

GGZMod Reference Manual

0.0.14

Generated by Doxygen 1.5.1

Fri Nov 30 14:59:06 2007

Contents

1	GGZMod Data Structure Index	1
1.1	GGZMod Data Structures	1
2	GGZMod File Index	3
2.1	GGZMod File List	3
3	GGZMod Data Structure Documentation	5
3.1	GGZSeat Struct Reference	5
3.2	GGZSpectatorSeat Struct Reference	7
4	GGZMod File Documentation	9
4.1	ggzmod.h File Reference	9

Chapter 1

GGZMod Data Structure Index

1.1 GGZMod Data Structures

Here are the data structures with brief descriptions:

GGZSeat (A seat at a GGZ game table)	5
GGZSpectatorSeat (A game spectator entry)	7

Chapter 2

GGZMod File Index

2.1 GGZMod File List

Here is a list of all documented files with brief descriptions:

<code>ggzmod.h</code> (Common functions for interfacing a game client and GGZ)	9
--	---

Chapter 3

GGZMod Data Structure Documentation

3.1 GGZSeat Struct Reference

A seat at a GGZ game table.

```
#include <ggzmod.h>
```

Data Fields

- unsigned int **num**
- GGZSeatType **type**
- const char * **name**

3.1.1 Detailed Description

A seat at a GGZ game table.

Each seat at the table is tracked like this.

3.1.2 Field Documentation

3.1.2.1 unsigned int GGZSeat::num

Seat index; 0..(num_seats-1).

3.1.2.2 GGZSeatType GGZSeat::type

Type of seat.

3.1.2.3 const char* GGZSeat::name

Name of player occupying seat.

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

- `ggzmod.h`

3.2 GGZSpectatorSeat Struct Reference

A game spectator entry.

```
#include <ggzmod.h>
```

Data Fields

- unsigned int **num**
- const char * **name**

3.2.1 Detailed Description

A game spectator entry.

Spectators are kept in their own table. A spectator seat is occupied if it has a name, empty if the name is NULL.

3.2.2 Field Documentation

3.2.2.1 unsigned int GGZSpectatorSeat::num

Spectator seat index

3.2.2.2 const char* GGZSpectatorSeat::name

The spectator's name (NULL => empty)

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

- **ggzmod.h**

Chapter 4

GGZMod File Documentation

4.1 ggzmod.h File Reference

Common functions for interfacing a game client and GGZ.

```
#include <ggz.h>
#include <ggz_common.h>
```

Data Structures

- struct **GGZSeat**
A seat at a GGZ game table.
- struct **GGZSpectatorSeat**
A game spectator entry.
- struct **GGZChat**
- struct **GGZPlayerInfo**

Defines

- #define **GGZMOD_VERSION_MAJOR** 0
- #define **GGZMOD_VERSION_MINOR** 0
- #define **GGZMOD_VERSION_MICRO** 14
- #define **GGZMOD_VERSION_IFACE** "5:0:1"

Typedefs

- typedef **GGZMod** GGZMod
A GGZmod object, used for tracking a ggz<->table connection.
- typedef void(*) **GGZModHandler** (GGZMod *mod, GGZModEvent e, const void *data)
Event handler prototype.

Enumerations

- enum **GGZModState** {
GGZMOD_STATE_CREATED, **GGZMOD_STATE_CONNECTED**,
GGZMOD_STATE_WAITING, **GGZMOD_STATE_PLAYING**,
GGZMOD_STATE_DONE }
Table states.
- enum **GGZModEvent** {
GGZMOD_EVENT_STATE, **GGZMOD_EVENT_SERVER**, **GGZMOD_EVENT_PLAYER**, **GGZMOD_EVENT_SEAT**,
GGZMOD_EVENT_SPECTATOR_SEAT, **GGZMOD_EVENT_CHAT**,
GGZMOD_EVENT_STATS, **GGZMOD_EVENT_INFO**,
GGZMOD_EVENT_ERROR }
Callback events.
- enum **GGZModType** { **GGZMOD_GGZ**, **GGZMOD_GAME** }
The "type" of ggzmod.

Functions

- int **ggzmod_is_ggz_mode** (void)
Is the program running in GGZ mode?
- **GGZMod *** **ggzmod_new** (**GGZModType** type)
Create a new ggzmod object.
- void **ggzmod_free** (**GGZMod ***ggzmod)
Destroy a finished ggzmod object.
- int **ggzmod_get_fd** (**GGZMod ***ggzmod)
Get the file descriptor for the GGZMod socket.
- **GGZModType** **ggzmod_get_type** (**GGZMod ***ggzmod)
Get the type of the ggzmod object.
- **GGZModState** **ggzmod_get_state** (**GGZMod ***ggzmod)
Get the current state of the table.
- int **ggzmod_get_server_fd** (**GGZMod ***ggzmod)
Get the fd of the game server connection.
- int **ggzmod_get_num_seats** (**GGZMod ***ggzmod)
Get the total number of seats at the table.
- **GGZSeat** **ggzmod_get_seat** (**GGZMod ***ggzmod, int seat)
Get all data for the specified seat.

