Research On GameSpy Protocol

Arves100, Xiaojiuwo

First Edition

January 2, 2020

Contents

1 General Construction **2**

1.1 SDK Module 2

1.2 Servers 3

1.3 Basic Description of Protocol 3

1.3.1 The String Pattern 3

2 Details of GameSpy Presence **4**

2.1 Presence Connection Manager 4

2.1.1 Server IP and Ports 4

2.1.2 Request For GameSpy Presence Connection Manager . . 4

2.1.2.1 Login Phase 5

Client Login Request 5

Login Response From Server 6

2.1.2.2 User Creation 7

2.2 GameSpy Presence Search Player 8

2.2.1 Search User 8

2.3 GameSpy Status and Tracking 8

Chapter 1

General Construction

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

1.1 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

1.2 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

1.3 Basic Description of Protocol

In this part, we describe some of the basic patterns that are used in all GameSpy servers.

1.3.1 The String Pattern

We first introduce the pattern of the string, which is using to make up a request and response. The following servers do use the pattern: Presence Connection Manager, Presence Search Player, Game Status, CD-Key, Query Report Version 1. This kind of string represents a value in a request and response sent by the client or the server as Table 1.1.

String	**Description**
$\backslash\langle content\rangle\backslash$	The value is $\langle content\rangle$

Table 1.1: Value string

This kind of string is represent a command in a request sends by the client or the server as Table 1.2. The command will end with \\ or \ depends on whether run at the server-side or client-side.

String	**Description**
$\backslash command\backslash\backslash$	This is a command
$\backslash error\backslash\backslash$	Error command
$\backslash lc\backslash$	Login command

Table 1.2: Command string

Chapter 2

Details of GameSpy Presence

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), a friend list (called buddies), private messages.

GameSpy Presence contains two server, GameSpy Presence Connection Manager (GPCM) and GameSpy Presence Search Player (GPSP). GPCM is a server that manages the profiles (such as login, storing the profile information).

2.0.1 Server IP and Ports

Table 2.1 are the GPCM and GPSP IP and Ports that client/game connect to.

IP	Port
gpcm.gamespy.com	29900
gpsp.gamespy.com	29901

Table 2.1: IP and Ports for GameSpy Presence Servers

2.0.2 Request For GameSpy Presence Connection Manager

Table 2.2 lists the request(already known by us) that clients send to GameSpy Presence Connection Manager server (GPSP).

Commands	Description
$\backslash inviteto\backslash\backslash$	Invite friends
$\backslash login\backslash\backslash$	Login to GPCM
$\backslash getprofile\backslash\backslash$	Get the profile of a player (including your own)
$\backslash addbuddy\backslash\backslash$	Add a player to my friend list
$\backslash delbuddy\backslash\backslash$	Delete a player from my friend list
$\backslash updateui\backslash\backslash$	Update login information (email, password)
$\backslash updatepro\backslash\backslash$	Update my profile such as first name, last name, gender etc.
$\backslash logout\backslash\backslash$	Logout manually by user
$\backslash status\backslash\backslash$	Update the status of a user (Such as what game is the player playing)
$\backslash ka\backslash\backslash$	Keep client alive (do not disconnect)

Table 2.2: Request For GameSpy Presence Connection Manager

Error response string for (GPCM, GPSP):

$$\backslash error\backslash\backslash err\backslash\langle errorcode\rangle\backslash fatal\backslash\backslash errmsg\backslash\langle errormessage\rangle\backslash id\backslash 1\backslash final\backslash \tag{2.1}$$

2.0.2.1 Login Phase

Client Login Request
There are three ways of login:

- AuthToken: Logging using an alphanumeric string that rapresents an user
- UniqueNick: Logging using a nickname that is unique from all the players
- User: Logging with the nickname and the password

The full login request string:

$$\begin{gathered}\backslash login\backslash challenge\backslash\langle challenge\rangle\backslash authtoken\backslash\langle authtoken\rangle\\ \backslash uniquenick\backslash\langle uniquenick\rangle\backslash user\backslash\langle user\rangle\backslash userid\backslash\langle userid\rangle\\ \backslash profileid\backslash\langle profileid\rangle\backslash partnerid\backslash\langle partnerid\rangle\backslash response\backslash\langle response\rangle\\ \backslash firewall\backslash 1\backslash port\backslash\langle port\rangle\backslash productid\backslash\langle productid\rangle\\ \backslash gamename\backslash\langle gamename\rangle\backslash namespaceid\backslash\langle namespaceid\rangle\\ \backslash sdkrevision\backslash\langle sdkrevision\rangle\backslash quiet\backslash\langle quiet\rangle\backslash id\backslash 1\backslash final\backslash\end{gathered} \tag{2.2}$$

The value $\langle challenge\rangle$ for $\backslash challenge\backslash$ in 2.2 is a 10 byte alphanumeric string.
The following Table 2.3 is a description of string used in login request, GameSpy can use these string to find value in database.

