Sun Game Server Matchmaking Client API Programmer's Guide

Table of Contents

Introduction

This document describes the client API for Sun Game Server (SGS) matchmaking services.

The API in this document is defined using the syntax of the Java language; the C++ API is not currently available. For more information on SGS client applications, see the programmer's guide for the J2SE, J2ME, or C++ Client API.

Overview

The Matchmaking Server is an SGS application that tracks the management of *lobbies* and *game rooms*.

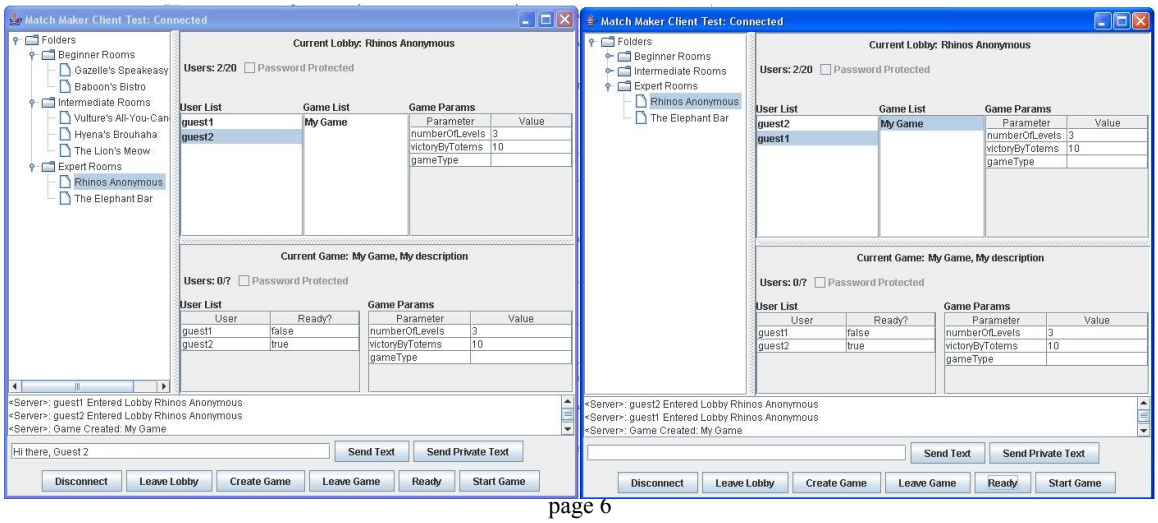
A *lobby* is a common area to which users can connect. In lobbies, users can chat (publicly or privately) and join or create game rooms. In addition, a lobby holds the default game parameters for the type of games that can be created from it.

In a *game room*, users can also chat while they wait for a game to start. All users must indicate that they are ready in order for a game to start. Once the game is started, the game room is deleted (it has served its purpose as a launching point for a game), and the users are disconnected from the lobby. Both lobbies and game rooms can be password-protected, and lobbies carry a user limit (game rooms currently don't have a limit, but may in future releases of the software).

The server manages all these interactions via well-known command protocols (for example, JOIN_LOBBY (0x45)). Each version of the client must communicate with the server via these well-known commands and provide an interface to the actual game client.

A typical game might use a client API to present its player with a list of lobbies from which to choose via a user interface. The player would pick a lobby suitable to his or her needs (lobbies may be organized by skill, for example). Once in the lobby the player could join a game about to start, or create one of his or her own. Once the game starts, the Matchmaker has done its job, and the game itself takes over.

Below is an example of a user interface. It shows two users, **guest1** and **guest2**, both connected to the same lobby and the same game room within that lobby.



Matchmaking Client Interfaces

IMatchMakingClient/MatchMakingClient

IMatchMaking and its implementation MatchMakingClient are the starting point for all interaction with the Matchmaking Server. It is a wrapper around the basic SGS API.

```
interface IMatchmakingClient
{
void setListener(IMatchmakingClientListener listener)

void listFolder(String folderPath);

void lookupUserID(String userName);
void lookupUserName(byte[] userID);
void locateUserInLobby(byte[] userID);

void joinLobby(byte[] lobbyID);
void joinLobby(byte[] lobbyID, String password);

void joinGame(byte[] gameID);
void joinGame(byte[] gameID, String password);
}
```

MatchmakingClient Constructor

This is the constructor for an implementation of IMatchMakingClient. It requires a reference to a ClientConnectionManager to function correctly.

```
MatchmakingClient(ClientConnectionManager manager)
```

Parameters

manager

This provides access to the SGS API.

listFolder

This command is used by the client to request a list of folders and lobbies contained at a particular level in the lobby hierarchy. The response to this method is received through *ImatchmakingClientListener.listedFolder*.

```
void listFolder(byte[] folderID)
```

Parameters

folderID

This specifies the folder ID on which the client wants information. To retrieve the root of the hierarchy, pass in null. Otherwise, pass an ID that was taken from the *getFolderID* member of a *FolderDescriptor*.

