< bool, string > callback ) [virtual]
```

Save this instance.

Index

- [_instance](#)
 - [Combu::CombuManager, 77](#)
 - [_platform](#)
 - [Combu::CombuManager, 77](#)
- [achievementUIFunction](#)
 - [Combu::CombuManager, 77](#)
- [achievementUIObject](#)
 - [Combu::CombuManager, 77](#)
- [AddBinaryData](#)
 - [Combu::CombuForm, 68](#)
- [AddContact](#)
 - [Combu::User, 140, 141](#)
- [AddField](#)
 - [Combu::CombuForm, 68](#)
- [AddUser](#)
 - [Combu::Match, 114](#)
- [appCustomData](#)
 - [Combu::Profile, 129](#)
- [appld](#)
 - [Combu::CombuManager, 77](#)
- [appSecret](#)
 - [Combu::CombuManager, 78](#)
- [Authenticate](#)
 - [Combu::CombuPlatform, 84, 85](#)
 - [Combu::User, 141, 142](#)
- [Authenticate< T >](#)
 - [Combu::CombuPlatform, 85](#)
- [AuthenticatePlatform](#)
 - [Combu::User, 142](#)
- [AutoLogin](#)
 - [Combu::User, 143](#)
- [AutoPing](#)
 - [Combu::CombuManager, 72](#)
- [COMBU_VERSION](#)
 - [Combu::CombuManager, 78](#)
- [CallWebservice](#)
 - [Combu::CombuManager, 73](#)
- [CanAutoLogin](#)
 - [Combu::User, 143](#)
- [CancelRequest](#)
 - [Combu::CombuManager, 73](#)
- [CaptureScreenshot](#)
 - [Combu::CombuManager, 73](#)
- [ChangePassword](#)
 - [Combu::User, 143, 144](#)
- [Combu, 59](#)
 - [eContactType, 60](#)
 - [eLeaderboardInterval, 60](#)
 - [eLeaderboardTimeScope, 60](#)
 - [eMailList, 61](#)
 - [eSearchOperator, 61](#)
- [Combu.Achievement, 63](#)
- [Combu.CombuEncryption, 64](#)
- [Combu.CombuForm, 67](#)
- [Combu.CombuManager, 70](#)
- [Combu.CombuPlatform, 82](#)
- [Combu.CombuServerInfo, 95](#)
- [Combu.Inventory, 96](#)
- [Combu.Leaderboard, 100](#)
- [Combu.Mail, 105](#)
- [Combu.MailCount, 112](#)
- [Combu.Match, 113](#)
- [Combu.Match.MatchRoundData, 122](#)
- [Combu.MatchAccount, 119](#)
- [Combu.MatchRound, 121](#)
- [Combu.MiniJSON, 122](#)
- [Combu.News, 125](#)
- [Combu.Profile, 127](#)
- [Combu.ProfilePlatform, 131](#)
- [Combu.Score, 131](#)
- [Combu.SearchCustomData, 133](#)
- [Combu.Tournament, 134](#)
- [Combu.User, 138](#)
- [Combu.UserFile, 155](#)
- [Combu.UserGroup, 162](#)
- [Combu::CombuEncryption](#)
 - [DecryptAES, 64](#)
 - [DecryptResponse, 65](#)
 - [EncryptAES, 65](#)
 - [EncryptMD5, 65](#)
 - [EncryptRSA, 66](#)
 - [EncryptSHA1, 66](#)
 - [LoadRSA, 66, 67](#)
 - [SetToken, 67](#)
- [Combu::CombuForm](#)
 - [AddBinaryData, 68](#)
 - [AddField, 68](#)
 - [GetBinaryField, 68](#)
 - [GetField, 69](#)
 - [GetForm, 69](#)
- [Combu::CombuManager](#)
 - [_instance, 77](#)
 - [_platform, 77](#)
 - [achievementUIFunction, 77](#)
 - [achievementUIObject, 77](#)
 - [appld, 77](#)
 - [appSecret, 78](#)

- AutoPing, [72](#)
- COMBU_VERSION, [78](#)
- CallWebservice, [73](#)
- CancelRequest, [73](#)
- CaptureScreenshot, [73](#)
- CreateForm, [74](#)
- DecryptAES, [74](#)
- defaultSocialPlatform, [80](#)
- dontDestroyOnLoad, [78](#)
- DownloadUrl, [74](#)
- EncryptAES, [75](#)
- EncryptMD5, [75](#)
- EncryptSHA1, [75](#)
- encryption, [78](#)
- GetServerInfo, [76](#)
- GetUrl, [76](#)
- instance, [80](#)
- isAuthenticated, [81](#)
- isCancelling, [81](#)
- isDownloading, [81](#)
- isInitialized, [81](#)
- language, [78](#)
- leaderboardUIFunction, [78](#)
- leaderboardUIObject, [79](#)
