Combu

2.1.14

Generated by Doxygen 1.8.11

Contents

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 2.x** only. The documentation for **version 1.x** is shipped as PDF(s) within the package purchased.

The 2nd generation of **Combu** marks a great step towards the optimal organization both for the ease of use and for a better integration within Unity system. The new version of Combu implements the Social API of Unity, this way most of the standard code will work the same way as any other ISocialPlatform.

1.2 Installation

1. **Purchase Combu**
   You can purchase Combu from **Skared Creations** website or Unity3D Asset Store.
2. **Setup the web server**
   You can use a local web server before going live to production. Read the FAQ to learn how to setup the web server.
3. **Import Combu package**
   Create an empty scene and import the Combu unitypackage (or from Asset Store window).
4. **Configure Combu Manager**
   Drag the prefab *Combu/Prefabs/CombuManager* in the scene and correct settings of the **CombuManager** component in order to connect to your web server.

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 get started visit Related Pages.

1.3 Documentation

You can download the API documentation as offline PDF or navigate online at http://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'll only need to create the database on MySQL (and execute the file *combu_db.sql* into it) and edit the file *lib/config.php* in your **Combu** folder (that is the constants *GAME_DB_SERVER*, *GAME_DB_NAME*, *GAME_DB_USER* and *GAME_DB_PASS*), as explained in the `tutorial` video.

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. Edit the file **php.ini** (if you're using XAMPP then it's usually located in the folder *xamppfiles/etc*) and assign the value *On* to the variable *short_open_tag* (the line should look like: **short_open_tag=On**)
3. **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)
4. Start both HTTP and MySQL services (if you're using XAMPP, it can be done through the XAMPP Control Panel else from Windows Services)
5. Create an empty database with name **combu**, select/open it and execute the file **combu_db.sql** in phpMyAdmin (in XAMPP it's usually installed at `http://localhost/phpmyadmin`) or `MySQL Workbench`
6. 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*); if the database created in the previous step is not named *combu* then you'll need to edit the file **lib**/**config.php** and modify the value of the constant *GAME_DB_NAME*

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. `Click here` to get started.

Chapter 4

Server and Client configuration

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

4.1 Server configuration

The server setup is stored in the file **lib/config.php** as PHP defines (if you are new to PHP, "define" is the equivalent of "const" in C#):

- **URL_ROOT**: it is the absolute URL path to the root of Combu (e.g.: */combu/*)
- **DEFAULT_TIMEZONE**: this property has been **deprecated** in 2.1.10 (timestamps are saved in UTC)
- **GAME_DB_SERVER**: it is the hostname or IP address of MySQL server
- **GAME_DB_NAME**: it is name of MySQL database
- **GAME_DB_USER**: it is the user name of MySQL connection
- **GAME_DB_PASS**: it is the user password of MySQL connection
- **REGISTER_EMAIL_REQUIRED**: it will require a valid email address upon new user creation
- **REGISTER_EMAIL_MULTIPLE**: it allows to use the same email address for multiple accounts
- **REGISTER_EMAIL_ACTIVATION**: 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**: the subject text of the user registration email
- **REGISTER_EMAIL_MESSAGE**: the full path to the html/text fle that contains the text of the user registration email; the message text 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
- **FRIENDS_REQUIRE_ACCEPT**: 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
- **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
- **EMAIL_SENDER_ADDRESS**: the sender address of outgoing mail
- **EMAIL_SENDER_NAME**: the sender name of outgoing mail
- **NEWSLETTER_SENDER_ADDRESS**: the sender address of outgoing newsletters
- **NEWSLETTER_SENDER_NAME**: the sender name of outgoing newsletters
- **DEFAULT_LIST_LIMIT**: it sets the default number of results fetched for pagination
- **LOG_FILEPATH**: it is the physical path to the log fle used by the class AppLog
- **LOG_MAXFILESIZE**: it is the maximum size in bytes of the log fle
- **URL_UPLOAD**: root folder for uploads to be used as URL
- **UPLOAD**: physical root folder for uploads
- **GUEST_PREFIX**: the prefix string attached to the id for guest accounts (ex.: "Guest-")
- **EXPIRE_CLIENT_SESSION**: expire account sessions (must be a valid parameter for DateInterval constructor: http://php.net/manual/en/class.dateinterval.php)

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**: 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 2.x 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**
- **Secret Key**: it is used to secure your webservices from spoofng attacks by signing each request; if you haven't a secure SSL connection on your server then we suggest to use a secret key, it must be the same as you defne on your webservice confg.php (please read the Server Documentation to defne one on your server)
- **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 window
- **Ping Interval Seconds**: 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**: 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**: it's the time in seconds from last action to be considered as Playing
- **Timezone**: this property has been **deprecated** in 2.1.10 (timestamps are saved in UTC)

- **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

Authentication and Users

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

5.1 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);
});
```

5.1.1 Authenticate a saved session

If you want to auto-login a previously saved session then you need to call **AuthenticateSession** on a **User** object:

```
string sessionToken = PlayerPrefs.GetString("SessionToken", "");
long userId = 0;
if (!long.TryParse (PlayerPrefs.GetString ("UserId", "0"), out userId) || userId < 0)
    userId = 0;
User.AuthenticateSession (userId, sessionToken, (bool success, string error) => {
    Debug.Log ("AuthenticateSession - Success=" + success + " > Error=" + error);
});
```

5.2 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);
});
```