Return Value

None

lookupUserID

This command is used by the client to request if it needs to convert a user name to a User ID. The response to this request is obtained via *IMatchMakingClientListener.foundUserID*.

```
void lookupUserID(String userName)
```

Parameters

userName

The user name whose User ID we want to find.

Return Value

None

lookupUserName

This command is used by the client to request if it needs to convert a user ID to a User name. The response to this request is obtained via *IMatchMakingClientListener.foundUserName.*

```
void lookupUserName(byte[] userID)
```

Parameters

userID

The user ID whose name we want to find.

Return Value

None

locateUserInLobby

This command is used by the client to find out which lobby or lobbies a particular user is in.

```
void locateUserInLobby(byte[] userID)
```

Parameters
userID
The User ID we want to find.

Return Value
None

joinLobby

The client issues this command to join a lobby that is not password-protected

```
void joinLobby(byte[] lobbyID)
```

Parameters
lobbyID
The identifier of the lobby the user wants to join

Return Value
None.

joinLobby

The client issues this command to join a lobby that is password-protected

```
void joinLobby(byte[] lobbyID, String password)
```

Parameters
lobbyID
The identifier of the lobby the user wants to join
password
The password required to join the game

Return Value
None.

joinGame

The client issues this command to join a game that is not password-protected

```
void joinGame(byte[] gameID)
```

Parameters
gameID
The identifier of the game the user wants to join

Return Value
None.

joinGame

The client issues this command to join a game that is password-protected

```
void joinGame(byte[] gameID, String password)
```

Parameters

gameID

The identifier of the game the user wants to join

password

The password required to join the game

Return Value

None.

setListener

This command is used to attach a listener to the IMatchmakingClient

```
void setListener(IMatchmakingClientListener listener)
```

Parameters

listener

Pointer to the object that will now receive event notifications. May be null to specify no listener.

Return Value

None.

leaveLobby

This command attempts to leave the currently connected lobby. The client should not assume it has actually left the lobby until the ImatchMakingClientListener.leftLobby call back is received.

```
void leaveLobby()
```

Parameters

None.

Return Value

None.

leaveGame

This command attempts to leave the currently connected game room. The client should not assume it has actually left the game room until the ImatchMakingClientListener.leftGame call back is received.

```
void leaveGame()
```

Parameters

None.

Return Value

None.

completeGame

This command will notify the server that the game given by the game ID has ended and that the corresponding game channel should be closed. Only the host may call this.

```
void completeGame(byte[] gameID)
```

Parameters

gameID

The unique identifier for the game

Return Value

None.

IMatchmakingClientListener

IMatchmakingClientListener provides notifications of events from the IMatchmakingClient.

```
interface IMatchmakingClientListener
{
void listedFolder(byte[] folderID, FolderDescriptor[] subfolders,
LobbyDescriptor[] lobbies)

void foundUserID(String userName, byte[] userID)
void foundUserName(String userName, byte[] userID)
void locatedUserInLobby(LobbyDescriptor[] lobbies)

void joinedLobby(ILobbyChannel lobbyChannel)
void joinedGame(IGameChannel gameChannel)
void leftGame()
void leftLobby()
void connected(byte[] myID)
void disconnected()
void validationRequest(Callback[] callbacks)
void error(String message)
}
```

listedFolder

This method is the response to a *IMatchmakingClient::listFolder* request.

```
void listedFolder(byte folderID, FolderDescriptor[] subfolders, LobbyDescriptor[]
lobbies)
```

Parameters

folderID

The ID of the folder that had its contents listed.

subfolders

An array containing all the subfolders of this folder.

lobbies

An array containing all the lobbies in this folder.

Return Value

None

foundUserID

This method is the response to a *ImatchmakingClient.lookupUserID* request.

```
void foundUserID(String userName, byte[] userID)
```

Parameters

userName

The user name that was specified in the original request

userID

This specifies the current User UUID for the specified user name. If this parameter is a zero-length array then the requested user is not currently connected.

