GameSpy protocol research and documentation

Arves100, Xiaojiuwo

First Edition

January 2, 2020

Contents

Chapter 1

General Construction

Inside the GameSpy SDK there are 16 compotents, which constructed the GameSpy services.

1.0.1 SDK Module

- Brigades
- Chat
- Presence & Messaging
- CDKey
- Stats & Tracking
- Persistent Storage
- Transport
- NAT Negotation
- Peer to Peer communication
- Patching & Tracking
- Server Browser
- Query & Reporting
- SAKE Persistent Storage
- ATLAS Competition
- Voice Chat
- Web Authentication

1.0.2 Servers

Those compoents allows accessing multiple GameSpy servers that they provide.

- Presence Connection Manager
- Presence Search Player
- Master (Backend)
- Chat
- SAKE Storage
- NAT Negotation Mathup
- Server Browser
- CDKey
- Stats & Tracking
- Web Services

Chapter 2

Presence Servers

Presence & Messaging system allows a game to add account authentication or registration, which includes a profile where personal information could be stored (such as email, first name) and a friend list (called buddies). The presence server also allows to set a location string, which could be used to indicate where the player is currently playing at (for example: GameSpy Arcade uses home://).

2.0.1 Account representation

GameSpy used to store accounts that are associated for a game in a specific way. In the presence server: User, Profile, Unique nickname and Namespace ID are dinstincted.

2.0.1.1 User

A Presence account rapresents an User (the term used by the SDK), an user contains the EMail and Password for such user. The Email address MUST BE unique, as it's used to identify an user in every context (for example: multiple games).

2.0.1.2 Profile

A profile contains the information of an user regarding a game (or the friend list for such game). It could also store information such as birth date and surname. An user could have multiple profiles, and a profile must be associated with only one user.

2.0.1.3 Namespace & Unique Nicknames

A namespace identify a space where a game is defined (for example: namespace 0 identify GameSpy Arcade). In this context, an unique nickname could be associated to the user, such nickname MUST BE unique in that namespace. An profile could have multiple unique nicknames for each namespace, but it could only have one unique nickname in such namespace.

2.0.2 Login methods

There are three methods that a client can login to the GPCM.

- Profile
- Unique nickname
- PreAuthenticated

2.0.2.1 Profile

The profile login is the standard way to login to GPCM, it requires that the client sends the email, the profile nickname and the namespace identification number, which will be used to identify both the profile and the user. The server returns the profile identification number, the user identification number, and the unique nickname associated to that profile in the specific namespace context.

2.0.2.2 Unique nickname

The unique nickname login is used for loggin in with a special nickname in a special game context. Both the unique nickname and the namespace are used to identify the profile and the user. The server returns the profile identification number, the user identification number and the profile nickname associated to that unique nickname in the specific namespace context.

2.0.2.3 PreAuthenticated

The Pre-Authenticated method (also known as Authentication token method) is a special method of logging into GameSpy services that works in conjunction of the Web Services. Using this method, you don't login with a password, nickname or email. You login by using an authentication token and a partner challenge. This two could be retrived from the Web Service Server. The user that is logging in with this method is usually not an ordinary account, it's a "shadow" account, created just for the following backend login (and possibly game). This method is used by special third-party backends that saves the login information in a different way. An example of this is shown with PlayStation 3 login (NP), the Web Service would generate an auth token and partner challenge from the SEN/PSN User information. The client needs to send the authentication token and the partner challenge disguized as an MD5 password. The server returns the profile identification number, the user identification number, the profile nickname and the unique nickname associated to the specific namespace context. Keep in mind that this are permanent account associated with the backend, in case of PlayStation 3 service, you can also see your SEN/PSN friends in your GameSpy buddies list if they had created a shadow account for use in GameSpy services.

2.0.3 Login system

Passwords aren't sent in login requests, rather, both client and server generate a MD5 hash known as "proof", which has in the intent to verify if an user could authenticate to the server or not. The proof is not only used to verify the user

password, but also to make sure that you're not connecting to a fake server, or that the password has not been tampred. It's the job of both client and server to verify the proof once it has been sent by the client or the server. (In the server response, it's known as response) If the proof is not verified, the server sends an error message and close the connection, the client directly cuts the connection with the server.