5.2.1 Create a guest account

If you want to create a guest account then you need to call **CreateGuest** on a **User** object:

```
var user = new User ();
user.CreateGuest ((bool success, string error) => {
    Debug.Log ("CreateGuest - Success=" + success + " > Error=" + error);
    if (success) {
        // Store the id and sessionToken
        PlayerPrefs.SetString ("UserId", CombuManager.localUser.id);
        PlayerPrefs.SetString ("SessionToken", CombuManager.localUser.sessionToken);
        PlayerPrefs.Save ();
    }
});
```

5.3 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
      pagesCount) => {
    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);
        }
    }
});
```

5.4 Custom Data

You can store any type of data in the **Hashtable** *customData* of **Profile** class, for example you could store the user's virtual currency, level, experience in a RPG game:

```
CombuManager.localUser.customData["Coins"] = 100;
CombuManager.localUser.Update( (bool success, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```

5.5 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 mantain the online state CombuManager uses the settings *pingIntervalSeconds*, *onlineSeconds* and *playing↩ Seconds*. 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 webservices updates the "last action" date/time of a user.

5.6 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["my↩ Property"]* (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 6

Server Settings

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

6.1 Server

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

6.2 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 7

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.

7.1 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);
});
```

7.2 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.Link↩ Platform**:

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

7.3 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);
});
```

Chapter 8

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.

8.1 Loading Contacts

To retrieve the list of contacts from the local user you need to call the *LoadFriends* method on **CombuManager.localUser**:

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

8.2 Adding Contacts

To add another user to a contact list of the local user you need to call *AddContact*:

```
// 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);
});
```

8.3 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 9

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.

9.1 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);
});
```

9.2 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);
});
```

9.3 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);
});
```

9.4 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);
});
```

9.5 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 10

Managing Inventory

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

10.1 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);
});
```

10.2 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);
});
```

10.3 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 11

Managing Messages

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

11.1 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 pagesCount, string error) =>
      {
    if (string.IsNullOrEmpty(error))
        Debug.Log("Success: " + messages.Length);
    else
        Debug.Log("Failed: " + error);
});
```

11.2 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);
});
```

11.2.1 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);
});
```

11.3 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);
});
```

11.4 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);
});
```

11.5 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);
});
```

11.6 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);
});
```

11.7 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 12

Managing User Groups

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

12.1 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);
});
```

12.2 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);
});
```

12.3 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);
});
```

12.4 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 13

Managing News

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

13.1 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 pagesCount, string error) => {
    if (string.IsNullOrEmpty(error))
        Debug.Log("Success: " + news.Length);
    else
        Debug.Log("Failed: " + error);
});
```

Chapter 14

Managing Leaderboards

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

14.1 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
Leaderboard 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
});
```

14.2 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, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
// 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, string error) => {
    if (success)
        Debug.Log("Success");
    else
        Debug.Log("Failed: " + error);
});
```

14.3 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 (success)
        Debug.Log("Success: " + leaderboard.title);
    else
        Debug.Log("Failed: " + error);
});
```

14.4 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
      attendible '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 (success)
        Debug.Log("Success: " + score.value + " at page " + page);
    else
        Debug.Log("Failed: " + error);
});
```

Chapter 15

Managing Achievements

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

15.1 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);
});
```

15.2 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 16

Managing Tournaments

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

16.1 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);
});
```

16.1.1 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");
});
```

16.1.2 Matches of the Tournament

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

16.2 Quick Tournament

To create a quick tournament with other users user 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);

        });
    }
});
```

16.2.1 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.

16.3 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 17

Managing Matches

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

17.1 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);
});
```

17.1.1 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");
});
```

17.1.2 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).

17.2 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");
});
```

17.2.1 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.