- localUser, [81](#)
- logDebugInfo, [79](#)
- onlineSeconds, [79](#)
- Ping, [76](#)
- pingIntervalSeconds, [79](#)
- platform, [81](#)
- playingSeconds, [79](#)
- rememberCredentials, [79](#)
- serverInfo, [82](#)
- sessionToken, [82](#)
- setAsDefaultSocialPlatform, [80](#)
- SetLocalUser, [77](#)
- urlRootProduction, [80](#)
- urlRootStage, [80](#)
- useStage, [80](#)
- Combu::CombuPlatform
 - Authenticate, [84, 85](#)
 - Authenticate< T >, [85](#)
 - CreateAchievement, [86](#)
 - CreateLeaderboard, [86](#)
 - GetLoading, [86](#)
 - LoadAchievementDescriptions, [87](#)
 - LoadAchievements, [87](#)
 - LoadAchievements< T >, [88](#)
 - LoadFriends, [88](#)
 - LoadScores, [89, 90](#)
 - LoadScoresByUser, [90](#)
 - LoadUsers, [91](#)
 - Logout, [91](#)
 - ReportProgress, [92](#)
 - ReportScore, [93, 94](#)
 - ResetAllAchievements, [94](#)
 - SetLocalUser, [94](#)
 - ShowAchievementsUI, [95](#)
 - ShowLeaderboardUI, [95](#)
- Combu::CombuServerInfo
 - ToString, [96](#)
- Combu::Inventory
 - Delete, [98](#)
 - FromHashtable, [98](#)
 - FromJson, [98](#)
 - Inventory, [97](#)
 - Load, [99](#)
 - Load< T >, [99](#)
 - Update, [99](#)
- Combu::Leaderboard
 - FromJson, [101](#)
 - Load, [101](#)
 - Load< T >, [102](#)
 - LoadScores, [102](#)
 - LoadScoresByUser, [102–104](#)
 - SetUserFilter, [104](#)
- Combu::Mail
 - Count, [106](#)
 - Delete, [106](#)
 - FromHashtable, [107](#)
 - FromJson, [107](#)
 - Load< T >, [107](#)
 - LoadConversations, [108](#)
 - Read, [108, 109](#)
 - Send, [109–111](#)
 - SendMailToGroup, [111](#)
 - Unread, [112](#)
- Combu::Match
 - AddUser, [114](#)
 - Delete, [115](#)
 - FromHashtable, [115](#)
 - FromJson, [116](#)
 - Load, [116](#)
 - Match, [114](#)
 - QuickMatch, [116](#)
 - RemoveUser, [117, 118](#)
 - Save, [118](#)
 - Score, [118](#)
- Combu::MatchAccount
 - FromHashtable, [119](#)
 - FromJson, [119](#)
- Combu::MatchRound
 - FromHashtable, [121](#)
 - FromJson, [121](#)
- Combu::MiniJSON
 - getLastErrorIndex, [123](#)
 - getLastErrorSnippet, [123](#)
 - jsonDecode, [123](#)
 - jsonEncode, [124](#)
 - lastDecodeSuccessful, [124](#)
 - lastErrorIndex, [124](#)
- Combu::News
 - FromHashtable, [125](#)
 - FromJson, [125](#)
 - Load, [126](#)
 - Load< T >, [126](#)

- Combu::Profile
 - appCustomData, 129
 - customData, 129
 - email, 129
 - FromHashtable, 129
 - FromJson, 129
 - id, 130
 - idLong, 130
 - image, 130
 - isFriend, 130
 - lastSeen, 130
 - Profile, 128
 - sessionToken, 130
 - state, 131
 - userName, 131
- Combu::Score
 - Initialize, 132
 - ReportScore, 133
- Combu::Tournament
 - Delete, 135, 136
 - FromHashtable, 136
 - FromJson, 136
 - Load, 136, 137
 - QuickTournament, 137
 - Save, 137
 - Tournament, 135
- Combu::User
 - AddContact, 140, 141
 - Authenticate, 141, 142
 - AuthenticatePlatform, 142
 - AutoLogin, 143
 - CanAutoLogin, 143
 - ChangePassword, 143, 144
 - CreateGuest, 144
 - Delete, 144, 145
 - Exists, 145
 - FromUser, 145
 - GetContact, 146
 - GetContact< T >, 146
 - LinkAccount, 146
 - LinkPlatform, 148
 - Load, 148–150
 - Load< T >, 150
 - LoadFriends, 150, 152
 - LoadFriends< T >, 152
 - Random< T >, 152
 - RemoveContact, 153, 154
 - ResetPassword, 154
 - Update, 155