2.0.3.1 The login proof

The login proof is a 32 hexadecimal string which is used for different purposed,the method of generating it it's different, but yet similar, for both client and server.

Variable user information: Inside the proof, a variable user information (referenced as "user" in the SDK) will be generated. This user information changes for each of the three login method used. If you're logging in with the Profile method, the user information will contain "profile nickname"@"email". If you're logging in with the Unique nickname method, the user information will contain just the unique nickname. If you're logging in with the Pre-Authenticated method, the user information will contain the authentication token. If the Partner ID specified is different than 0 (GameSpy), it will append to the beginning of the user string the following data: "partner id number@". This is not applied to the PreAuthenticated method.

Client generation: First, an MD5 hash of the password is calculated (in case of the Pre-Authenticated method, it's the Partner challenge), then 48 white spaces are appended to the md5 hash password. Then, the variable user information, client challenge and server challenge are appended to the previous string. This long generated string is hashed with MD5 and expressed in hexadecimal.

Server generation: The server generation is almost the same as the client one, the only difference is that the server challenge is appended BEFORE the client challenge, and not after.

2.0.3.2 Login request and response

The login starts by the server, after a client has been accepted, the server sends a "lc" (login challenge) command with the value 1, containing it's id and the server challenge. This challenge will be used for generating the proof.

The client answers by sending a "login" command containing the data required to login with it's login method and the client challenge and the client generated proof, that the server will verify. The server then, calculated it's proof and verify the client one, in case there is a mismatch, the server disconnects the client after sending an error (See /refGPCMErrors for a list of the known errors). If the verification of the client proof has succeeded, the server sends a "lc" command with the value 2, containing the profile's unique nickname, nickname, the server proof and the session key. The clients now, needs to verify if the server proof matches the generated one, in case there is a mismatch, the client disconnects from the server.

2.0.4 Protocol information

The data in both the presence servers (Connection Manager and Search Player) are exchanged with ASCII strings that contains request & response of the client and the server. This ASCII data is known as "Command".

2.0.4.1 Command Pattern

This type of string is represent map containing both a key and value. A command starts with the character "\" and it ends with the following data "$\backslash final\backslash$". As an example, we provide two example strings, one from the client **??** and one from the server **??**

$\backslash login\backslash\backslash challenge\backslash jfeuihj48t4h3w8wj8f34j8\backslash user$
$\backslash spyguy@spyguy@gamespy.com\backslash response$
$\backslash fje4ifh482teifh348h7u\backslash port$
$\backslash -9805\backslash productid\backslash 10469\backslash gamename$
$\backslash conflictsopc\backslash namespaceid\backslash 1\backslash id\backslash 1\backslash final\backslash$

$\backslash lc\backslash 1\backslash challenge\backslash JFU78DCH78\backslash id\backslash 1\backslash final\backslash$

Each \ divide a key to it's value, for example: the first key is "lc" and the first value is "1". The first key is always the command name, for example "lc" is the command name of that server request, while "login" is the command name of the client request. The value after the command name could be used to decribe what action is performing at the moment, in the case of $\backslash lc\backslash 1\backslash$ it is perfoming the action 1 of the command "lc". Client usually doesn't send a value after the command name, leaving two \ after the command name.

2.0.4.2 Errors

The error command is the same for both the two presence server. The table **??** describes the keys that the error string could have.

Table 2.1: GP Error

Name	Description
error	This is the command name, it does not require any extra parameter
err	The ID of the error
errmsg	A string that describes the error

2.0.5 Connection Manager Server

Connection Manager Server (also known as GPCM), is a server that manage the user authentication and profile information. The IP, port and protocol of the server are shown in the table **??**.

IP	Port	Protocol
gpcm.gamespy.com	29900	TCP only

Table 2.2: IP Port and protcol of GPCM