Return Value

None

foundUserName

This method is the response to a *ImatchmakingClient.lookupUserName* request.

```
void foundUserName(String userName, byte[] userID)
```

Parameters

userName

The user name for the specified User ID.

userID

The User ID that was specified in the original request

Return Value

None

locatedUserInLobby

This method is the response to a *ImatchmakingClient.locateUserInLobby* request.

```
void locatedUserInLobby(LobbyDescriptor[] lobbies)
```

Parameters

lobbies

An array containing all the lobbies the user is currently in. This may be an empty list.

Return Value

None

joinedLobby

This method is sent in response to a request to join a lobby

```
void joinedLobby(ILobbyChannel lobbyChannel)
```

Parameters

lobbyChannel

The interface used to talk to the lobby channel

Return Value

None.

joinedGame

Description

```
void joinedGame(IGameChannel gameChannel)
```

Parameters

gameChannel

The interface used to talk to the game channel

Return Value

None.

leftLobby

This call back is called as confirmation that the user has left whichever lobby they were previously connected to.

```
void leftLobby()
```

Parameters

None.

Return Value

None.

leftGame

This call back is called as confirmation that the user has successfully left whichever game they were previously connected to.

```
void leftGame()
```

Parameters

None.

Return Value

None.

connected

This call back is called when the client has successfully connected to the server and received confirmation that is has been successfully joined to the Lobby Manager Control channel. After receiving this call back the client is free to start sending commands to the server.

```
void connected(byte[] myID)
```

Parameters

None.

Return Value

None.

disconnected

This call back is called as confirmation that the user has been disconnected from the server. The disconnect may or may not have been initiated by the client.

```
void disconnected()
```

Parameters

None

Return Value

None.

validationRequest

This call back is a request for user credentials, such as user name and password, from the server. The client should populate the call backs as appropriate.

```
void validationRequest(Callback[] callbacks)
```

Parameters

callbacks

The security call backs used for client authentication.

Return Value

None.

error

This call back is called when a command results in an error on the server.

```
void error(int code)
```

Parameters

code

The error code

Return Value

None.

FolderDescriptor

FolderDescriptor is a new class used as a container for information describing a particular folder in the folder hierarchy.

getName

This accesses the user-friendly name of the folder. This is the string that should be displayed to the user when representing folders in a user interface.

```
String getName()
```

Parameters

None

Return Value

The user-friendly name of the folder.

getDescription

This accesses a user-friendly long description of the folder. This is the string that should be displayed to the user when representing the full details of a folder in the user interface.

```
String getDescription()
```

Parameters

None

Return Value

The user-friendly description of the folder.

getFolderID

This accesses the full path to the folder. This string should be treated as an opaque value and only used in calls to *UserManagerClient.listFolder*.

```
Byte[] getFolderID()
```

Parameters

None

Return Value

The unique identifier of the folder.

LobbyDescriptor

LobbyDescriptor is a new class used as a container for information describing a particular lobby in the folder hierarchy.

getName

This accesses the user-friendly name of the lobby. This is the string that should be displayed to the user when representing lobbies in a user interface.

```
String getName()
```

Parameters

None

Return Value

The user-friendly name of the lobby.

getChannelName

The channel name ties a LobbyDescriptor to an ILobbyChannel. The channel name is unique across the lobby system.

```
String getChannelName()
```

Parameters

None.

Return Value

None.

getDescription

This accesses a user-friendly long description of the lobby. This is the string that should be displayed to the user when representing the full details of a lobby in the user interface.

```
String getDescription()
```

Parameters

None

Return Value

The user-friendly description of the lobby.

getNumUsers

This enables the client to find out how many users are currently in the lobby.

```
int getNumUsers()
```

Parameters

None

Return Value

The number of users currently in the lobby.

getMaxUsers

This enables the client to find out the maximum number of users allowed in the lobby.

```
int getMaxUsers()
```

Parameters

None

Return Value

The maximum number of users currently in the lobby.

isPasswordProtected

This enables the client to find out if the lobby is protected by a password.

```
boolean IsPasswordProtected ()
```

Parameters

None

Return Value

true if the lobby is password protected, otherwise false.

getLobbyID

This enables the client to uniquely identify a lobby in the lobby system.

```
byte[] getLobbyID()
```

Parameters

None

Return Value

The unique identifier of this lobby.

ILobbyChannel

The ILobbyChannel interface is used to execute all communication with a particular lobby. It has a matching ILobbyChannelListener interface that provides asynchronous event notifications.

```
interface ILobbyChannel
{
void setListener(ILobbyChanneListener listener)

void sendText(String text)
void sendPrivateText(byte[] to, String text)

void requestGameParameters()

void createGame(String name, string description, string password,
HashMap<String, Object> parameters)

}
```

setListener

Description

```
void setListener(ILobbyChanneListener listener)
```

Parameters

listener

The listener that will receive events on this lobby channel.