17.3 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);
});
```

17.4 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 18

Namespace Index

18.1 Packages

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

Chapter 19

Hierarchical Index

19.1 Class Hierarchy

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

Chapter 20

Class Index

20.1 Class List

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

Chapter 21

Namespace Documentation

21.1 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 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

  Combu server info.
- 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

  Search custom data.
- 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 { **Friend** = 0, **Request**, **Ignore** }

  Contact type.
- enum eMailList : int { **Received** = 0, **Sent**, **Both** }

  Mail list.
- 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.

21.1.1 Detailed Description

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

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

21.1.2 Enumeration Type Documentation

21.1.2.1 enum Combu.eContactType : int [strong]

Contact type.

21.1.2.2 enum Combu.eLeaderboardInterval : int [strong]

Leaderboard interval.

21.1.2.3 enum Combu.eLeaderboardTimeScope : int [strong]

Leaderboard time scope.

21.1.2.4 enum Combu.eMailList : int [strong]

Mail list.

21.1.2.5 enum Combu.eSearchOperator [strong]

Custom search operator.

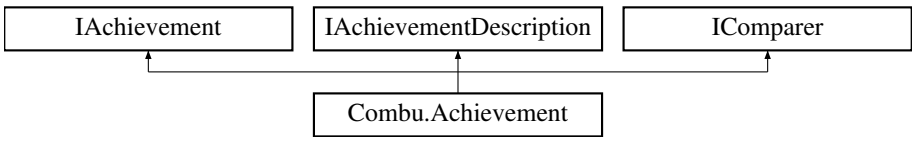
Chapter 22

Class Documentation

22.1 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]`

22.1.1 Detailed Description

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

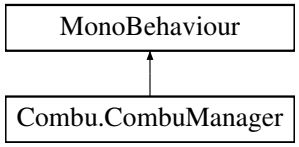
The documentation for this class was generated from the following file:

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/Achievement.cs

22.2 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).
- WWWForm CreateForm (User user=null)

  Creates a new form to be passed to a webservice.
- string GetUrl (string relativeUrl)

  Gets the absolute URL from a relative.
- void GetServerInfo (Action< bool, CombuServerInfo > callback)

  Gets the server info.
- 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.

Public Attributes

- const string COMBU_VERSION = "2.1.14"

  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 secretKey

  *The secret key: it must match the define **SECRET_KEY** configured on the web.*
- 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.
- float pingIntervalSeconds = 30f

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

  Can be used to filter the current system date timezone (the value must be valid in PHP: `http://www.php.net/manual/en/timezones.php`).
- 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** ()
- IEnumerator DownloadUrl (string url, WWWForm form, Action< string, string > onComplete)

  Cancels all the current requests (immediately).
- void SecureRequest (WWWForm form)

  Secure the WWWForm by signing the request.

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.

Properties

- 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.

22.2.1 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.

22.2.2 Member Function Documentation

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

Calls a webservice.

Parameters

url	URL.
form	Form.
onComplete	On complete callback.

22.2.2.2 void Combu.CombuManager.CancelRequest ( )

Cancels the current request (next frame).

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

Captures the screen shot.

Returns

The screen shot.

Parameters

thumbnailHeight	Thumbnail height.
excludeCams	Exclude cams.

22.2.2.4 WWWForm Combu.CombuManager.CreateForm (User *user* = null)

Creates a new form to be passed to a webservice.

Returns

The form.

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

Cancels all the current requests (immediately).

Downloads the content of an URL with the specified form.

Returns

The URL.

Parameters

url	URL.
form	Form.
onComplete	On complete.

22.2.2.6 static string Combu.CombuManager.EncryptMD5 (string *inputString*) [static]

Encrypts a string in MD5.

Returns

The M d5.

Parameters

inputString	Input string.

22.2.2.7 static string Combu.CombuManager.EncryptSHA1 (string *inputString*) [static]

Encrypts a string in SHA1.

Returns

The SH a1.

Parameters

inputString	Input string.

22.2.2.8 void Combu.CombuManager.GetServerInfo (Action< bool, **CombuServerInfo** > *callback*)

Gets the server info.

Parameters

callback	Callback.

22.2.2.9 string Combu.CombuManager.GetUrl (string *relativeUrl*)

Gets the absolute URL from a relative.

Returns

The URL.

Parameters

relativeUrl	Relative URL.

22.2.2.10 void Combu.CombuManager.Ping (bool *onlyIfAuthenticated* = `true`, Action< bool > *callback* = `null`)

Ping the server.

Parameters

onlyIfAuthenticated	If set to `true` then it runs only if local user is authenticated.
callback	Callback.

22.2.2.11 void Combu.CombuManager.SecureRequest (WWWForm *form*) [protected]

Secure the WWWForm by signing the request.

Parameters

form	Form.

22.2.3 Member Data Documentation

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

The singleton instance of CombuManager

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

The singleton instance of CombuPlatform.

22.2.3.3 string Combu.CombuManager.achievementUIFunction

The achievement user interface function for CombuPlatform.ShowAchievementsUI().

22.2.3.4 GameObject Combu.CombuManager.achievementUIObject

The achievement user interface object for CombuPlatform.ShowAchievementsUI().

22.2.3.5 const string Combu.CombuManager.COMBU_VERSION = "2.1.14"

Current API version.

22.2.3.6 bool Combu.CombuManager.dontDestroyOnLoad = true

Should call DontDestroyOnLoad on the CombuManager gameObject? Recommended: set to true

22.2.3.7 string Combu.CombuManager.leaderboardUIFunction

The leaderboard user interface function for CombuPlatform.ShowLeaderboardUI().

22.2.3.8 GameObject Combu.CombuManager.leaderboardUIObject

The leaderboard user interface object for CombuPlatform.ShowLeaderboardUI().

22.2.3.9 bool Combu.CombuManager.logDebugInfo

Print debug info in the console log.

22.2.3.10 int Combu.CombuManager.onlineSeconds = 120

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

22.2.3.11 float Combu.CombuManager.pingIntervalSeconds = 30f

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

22.2.3.12 int Combu.CombuManager.playingSeconds = 120

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

22.2.3.13 string Combu.CombuManager.secretKey

The secret key: it must match the define **SECRET_KEY** configured on the web.

22.2.3.14 bool Combu.CombuManager.setAsDefaultSocialPlatform

Should set Combu as the active social platform? The previous platform is accessible from defaultSocialPlatform

22.2.3.15 string Combu.CombuManager.timezone

Can be used to filter the current system date timezone (the value must be valid in PHP: http://www.php.↩net/manual/en/timezones.php).

22.2.3.16 string Combu.CombuManager.urlRootProduction