- Combu::UserFile
 - Delete, 157
 - Download, 158
 - FromHashtable, 158
 - FromJson, 158
 - Like, 158, 159
 - Load, 159
 - Load< T >, 160
 - Update, 160
 - UserFile, 156, 157
 - View, 161
- Combu::UserGroup
 - Delete, 163
 - FromHashtable, 164
 - FromJson, 164
 - Join, 164, 165
 - Join< T >, 165
 - Leave, 166
 - Leave< T >, 167
 - Save, 167
 - UserGroup, 163
- Count
 - Combu::Mail, 106
- CreateAchievement
 - Combu::CombuPlatform, 86
- CreateForm
 - Combu::CombuManager, 74
- CreateGuest
 - Combu::User, 144
- CreateLeaderboard
 - Combu::CombuPlatform, 86
- customData
 - Combu::Profile, 129
- DecryptAES
 - Combu::CombuEncryption, 64
 - Combu::CombuManager, 74
- DecryptResponse
 - Combu::CombuEncryption, 65
- defaultSocialPlatform
 - Combu::CombuManager, 80
- Delete
 - Combu::Inventory, 98
 - Combu::Mail, 106
 - Combu::Match, 115
 - Combu::Tournament, 135, 136
 - Combu::User, 144, 145
 - Combu::UserFile, 157
 - Combu::UserGroup, 163
- dontDestroyOnLoad
 - Combu::CombuManager, 78
- Download
 - Combu::UserFile, 158
- DownloadUrl
 - Combu::CombuManager, 74
- eContactType
 - Combu, 60
- eLeaderboardInterval
 - Combu, 60
- eLeaderboardTimeScope
 - Combu, 60
- eMailList
 - Combu, 61
- eSearchOperator
 - Combu, 61
- email
 - Combu::Profile, 129

- EncryptAES
 - Combu::CombuEncryption, 65
 - Combu::CombuManager, 75
- EncryptMD5
 - Combu::CombuEncryption, 65
 - Combu::CombuManager, 75
- EncryptRSA
 - Combu::CombuEncryption, 66
- EncryptSHA1
 - Combu::CombuEncryption, 66
 - Combu::CombuManager, 75
- encryption
 - Combu::CombuManager, 78
- Exists
 - Combu::User, 145
- FromHashtable
 - Combu::Inventory, 98
 - Combu::Mail, 107
 - Combu::Match, 115
 - Combu::MatchAccount, 119
 - Combu::MatchRound, 121
 - Combu::News, 125
 - Combu::Profile, 129
 - Combu::Tournament, 136
 - Combu::UserFile, 158
 - Combu::UserGroup, 164
- FromJson
 - Combu::Inventory, 98
 - Combu::Leaderboard, 101
 - Combu::Mail, 107
 - Combu::Match, 116
 - Combu::MatchAccount, 119
 - Combu::MatchRound, 121
 - Combu::News, 125
 - Combu::Profile, 129
 - Combu::Tournament, 136
 - Combu::UserFile, 158
 - Combu::UserGroup, 164
- FromUser
 - Combu::User, 145
- GetBinaryField
 - Combu::CombuForm, 68
- GetContact
 - Combu::User, 146
- GetContact< T >
 - Combu::User, 146
- GetField
 - Combu::CombuForm, 69
- GetForm
 - Combu::CombuForm, 69
- getLastErrorIndex
 - Combu::MiniJSON, 123
- getLastErrorSnippet
 - Combu::MiniJSON, 123
- GetLoading
 - Combu::CombuPlatform, 86
- GetServerInfo
 - Combu::CombuManager, 76
- GetUrl
 - Combu::CombuManager, 76
- id
 - Combu::Profile, 130
- idLong
 - Combu::Profile, 130
- image
 - Combu::Profile, 130
- Initialize
 - Combu::Score, 132
- instance
 - Combu::CombuManager, 80
- Inventory
 - Combu::Inventory, 97
- isAuthenticated
 - Combu::CombuManager, 81
- isCancelling
 - Combu::CombuManager, 81
- isDownloading
 - Combu::CombuManager, 81
- isFriend
 - Combu::Profile, 130
- isInitialized
 - Combu::CombuManager, 81
- Join
 - Combu::UserGroup, 164, 165
- Join< T >
 - Combu::UserGroup, 165
- jsonDecode
 - Combu::MiniJSON, 123
- jsonEncode
 - Combu::MiniJSON, 124
- language
 - Combu::CombuManager, 78
- lastDecodeSuccessful
 - Combu::MiniJSON, 124
- lastErrorIndex