Return Value

None.

sendText

Description

```
void sendText(String text)
```

Parameters

text

The message to send.

Return Value

None.

sendPrivateText

Description

```
void sendPrivateText(byte[] to, String text)
```

Parameters

to

The recipient of the message

text

The message

Return Value

None.

requestGameParameters

Description

```
void requestGameParameters()
```

Parameters

None.

Return Value

None.

createGame

Description

```
void CreateGame(String name, string description, string password,
HashMap<String, Object> parameters)
```

Parameters

name

The proposed game name

description

The proposed game description

password

An option password to connect to the game

parameters

A map of configurable game parameters

Return Value

None.

ILobbyChannelListener

This is the matching class to ILobbyChannel that provides for event notifications.

```
interface ILobbyChannelListener
{
void playerEntered(byte[] userID, String userName)
void playerLeft(byte[] userID)

void receiveText(byte[] from, String text, boolean wasPrivate)

void receivedGameParameters(HashMap<String, Object> parameters)

void createGameFailed(String name, String reason)

void gameCreated(GameDescriptor game)
void gameUpdated(GameDescriptor game)
void gameDeleted(GameDescriptor game)
void gameStarted(GameDescriptor game)

void playerJoinedGame (GameDescriptor game, byte[] user)
void playerLeftGame (GameDescriptor game, byte[] user)
void playerBootedFromGame (GameDescriptor game, byte[] booter, byte[]
bootee, boolean isBanned, String reason)
}
```

playerEntered

Description

```
void PlayerEntered(byte[] userID, String userName)
```

Parameters

Parameter 1

Parameter 1 Description

Return Value

None.

playerLeft

Description

```
void playerLeft(byte[] userID)
```

Parameters

Parameter 1

Parameter 1 Description

Return Value

None.

Description

receiveText

Description

```
void receiveText(byte[] from, String text, boolean wasPrivate)
```

Parameters

Parameter 1

Parameter 1 Description

Return Value

None.

receivedGameParameters

Description

```
void receivedGameParameters(HashMap<String, Object> parameters)
```

Parameters

Parameter 1

Parameter 1 Description

Return Value

None.

createGameFailed

Description

```
void createGameFailed(String name, int errorCode)
```

Parameters

name

The proposed game name

errorCode

A numeric code describing the error

Return Value

None.

gameCreated

Description

```
void gameCreated(GameDescriptor game)
```

Parameters
Parameter 1
Parameter 1 Description

Return Value
None.

gameUpdated (Not yet implemented)

Description

```
void gameUpdated(GameDescriptor game)
```

Parameters
Parameter 1
Parameter 1 Description

Return Value
None.

gameDeleted

Description

```
void gameDeleted(GameDescriptor game)
```

Parameters
Parameter 1
Parameter 1 Description

Return Value
None.

gameStarted

Description

```
void gameStarted(GameDescriptor game)\
```

Parameters
Parameter 1
Parameter 1 Description

Return Value
None.

playerJoinedGame

Description

```
void playerJoinedGame (GameDescriptor game, byte[] user)
```

Parameters

Parameter 1

Parameter 1 Description

Return Value

None.

playerLeftGame

Description

```
void playerLeftGame (GameDescriptor game, byte[] user)
```

Parameters

Parameter 1

Parameter 1 Description

Return Value

None.

playerBootedFromGame (Not yet implemented)

Description

```
void playerBootedFromGame (GameDescriptor game, byte[] booter, byte[]
bootee, boolean isBanned, String reason)
```

Parameters

Parameter 1

Parameter 1 Description

Return Value

None.

GameDescriptor

getName

Description

```
String getName()
```

Parameters
Parameter 1
Parameter 1 Description

Return Value
None.

getChannelName

The channel name ties a GameDescriptor to an IGameChannel. The channel name is unique across the lobby system.

```
String getChannelName()
```

Parameters
None.

Return Value
None.

getDescription

Description

```
String getDescription()
```

Parameters
Parameter 1
Parameter 1 Description

Return Value
None.

isPasswordProtected

Description

```
Boolean IsPasswordProtected()
```

Parameters

Parameter 1

Parameter 1 Description

Return Value

None.

getParameters

This method returns a map of game parameter names to their associated values.

```
HashMap<String, Object> getParameters()
```

Parameters

None.

Return Value

Returns a map of game parameters names to values for this game.

getGameID

The game ID uniquely identifies the associated game in the lobby system.

```
byte[] getGameID()
```

Parameters

None.