The URL root for the production environment.

22.2.3.17 string Combu.CombuManager.urlRootStage

The URL root for the stage environment.

22.2.3.18 bool Combu.CombuManager.useStage

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

22.2.4 Property Documentation

22.2.4.1 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.

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

Gets the current singleton instance.

The instance.

22.2.4.3 bool Combu.CombuManager.isAuthenticated [get]

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

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

22.2.4.4 bool Combu.CombuManager.isCancelling [get]

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

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

22.2.4.5 bool Combu.CombuManager.isDownloading [get]

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

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

22.2.4.6 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`.

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

Gets the local user.

The local user.

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

Gets the Combu ISocialPlatform implementation.

The platform.

22.2.4.9 CombuServerInfo Combu.CombuManager.serverInfo [get]

Gets the server info.

The server info.

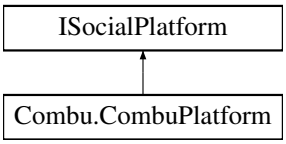
The documentation for this class was generated from the following file:

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/CombuManager.cs

22.3 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.
- 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.

Properties

- ILocalUser **localUser** `[get]`

22.3.1 Detailed Description

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

22.3.2 Member Function Documentation

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

Authenticates the user.

Parameters

user	User.
callback	Callback.

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

Authenticates the user with specified username and password.

Parameters

username	Username.
password	Password.
callback	Callback.

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

Authenticate the specified user.

Parameters

user	User.
callback	Callback.

22.3.2.4 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

username	Username.
password	Password.
callback	Callback.

Template Parameters

T	Type of the returned profiles.

Type Constraints

T* : *User

T* : *new()

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

Creates the achievement.

Returns

The achievement.

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

Creates the leaderboard.

Returns

The leaderboard.

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

Gets the loading state of a Leaderboard.

Returns

`true`, if loading was gotten, `false` otherwise.

Parameters

board	Board.

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

Loads the achievement descriptions.

Parameters

callback	Callback.

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

Loads the achievements.

Parameters

callback	Callback.

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

Loads the achievements.

Parameters

callback	Callback.

Template Parameters

T	The 1st type parameter.

Type Constraints

T* : *Achievement

T* : *new()

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

Loads the friends of localUser.

Parameters

user	User.
callback	Callback.

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

Loads the scores of a Leaderboard.

Parameters

leaderboardID	Leaderboard Id.
callback	Callback.

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

Loads the scores of a Leaderboard.

Parameters

leaderboardID	Leaderboard Id.
page	Page.
countPerPage	Count per page.
callback	Callback.

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

Loads the scores of a Leaderboard.

Parameters

board	Board.
callback	Callback.

22.3.2.15 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

leaderboardId	Leaderboard identifier.
user	User.
interval	Interval.
callback	Callback.

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

Loads the users by Id.

Parameters

userIDs	User I ds.
callback	Callback.

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

Logout localUser.

Parameters

callback	Callback.

22.3.2.18 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

achievementID	Achievement identifier.
progress	Progress.
callback	Callback.

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

Reports the progress of an Achievement.

Parameters

achievement↩ Id	Achievement identifier.
progress	Progress.
callback	Callback.

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

Reports the score of a Leaderboard.

Parameters

score	Score.
board	Board.
callback	Callback.

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

Reports the score of a Leaderboard.

Parameters

score	Score.
board	Board.
callback	Callback.

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

Reports the score of a Leaderboard.

Parameters

score	Score.
board	Board.
username	Username.
callback	Callback.

22.3.2.23 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

user	User.

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

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

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

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

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/CombuPlatform.cs

22.4 Combu.CombuServerInfo Class Reference

Combu server info.

Public Member Functions

- **CombuServerInfo** (Hashtable data)
- override string **ToString** ()

Public Attributes

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

22.4.1 Detailed Description

Combu server info.

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/CombuManager.cs

22.5 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]`

22.5.1 Constructor & Destructor Documentation

22.5.1.1 Combu.Inventory.Inventory ( )

Initializes a new instance of the Inventory class.

22.5.1.2 Combu.Inventory.Inventory (string *jsonString*)

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

Parameters

jsonString	JSON formatted string.

22.5.1.3 Combu.Inventory.Inventory (Hashtable *hash*)

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

Parameters

hash	Hash.

22.5.2 Member Function Documentation

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

Delete this inventory item from server.

Parameters

callback	Callback.

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

Delete the specified inventory item from server.

Parameters

idInventory	Identifier inventory.
callback	Callback.

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

Initialize the object from a hashtable.

Parameters

hash	Hash.

22.5.2.4 virtual void Combu.Inventory.FromJson (string *jsonString*) [virtual]

Initialize the object from a JSON formatted string.

Parameters

jsonString	Json string.

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

Load the inventory items of a User.

Parameters