Keys	**Description**	**Type**
login	The login command which use to identify the login request of client	
challenge	The user challenge used to verify the authenticity of the client	
authtoken	The token used to login (represent of an user)	
uniquenick	The unique nickname used to login	
user	The users account (format is NICKNAME@EMAIL)	
userid	Send the userid (for example when you disconnect you will keep this)	
profileid	Send the profileid (for example when you disconnect you will keep this)	
partnerid	This ID is used to identify a backend service logged with gamespy.(Nintendo WIFI Connection will identify his partner as 11, which means that for gamespy, you are logging from a third party connection)	
response	The client challenge used to verify the authenticity of the client	
firewall	If this option is set to 1, then you are connecting under a firewall/limited connection	
port	The peer port (used for p2p stuff)	
productid	An ID that identify the game you're using	
gamename	A string that rapresents the game that you're using, used also for several activities like peerchat server identification	
namespaceid	?	
sdkrevision	The version of the SDK you're using	
quiet	? Maybe indicate invisible login which can not been seen at friends list	
id	The value is 1	
final	Message end	

Table 2.3: Login parameter string

Login Response From Server

This response string 2.3, 2.4 is send by the server when a connection is accepted, and followed by a challenge2.2, which verifies the server that client connect to.

There are two kinds of login response string:

$$\begin{gathered}\backslash lc\backslash 1\backslash challenge\backslash\langle challenge\rangle\backslash nur\\ \backslash userid\backslash\langle userid\rangle\backslash profileid\backslash\langle profileid\rangle\backslash final\backslash\end{gathered}\tag{2.3}$$

Keys	Description	Type
challenge	The challenge string sended by GameSpy Presence server	
nur	?	
userid	The userID of the profile	
profileid	The profileID	
final		

Table 2.4: The first type login response

$$\backslash lc\backslash 2\backslash sesskey\backslash\langle sesskey\rangle\backslash userid\backslash\langle userid\rangle\backslash profileid\backslash\langle profileid\rangle \\ \backslash uniquenick\backslash\langle uniquenick\rangle\backslash lt\backslash\langle lt\rangle\backslash proof\backslash\langle proof\rangle\backslash final\backslash \tag{2.4}$$

Keys	Description	Type
sesskey	The session key, which is a integer rapresentating the client connection	
userid	The userID of the profile	
profileid	The profileID	
uniquenick	The logged in unique nick	
lt	The login ticket, unknown usage	
proof	The proof is something similar to the response but it vary	
final		

Table 2.5: The second type login response

Proof in 2.5 generation: $md5(password)||48spaces$ The user could be AuthToken or the User/UniqueNick (with the extra PartnerID). server challenge that we received before. the client challenge that was generated before.

2.0.2.2 User Creation

This commmand 2.5 is used to create a user in GameSpy.

$$\backslash newuser\backslash email\backslash\langle email\rangle\backslash nick\backslash\langle nick\rangle \\ \backslash passwordenc\backslash\langle passwordenc\rangle\backslash productid\backslash\langle productid\rangle \\ \backslash gamename\backslash\langle gamename\rangle\backslash uniquenick\backslash\langle uniquenick\rangle \\ \backslash cdkeyenc\backslash\langle cdkeyenc\rangle\backslash partnerid\backslash\langle partnerid\rangle\backslash id\backslash 1\backslash final\backslash \tag{2.5}$$

The description of each parameter string is shown in Table 2.6.

String	Description	Type
email	The email used to create	
nick	The nickname that will be created	
passwordenc	The encoded password (password XOR with Gamespy seed and the Base64 encoded)	
productid	An ID that identify the game you're using	
gamename	A string that rapresents the game that you're using, used also for several	
namespaceid	?Unknown	
uniquenick	Uniquenick that will be created	
cdkeyenc	The encrypted CDkey, encrypted method is the same as the passwordenc	
partnerid	This ID is used to identify a backend service logged with gamespy	
id	The value of id is 1	
final	Message end	

Table 2.6: User creation string

2.1 GameSpy Presence Search Player

Table **??** are the GPSP IP and Ports that client/game connect to.

IP	Port
gpsp.gamespy.com	29901

Table 2.7: IP and Ports for GameSpy Presence Search Player

2.1.1 Search User

This is the request that client sends to server:

$$\begin{aligned}\backslash search\backslash\backslash sesskey\backslash < sesskey > \backslash profileid\backslash < profileid >\\ \backslash namespaceid\backslash < namespaceid > \backslash partnerid\backslash < partnerid >\\ \backslash nick\backslash < nick > \backslash uniquenick\backslash < uniquenick >\\ \backslash email\backslash < email > \backslash gamename\backslash < gamename > \backslash final\backslash\end{aligned} \tag{2.6}$$

This is the response that server sends to client:

$$\begin{aligned}\backslash bsr\backslash < profileid > \backslash nick\backslash < nick > \backslash uniquenick\backslash < uniquenick >\\ \backslash namespaceid\backslash < namespaceid > \backslash firstname\backslash < firstname >\\ \backslash lastname\backslash < lastname > \backslash email\backslash < email >\\ \backslash bsrdone\backslash < gamespyencdeterminator > \backslash final\backslash\end{aligned} \tag{2.7}$$

2.2 GameSpy Status and Tracking

when game connect to GSTATS server, server will send an message to game which contains the challenge, the total length of message must bigger than

38bytes, and the challenge must bigger than 20bytes. when game received the challenge it will compute a response, the response is formed as follows. response = CRC32(<server challenge>,<length of server challenge>)||<game secret key> then game will compute the MD5 hash as MD5value = MD5(<response>,<length of response>) then encoded with Enctype3 then construct the challenge-response message as $\backslash auth\backslash\backslash gamename\backslash < gamename > \backslash response\backslash < MD5value > \backslash port\backslash < port > \backslash id\backslash < id >$

session key length (unknown) connction id = transfer ascii of sessionkey to integer

the initialization phase is finished. server challenge message length (bigger than 38-byte) server challenge length (bigger than 20-byte) $\backslash final\backslash$ is encrypted using XOR Enctype1 at the end of the challenge that sends by the server.