 - Combu::MiniJSON, 124
- lastSeen
 - Combu::Profile, 130
- leaderboardUIFunction
 - Combu::CombuManager, 78
- leaderboardUIObject
 - Combu::CombuManager, 79
- Leave
 - Combu::UserGroup, 166
- Leave< T >
 - Combu::UserGroup, 167
- Like
 - Combu::UserFile, 158, 159
- LinkAccount
 - Combu::User, 146
- LinkPlatform
 - Combu::User, 148
- Load

- Combu::Inventory, 99
- Combu::Leaderboard, 101
- Combu::Match, 116
- Combu::News, 126
- Combu::Tournament, 136, 137
- Combu::User, 148–150
- Combu::UserFile, 159
- Load< T >
 - Combu::Inventory, 99
 - Combu::Leaderboard, 102
 - Combu::Mail, 107
 - Combu::News, 126
 - Combu::User, 150
 - Combu::UserFile, 160
- LoadAchievementDescriptions
 - Combu::CombuPlatform, 87
- LoadAchievements
 - Combu::CombuPlatform, 87
- LoadAchievements< T >
 - Combu::CombuPlatform, 88
- LoadConversations
 - Combu::Mail, 108
- LoadFriends
 - Combu::CombuPlatform, 88
 - Combu::User, 150, 152
- LoadFriends< T >
 - Combu::User, 152
- LoadRSA
 - Combu::CombuEncryption, 66, 67
- LoadScores
 - Combu::CombuPlatform, 89, 90
 - Combu::Leaderboard, 102
- LoadScoresByUser
 - Combu::CombuPlatform, 90
 - Combu::Leaderboard, 102–104
- LoadUsers
 - Combu::CombuPlatform, 91
- localUser
 - Combu::CombuManager, 81
- logDebugInfo
 - Combu::CombuManager, 79
- Logout
 - Combu::CombuPlatform, 91
- Match
 - Combu::Match, 114
- onlineSeconds
 - Combu::CombuManager, 79
- Ping
 - Combu::CombuManager, 76
- pingIntervalSeconds
 - Combu::CombuManager, 79
- platform
 - Combu::CombuManager, 81
- playingSeconds
 - Combu::CombuManager, 79
- Profile
 - Combu::Profile, 128
- QuickMatch
 - Combu::Match, 116
- QuickTournament
 - Combu::Tournament, 137
- Random< T >
 - Combu::User, 152
- Read
 - Combu::Mail, 108, 109
- rememberCredentials
 - Combu::CombuManager, 79
- RemoveContact
 - Combu::User, 153, 154
- RemoveUser
 - Combu::Match, 117, 118
- ReportProgress
 - Combu::CombuPlatform, 92
- ReportScore
 - Combu::CombuPlatform, 93, 94
 - Combu::Score, 133
- ResetAllAchievements
 - Combu::CombuPlatform, 94
- ResetPassword
 - Combu::User, 154
- Save
 - Combu::Match, 118
 - Combu::Tournament, 137
 - Combu::UserGroup, 167
- Score
 - Combu::Match, 118
- Send
 - Combu::Mail, 109–111
- SendMailToGroup
 - Combu::Mail, 111
- serverInfo
 - Combu::CombuManager, 82
- sessionToken
 - Combu::CombuManager, 82
 - Combu::Profile, 130
- setAsDefaultSocialPlatform
 - Combu::CombuManager, 80
- SetLocalUser
 - Combu::CombuManager, 77
 - Combu::CombuPlatform, 94
- SetToken
 - Combu::CombuEncryption, 67
- SetUserFilter
 - Combu::Leaderboard, 104
- ShowAchievementsUI
 - Combu::CombuPlatform, 95
- ShowLeaderboardUI
 - Combu::CombuPlatform, 95
- state
 - Combu::Profile, 131
- ToString

- Combu::CombuServerInfo, [96](#)
- Tournament
 - Combu::Tournament, [135](#)
- Unread
 - Combu::Mail, [112](#)
- Update
 - Combu::Inventory, [99](#)
 - Combu::User, [155](#)
 - Combu::UserFile, [160](#)
- urlRootProduction
 - Combu::CombuManager, [80](#)
- urlRootStage
 - Combu::CombuManager, [80](#)
- useStage
 - Combu::CombuManager, [80](#)
- UserFile
 - Combu::UserFile, [156](#), [157](#)
- UserGroup
 - Combu::UserGroup, [163](#)
- userName
 - Combu::Profile, [131](#)
- View
 - Combu::UserFile, [161](#)