userId	User identifier.
callback	Callback.

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

Load the inventory of a User.

Parameters

userId	User identifier.
callback	Callback.

Template Parameters

T	The 1st type parameter.

Type Constraints

T* : *Inventory

T* : *new()

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

Update this inventory item to server.

Parameters

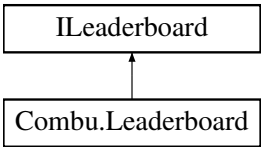
callback	Callback.

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/Inventory.cs

22.6 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 by user.

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.

Public Attributes

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

Properties

- bool **loading** `[get]`
- string **id** `[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]`

22.6.1 Member Function Documentation

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

Initialize the object from a JSON formatted string.

Parameters

jsonString	Json string.

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

Load the specified leaderboardId.

Parameters

leaderboardId	Leaderboard identifier.
callback	Callback.

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

Load the specified leaderboardId.

Parameters

leaderboardId	Leaderboard identifier.
callback	Callback.

Template Parameters

T	The 1st type parameter.

Type Constraints

T* : *Leaderboard

T* : *new()

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

Loads the scores.

Parameters

callback	Callback.

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

Loads the scores by user.

Parameters

user	User.
interval	Interval.
callback	Callback.

22.6.1.6 void Combu.Leaderboard.SetUserFilter (string[ ] *userIDs*)

Sets the user filter.

Parameters

userIDs	User I ds.

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/Leaderboard.cs

22.7 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]`

22.7.1 Member Function Documentation

22.7.1.1 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.

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

Delete this instance.

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

Delete the specified Mail.

Parameters

idMail	Identifier mail.
callback	Callback.

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

Initialize the object from a hashtable.

Parameters

hash	Hash.

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

Initialize the object from a JSON formatted string.

Parameters

jsonString	Json string.

22.7.1.6 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

pageNumber	Page number.
limit	Limit.

Template Parameters

T	Type of returned objects.

Type Constraints

T* : *Mail

T* : *new()

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

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

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

Mark this mail as read.

Parameters

callback	Callback.

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

Mark a single mail as read.

Parameters

idMail	Identifier mail.
callback	Callback.

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

Mark a set of mails as read.

Parameters

idSenders	Identifier senders.
idGroups	Identifier groups.

22.7.1.11 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

idMail	Identifier mail.
idSenders	Identifier senders.
idGroups	Identifier groups.
callback	Callback.

22.7.1.12 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

idDestination	Identifier Id of the destination user.
subject	Subject.
message	Message.
isPublic	If set to `true` is public.

22.7.1.13 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

recipients↩ Id	Recipients identifier.
subject	Subject.
message	Message.
isPublic	If set to `true` is public.

22.7.1.14 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

usernameDestination	Username of the destination user.
subject	Subject.
message	Message.
isPublic	If set to `true` is public.

22.7.1.15 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

recipientsUsername	Recipients username.
subject	Subject.
message	Message.
isPublic	If set to `true` is public.

22.7.1.16 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

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

22.7.1.17 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

groupId	Group identifier.
subject	Subject.
message	Message.
isPublic	If set to `true` is public.

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

Mark this mail as read.

Parameters

callback	Callback.

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

Mark a single mail as read.

Parameters

idMail	Identifier mail.
callback	Callback.

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/Mail.cs

22.8 Combu.MailCount Class Reference

Public Member Functions

- **MailCount** (Hashtable hash)

Public Attributes

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

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/Mail.cs

22.9 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 Matchs 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]

22.9.1 Constructor & Destructor Documentation

22.9.1.1 Combu.Match.Match (string *jsonString*)

Initializes a new instance of the Combu.Match class.

Parameters

jsonString	JSON string to initialize the instance.

22.9.1.2 Combu.Match.Match (Hashtable *data*)

Initializes a new instance of the Combu.Match class.

Parameters

data	Data to initialize the instance.

22.9.2 Member Function Documentation

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

Adds the user to this match.

Parameters

user	User.

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

Delete this instance from the database.

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

Delete the specified Match.

Parameters

idMatch	Identifier match.
callback	Callback.

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

Initialize the object from a hashtable.

Parameters

hash	Hash.

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

Initialize the object from a JSON formatted string.

Parameters

jsonString	Json string.

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

Load the list of Matchs by specified filters.

Parameters

idTournament	Identifier tournament.
activeOnly	If set to `true` then displays active matches only, else archived matches.
title	Title.
callback	Callback.

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

Load the specified Match.

Parameters

idMatch	Identifier match.
callback	Callback.

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

Creates a quick match.

Parameters

callback	Callback.

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

Removes the user.

Parameters

user	User.

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

Removes the user.

Parameters

idUser	Identifier user.

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

Removes the user.

Parameters

username	Username.

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

Removes the user.

Parameters

idUser	Identifier user.
username	Username.