Return Value

Returns a unique ID for the game.

IGameChannel

This class provides access to all functionality needed within a particular game room. It has a matching listener class called IgameChannelListener to receive command call backs.

```
interface IGameChannel
{
void setListener(IGameChannelListener listener)

void sendText(String text)
void sendPrivateText(byte[] to, String text)

void updateGame(String name, String description, String password,
HashMap<String, object> parameters)

void boot(byte[] userID, boolean isBanned, String reason)

void ready(GameDescriptor game, boolean isReady)

void startGame()
}
```

setListener

Description

```
void setListener(GameChannelListener listener)
```

Parameters

listener

Return Value

None.

sendText

Description

```
void sendText(String text)
```

Parameters

text

The message to send

Return Value

None.

sendPrivateText

Send a private message to the given user.

```
void sendPrivateText(byte[] to, String text)
```

Parameters

to

the recipient of the message

text

the message

Return Value

None.

updateGame (Not yet implemented)

Description

```
void updateGame(String name, String description, String password,
HashMap<String, object> parameters)
```

Parameters

Parameter 1

Parameter 1 Description

Return Value

None.

boot (Not yet implemented)

Description

```
void Boot(byte[] userID, boolean isBanned, String reason)
```

Parameters

Parameter 1

Parameter 1 Description

Return Value

None.

ready

This command is sent to the server to indicate that the player is ready (or not) to start the game. All players must indicate that they are ready before a game can start.

```
void ready(GameDescriptor game, boolean isReady)
```

Parameters

game

This game descriptor must match the game settings on the server in order for the readiness state to be accepted.

isReady

If true, the player is ready to start the game.

Return Value

None.

startGame

The command attempts to start the game to which the player is currently connected. Only the host may start a game, and additionally, all the players must indicate that they are ready.

```
void startGame()
```

Parameters

None.

Return Value

None.

IGameChannelListener

This class is the event notification class for the IGameChannel interface.

```
interface IGameChannelListener
{
void receiveText(byte[] from, String text, boolean wasPrivate)

void playerEntered(byte[] userID, String userName)
void playerLeft(byte[] userID, String userName)

void updateGameFailed(GameDescriptor game, String reason)
void gameUpdated(GameDescriptor game)

void  bootFailed(byte[] userID, boolean isBanned, String reason)
void playerBootedFromGame(byte[] booter, byte[] bootee, boolean isBanned,
String reason)

void playerReady(byte[] userID, boolean isReady)

void startGameFailed(String reason)
void gameStarted(GameDescriptor game)
void gameCompleted();
}
```

receiveText

This call back receives text on the game channel.

```
void receiveText(byte[] from, String text, boolean wasPrivate)
```

Parameters

Parameter 1

Parameter 1 Description

Return Value

None.

playerEntered

Description

```
void playerEntered(byte[] userID, String userName)
```

Parameters

Parameter 1

Parameter 1 Description

Return Value

None.

playerLeft

Description

```
void playerLeft(byte[] userID, String userName)
```

Parameters
Parameter 1
Parameter 1 Description

Return Value
None.

updateGameFailed (Not yet implemented)

Description

```
void updateGameFailed(GameDescriptor game, String reason)
```

Parameters
Parameter 1
Parameter 1 Description

Return Value
None.

gameUpdated (Not yet implemented)

Description

```
void gameUpdated(GameDescriptor game)
```

Parameters
Parameter 1
Parameter 1 Description

Return Value
None.

bootFailed (Not yet implemented)

Description

```
void bootFailed(byte[] userID, boolean isBanned, String reason)
```

Parameters
Parameter 1
Parameter 1 Description

Return Value
None.

playerBootedFromGame (Not yet implemented)

Description

```
void playerBootedFromGame(byte[] booter, byte[] bootee, boolean isBanned,
String reason)
```

Parameters

Parameter 1

Parameter 1 Description

Return Value

None.

playerReady

Description

```
void playerReady(byte[] userID, boolean isReady)
```

Parameters

Parameter 1

Parameter 1 Description

Return Value

None.

startGameFailed

Description

```
void startGameFailed(String reason)
```

Parameters

Parameter 1

Parameter 1 Description

Return Value

None.

gameStarted

Description

```
void gameStarted(GameDescriptor game)
```

Parameters

Parameter 1

Parameter 1 Description

Return Value

None.

gameCompleted

This call back is called in response to the *IMatchMakerClient.completeGame* command from the host. It serves as confirmation that the game channel has closed.

```
void gameCompleted()
```

Parameters

None.

Return Value

None.