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

Save the this instance in the server.

Parameters

callback	Callback.

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

Send the specified score.

Parameters

score	Score.
callback	Callback.

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/Match.cs

22.10 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]`

22.10.1 Member Function Documentation

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

Initialize the object from a hashtable.

Parameters

hash	Hash.

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

Initialize the object from a JSON formatted string.

Parameters

jsonString	Json string.

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/MatchAccount.cs

22.11 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]

22.11.1 Member Function Documentation

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

Initialize the object from a hashtable.

Parameters

hash	Hash.

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

Initialize the object from a JSON formatted string.

Parameters

jsonString	Json string.

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/MatchRound.cs

22.12 Combu.Match.MatchRoundData Class Reference

Public Attributes

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

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/Match.cs

22.13 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** = ""

22.13.1 Member Function Documentation

22.13.1.1 static int Combu.MiniJSON.getLastErrorIndex ( ) [static]

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

Returns

22.13.1.2 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

22.13.1.3 static object Combu.MiniJSON.jsonDecode (string *json*) [static]

Parses the string json into a value

Parameters

json	A JSON string.

Returns

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

22.13.1.4 static string Combu.MiniJSON.jsonEncode (object *json*) [static]

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

Parameters

json	A Hashtable / ArrayList

Returns

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

22.13.1.5 static bool Combu.MiniJSON.lastDecodeSuccessful ( ) [static]

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

Returns

22.13.2 Member Data Documentation

22.13.2.1 int Combu.MiniJSON.lastErrorIndex = -1 [static],[protected]

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

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/MiniJSON.cs

22.14 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** = ""

22.14.1 Member Function Documentation

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

Initialize the object from a hashtable.

Parameters

hash	Hash.

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

Initialize the object from a JSON formatted string.

Parameters

jsonString	Json string.

22.14.1.3 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

pageNumber	Page number.
limit	Limit.
callback	Callback.

22.14.1.4 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

pageNumber	Page number.
limit	Limit.
callback	Callback.

Template Parameters

T	The 1st type parameter.

Type Constraints

T* : *News

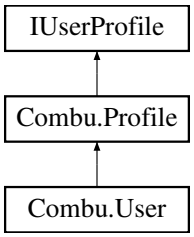
T* : *new()

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/News.cs

22.15 Combu.Profile Class Reference

Inheritance diagram for Combu.Profile:



Public Member Functions

- Profile (string jsonString)

  Initializes a new instance of the CBUser class from a JSON formatted string.
- Profile (Hashtable hash)

  Initializes a new instance of the CBUser 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 custom data.

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.

22.15.1 Constructor & Destructor Documentation

22.15.1.1 Combu.Profile.Profile (string *jsonString*)

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

Parameters

jsonString	JSON formatted string.

22.15.1.2 Combu.Profile.Profile (Hashtable *hash*)

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

Parameters

hash	Hash.

22.15.2 Member Function Documentation

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

Initialize the object from a hashtable.

Parameters

hash	Hash.

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

Initialize the object from a JSON formatted string.

Parameters

jsonString	Json string.

22.15.3 Member Data Documentation

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

The custom data.

22.15.3.2 string Combu.Profile.email

The email address.

22.15.4 Property Documentation

22.15.4.1 string Combu.Profile.id [get]

Gets the identifier value as string.

The identifier.

22.15.4.2 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.

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

Gets or sets the image.

The image.

22.15.4.4 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`.

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

Gets the last seen date/time.

The last seen.

22.15.4.6 string Combu.Profile.sessionToken [get]

Gets the session token.

The session token.

22.15.4.7 virtual UserState Combu.Profile.state [get]

Gets the online state.

The state.

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

Gets or sets the name of the user.

The name of the user.

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/Profile.cs

22.16 Combu.ProfilePlatform Class Reference

Public Member Functions

- **ProfilePlatform** (Hashtable data)

Public Attributes

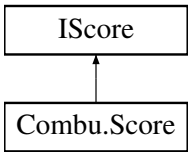
- string **platformKey** = ""
- string **platformId** = ""

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/ProfilePlatform.cs

22.17 Combu.Score Class Reference

Inheritance diagram for Combu.Score:



Public Member Functions

- **Score** (string idLeaderboard, 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 ReportScore (System.Action< bool > callback)

  Reports the score.

Properties

- string **leaderboardID** `[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]`

22.17.1 Member Function Documentation

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

Initialize this Score with the specified idLeaderboard, rank, user and score.

Parameters

idLeaderboard	Identifier leaderboard.
rank	Rank.
user	User.
score	Score.

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

Reports the score.

Parameters

callback	Callback.

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/Leaderboard.cs

22.18 Combu.SearchCustomData Class Reference

Search custom data.

Public Member Functions

- **SearchCustomData** (string key, eSearchOperator op, string value)

Public Attributes

- string **key**
- eSearchOperator **op**
- string **value**

22.18.1 Detailed Description

Search custom data.

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/CombuManager.cs

22.19 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]`

22.19.1 Detailed Description

Tournaments class.

22.19.2 Constructor & Destructor Documentation

22.19.2.1 Combu.Tournament.Tournament (string *jsonString*)

Initializes a new instance of the Combu.Tournament class.

Parameters

jsonString	JSON string to initialize the instance.

22.19.2.2 Combu.Tournament.Tournament (Hashtable *data*)

Initializes a new instance of the Combu.Tournament class.

Parameters

data	Data to initialize the instance.

22.19.3 Member Function Documentation

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

Delete this instance from the database.

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

Delete the specified Tournament.

Parameters

idTournament	Identifier tournament.
callback	Callback.

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

Initialize the object from a hashtable.

Parameters

hash	Hash.

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

Initialize the object from a JSON formatted string.

Parameters

jsonString	Json string.

22.19.3.5 static void Combu.Tournament.Load (bool *finished,* SearchCustomData[ ] *customData,* System.Action< Tournament[ ]> *callback*) [static]

Load the list of tournaments with the specified filters.

Parameters

finished	If set to `true` then it returns also the finished tournaments.
customData	Custom data.
callback	Callback.

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

Load the tournament with the specified id.

Parameters

id	Identifier.
callback	Callback.

22.19.3.7 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

users	The list of other users (excluding localUser).

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

Save this instance in the database.

Parameters

callback	Callback.

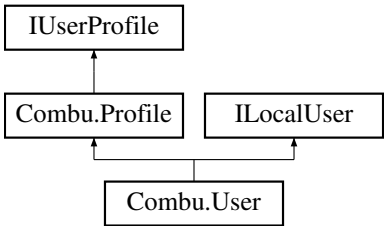
The documentation for this class was generated from the following file:

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/Tournament.cs

22.20 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)
- 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)
- 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 AddContact (string otherUsername, 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 (Profile otherUser, System.Action< bool, string > callback)

  Removes the contact.

Static Public Member Functions

- static void **AuthenticateSession** (long userId, string token, 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.

Public Attributes

- string **password**

Properties

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

Additional Inherited Members

22.20.1 Detailed Description

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

22.20.2 Member Function Documentation

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

Adds the contact.

Parameters

otherUsername	Other username.
contactType	Contact type.
callback	Callback.

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

Adds the contact.

Parameters

otherUser	Other user.
contactType	Contact type.
callback	Callback.

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

Authenticate the user.

Parameters

callback	Callback.

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

Authenticate the user.

Parameters

callback	Callback.

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

Authenticate the user with the specified password.

Parameters

password	Password.
callback	Callback.

Template Parameters

T	Type for User.

22.20.2.6 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

platformKey	Platform key.
platformId	Platform identifier.
callback	Callback.

22.20.2.7 virtual void Combu.User.ChangePassword (string *newPassword,* System.Action< bool, string > *callback*) [virtual]

Changes the password of this user.

Parameters

newPassword	New password.
callback	Callback.

22.20.2.8 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

idUser	Identifier user.
username	Username.
resetCode	Reset code.
newPassword	New password.
callback	Callback.

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

Delete this instance from the server.

Parameters

callback	Callback.

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

Delete a user from the server.

Parameters

username	Username.
password	Password.
callback	Callback.

22.20.2.11 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

username	Username.
email	Email.
callback	Callback.

22.20.2.12 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

username	Username.
password	Password.
callback	Callback.

22.20.2.13 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

platformKey	Platform key.
platformId	Platform identifier.
callback	Callback.

22.20.2.14 static void Combu.User.Load (User[] *updateUsers,* System.Action< bool > *callback*) [static]

Reload the specified users from server (by Id).

Parameters

updateUsers	List of users.
callback	Callback.

22.20.2.15 void Combu.User.Load (System.Action< bool > *callback*)

Load of the current user from server.

Parameters

callback	Callback.

22.20.2.16 static void Combu.User.Load (long *userId,* System.Action< User > *callback*) [static]

Loads a user by Id.

Parameters

userId	User Id.
callback	Callback.

22.20.2.17 static void Combu.User.Load (string *userName,* System.Action< User > *callback*) [static]

Loads a user by userName.

Parameters

userName	User Name.
callback	Callback.

22.20.2.18 static void Combu.User.Load (long[ ] *userIds,* System.Action< User[ ]> *callback*) [static]

Loads the users by Id.

Parameters

userIds	User Ids.
callback	Callback.
updateUser	If passed its data will be replaced with the server result.

22.20.2.19 static void Combu.User.Load (string[ ] *userNames,* System.Action< User[ ]> *callback*) [static]

Loads the users by userName.

Parameters

userNames	User Names.
callback	Callback.

22.20.2.20 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

username	Username.
email	Email.
customData	Custom data.
pageNumber	Page number.
limit	Limit.
callback	Callback.

Type Constraints

T* : *User

T* : *new()

22.20.2.21 virtual void Combu.User.LoadFriends (System.Action< bool > *callback*) [virtual]

Loads the friends of the current logged user.

Parameters

callback	Callback.

22.20.2.22 virtual void Combu.User.LoadFriends (eContactType *contactType,* System.Action< bool > *callback*) [virtual]

Loads the friends of the current logged user.

Parameters

contactType	Contact type.
callback	Callback.

22.20.2.23 virtual void Combu.User.LoadFriends< T > (eContactType *contactType,* System.Action< bool > *callback*) [virtual]

Loads the friends of the current logged user.

Parameters

contactType	Contact type.
callback	Callback.

Template Parameters

T	Type for User.

Type Constraints

T* : *User

T* : *new()

22.20.2.24 static void Combu.User.Random< T > (SearchCustomData[ ] *customData,* int *count,* System.Action< T[ ]> *callback*) [static]

Loads a specified count of random users.

Parameters

customData	Custom data.
count	Count.
callback	Callback.

Type Constraints

T* : *User

T* : *new()

22.20.2.25 void Combu.User.RemoveContact (string *otherUsername,* System.Action< bool, string > *callback*)

Removes the contact.

Parameters

otherUsername	Other username.
callback	Callback.

22.20.2.26 void Combu.User.RemoveContact (Profile *otherUser,* System.Action< bool, string > *callback*)

Removes the contact.

Parameters

otherUser	Other user.
callback	Callback.

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

Resets the password of this user.

Parameters

callback	Callback.

22.20.2.28 static void Combu.User.ResetPassword (long *idUser,* System.Action< bool, string > *callback*) [static]

Resets the password of a user by Id Account.

Parameters

idUser	Identifier user.
callback	Callback.

22.20.2.29 static void Combu.User.ResetPassword (string *username,* System.Action< bool, string > *callback*) [static]

Resets the password of a user by Username.

Parameters

username	Username.
callback	Callback.

22.20.2.30 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

callback	Callback.

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/User.cs

22.21 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]`

22.21.1 Constructor & Destructor Documentation

22.21.1.1 Combu.UserFile.UserFile ( )

Initializes a new instance of the UserFile class.

22.21.1.2 Combu.UserFile.UserFile (string *jsonString*)

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

Parameters

jsonString	JSON formatted string.

22.21.1.3 Combu.UserFile.UserFile (Hashtable *hash*)

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

Parameters

hash	Hash.

22.21.2 Member Function Documentation

22.21.2.1 virtual void Combu.UserFile.Delete (Action< bool, string > *callback*) [virtual]

Deletes this UserFile from server.

Parameters

callback	Callback.

22.21.2.2 static void Combu.UserFile.Delete (long *idFile,* Action< bool, string > *callback*) [static]

Deletes the specified File from server.

Parameters

idFile	Identifier file.
callback	Callback.

22.21.2.3 void Combu.UserFile.Download (Action< byte[ ]> *callback*)

Download the bytes from the url.

Parameters

callback	Callback.

22.21.2.4 virtual void Combu.UserFile.FromHashtable (Hashtable *hash*) [virtual]

Initialize the object from a hashtable.

Parameters

hash	Hash.

22.21.2.5 virtual void Combu.UserFile.FromJson (string *jsonString*) [virtual]

Initialize the object from a JSON formatted string.

Parameters

jsonString	Json string.

22.21.2.6 virtual void Combu.UserFile.Like (Action< bool, string > *callback*) [virtual]

Increase the Like count of this UserFile.

Parameters

callback	Callback.

22.21.2.7 static void Combu.UserFile.Like (long *idFile,* Action< bool, string > *callback*) [static]

Increase the Like count of a UserFile.

Parameters

idFile	Identifier file.
callback	Callback.

22.21.2.8 static void Combu.UserFile.Like (UserFile *file,* long *idFile,* Action< bool, string > *callback*) [static], [protected]

Increase the Like count of a UserFile.

Parameters

file	File.
idFile	Identifier file (if File is null).
callback	Callback.

22.21.2.9 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

userId	User identifier.
includeShared	If set to `true` include shared.
pageNumber	Page number.
countPerPage	Count per page.
callback	Callback.

22.21.2.10 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

userId	User identifier.
includeShared	If set to `true` include shared.
pageNumber	Page number.
countPerPage	Count per page.
callback	Callback.

Template Parameters

T	Type for UserFile.

Type Constraints

T* : *UserFile

T* : *new()

22.21.2.11 virtual void Combu.UserFile.Update (byte[ ] *contents,* Action< bool, string > *callback*) [virtual]

Update this file to server.

Parameters

contents	Contents of the file to send.
callback	Callback.

22.21.2.12 virtual void Combu.UserFile.View (Action< bool, string > *callback*) [virtual]

Increase the View count of this UserFile.

Parameters

callback	Callback.

22.21.2.13 static void Combu.UserFile.View (long *idFile,* Action< bool, string > *callback*) [static]

Increase the View count of a UserFile.

Parameters

idFile	Identifier file.
callback	Callback.

22.21.2.14 static void Combu.UserFile.View (UserFile *file,* long *idFile,* Action< bool, string > *callback*) [static], [protected]

Increase the View count of a UserFile.

Parameters

file	File.
idFile	Identifier file (if File is null).
callback	Callback.

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/UserFile.cs

22.22 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.

22.22.1 Constructor & Destructor Documentation

22.22.1.1 Combu.UserGroup.UserGroup ( )

Initializes a new instance of the UserGroup class.

22.22.1.2 Combu.UserGroup.UserGroup (string *jsonString*)

Initializes a new instance of the UserGroup class.

Parameters

jsonString	Json string.

22.22.1.3 Combu.UserGroup.UserGroup (Hashtable *hash*)

Initializes a new instance of the UserGroup class.

Parameters

hash	Hash.

22.22.2 Member Function Documentation

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

Delete this instance.

22.22.2.2 virtual void Combu.UserGroup.FromHashtable (Hashtable *hash*) [virtual]

Initialize the object from a hashtable.

Parameters

hash	Hash.

22.22.2.3 virtual void Combu.UserGroup.FromJson (string *jsonString*) [virtual]

Initialize the object from a JSON formatted string.

Parameters

jsonString	Json string.

22.22.2.4 virtual void Combu.UserGroup.Join (System.Action< bool, string > *callback*) [virtual]

Join the local user to this group.

Parameters

callback	Callback.

22.22.2.5 virtual void Combu.UserGroup.Join (long[ ] *idUsers,* System.Action< bool, string > *callback*) [virtual]

Join the specified idUsers to this group.

Parameters

idUsers	Identifier users.
callback	Callback.

22.22.2.6 virtual void Combu.UserGroup.Join (string[ ] *usernames,* System.Action< bool, string > *callback*) [virtual]

Join the specified usernames to this group.

Parameters

usernames	Usernames.
callback	Callback.

22.22.2.7 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

idUsers	Identifier users.
usernames	Usernames.
callback	Callback.

Template Parameters

T	Type for UserGroup records.

Type Constraints

T* : *UserGroup

T* : *new()

22.22.2.8 virtual void Combu.UserGroup.Leave (System.Action< bool, string > *callback*) [virtual]

Leave the local user from this group.

Parameters

callback	Callback.

22.22.2.9 virtual void Combu.UserGroup.Leave (long[ ] *idUsers,* System.Action< bool, string > *callback*) [virtual]

Leave the specified idUsers from this group.

Parameters

idUsers	Identifier users.
callback	Callback.

22.22.2.10 virtual void Combu.UserGroup.Leave (string[ ] *usernames,* System.Action< bool, string > *callback*) [virtual]

Leave the specified usernames from this group.

Parameters

usernames	Usernames.
callback	Callback.

22.22.2.11 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

idUsers	Identifier users.
usernames	Usernames.
callback	Callback.

Template Parameters

T	Type for UserGroup records.

Type Constraints

T* : *UserGroup

T* : *new()

22.22.2.12 virtual void Combu.UserGroup.Save (System.Action< bool, string > *callback*) [virtual]

Save this instance.

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

- /Users/ziored/Documents/Projects/Unity/Empty Test/Assets/Combu/Scripts/UserGroup.cs

Index