- **int ggzmod_get_num_spectator_seats (GGZMod *ggzmod)**
Get the maximum number of spectators. This function returns the maximum number of spectator seats available. A game can use this to iterate over the spectator seats to look for spectators occupying them. Since spectators may come and go at any point and there is no limit on the number of spectators, you should consider this value to be dynamic and call this function again each time you're looking for spectators.
- **GGZSpectatorSeat ggzmod_get_spectator_seat (GGZMod *ggzmod, int seat)**
Get a spectator's data.
- **const char * ggzmod_get_player (GGZMod *ggzmod, int *is_spectator, int *seat_num)**
Get data about this player.
- **void * ggzmod_get_gamedata (GGZMod *ggzmod)**
Return gamedata pointer.
- **void ggzmod_set_gamedata (GGZMod *ggzmod, void *data)**
Set gamedata pointer.
- **void ggzmod_set_handler (GGZMod *ggzmod, GGZModEvent e, GGZModHandler func)**
Set a handler for the given event.
- **int ggzmod_dispatch (GGZMod *ggzmod)**
Check for and handle input.
- **int ggzmod_set_state (GGZMod *ggzmod, GGZModState state)**
Change the table's state.
- **int ggzmod_connect (GGZMod *ggzmod)**
Connect to ggz.
- **int ggzmod_disconnect (GGZMod *ggzmod)**
Disconnect from ggz.
- **void ggzmod_request_stand (GGZMod *ggzmod)**
Stand up (move from your seat into a spectator seat).
- **void ggzmod_request_sit (GGZMod *ggzmod, int seat_num)**
Sit down (move from a spectator seat into a player seat).
- **void ggzmod_request_boot (GGZMod *ggzmod, const char *name)**
Boot a player. Only the game host may do this.
- **void ggzmod_request_bot (GGZMod *ggzmod, int seat_num)**
Change the requested seat from an open seat to a bot.
- **void ggzmod_request_open (GGZMod *ggzmod, int seat_num)**
Change the requested seat from a bot to an open seat.

- void **ggzmod_request_chat** (**GGZMod** *ggzmod, const char *chat_msg)
Chat! This initiates a table chat.
- int **ggzmod_player_get_record** (**GGZMod** *ggzmod, **GGZSeat** *seat, int *wins, int *losses, int *ties, int *forfeits)
Get the player's win-loss record.
- int **ggzmod_player_get_rating** (**GGZMod** *ggzmod, **GGZSeat** *seat, int *rating)
Get the player's rating.
- int **ggzmod_player_get_ranking** (**GGZMod** *ggzmod, **GGZSeat** *seat, int *ranking)
Get the player's ranking.
- int **ggzmod_player_get_highscore** (**GGZMod** *ggzmod, **GGZSeat** *seat, int *highscore)
Get the player's highscore.
- int **ggzmod_spectator_get_record** (**GGZMod** *ggzmod, **GGZSpectatorSeat** *seat, int *wins, int *losses, int *ties, int *forfeits)
Get the spectator's win-loss record.
- int **ggzmod_spectator_get_rating** (**GGZMod** *ggzmod, **GGZSpectatorSeat** *seat, int *rating)
Get the spectator's rating.
- int **ggzmod_spectator_get_ranking** (**GGZMod** *ggzmod, **GGZSpectatorSeat** *seat, int *ranking)
Get the spectator's ranking.
- int **ggzmod_spectator_get_highscore** (**GGZMod** *ggzmod, **GGZSpectatorSeat** *seat, int *highscore)
Get the spectator's highscore.
- int **ggzmod_player_request_info** (**GGZMod** *ggzmod, int seat_num)
Request extended player information for one or more players.
- **GGZPlayerInfo** * **ggzmod_player_get_info** (**GGZMod** *ggzmod, int seat)
Get the extended information for the specified seat.

4.1.1 Detailed Description

Common functions for interfacing a game client and GGZ.

This file contains all libggzmod functions used by game clients to interface with GGZ. Just include **ggzmod.h** (p. 9) and make sure your program is linked with libggzmod. Then use the functions below as appropriate.

GGZmod currently provides an event-driven interface. Data from communication sockets is read in by the library, and a handler function (registered as a callback) is invoked to handle any events.

The calling program should not read/write data from/to the GGZ socket unless it really knows what it is doing.

This does not apply to the game server sockets: ggzmod provides one file descriptor for communicating (TCP) to the game server. If data is ready to be read this file descriptor, ggzmod may invoke the appropriate handler (see below), but will never actually read any data.

For more information, see the documentation at <http://www.ggzgamingzone.org/>.

4.1.2 Typedef Documentation

4.1.2.1 typedef struct GGZMod GGZMod

A GGZmod object, used for tracking a ggz<->table connection.

A game client should track a pointer to a GGZMod object; it contains all the state information for communicating with GGZ. The GGZ client will track one such object for every game table that is running.

4.1.2.2 typedef void(*) GGZModHandler(GGZMod *mod, GGZModEvent e, const void *data)

Event handler prototype.

A function of this type will be called to handle a ggzmod event.

Parameters:

mod The ggzmod state object.

e The event that has occurred.

data Pointer to additional data for the event. The additional data will be of the following form:

- GGZMOD_EVENT_STATE: The old state (GGZModState*)
- GGZMOD_EVENT_SERVER: The fd of the server connection (int*)
- GGZMOD_EVENT_PLAYER: The old player data (int[2])
- GGZMOD_EVENT_SEAT: The old seat (GGZSeat*)
- GGZMOD_EVENT_SPECTATOR_SEAT: The old seat (GGZSpectatorSeat*)
- GGZMOD_EVENT_ERROR: An error string (char*)

4.1.3 Enumeration Type Documentation

4.1.3.1 enum GGZModEvent

Callback events.

Each of these is a possible GGZmod event. For each event, the table may register a handler with GGZmod to handle that event.

See also:

GGZModHandler (p. 13)

ggzmod_set_handler (p. 22)

Enumerator:

GGZMOD_EVENT_STATE Module status changed This event occurs when the game's status changes. The old state (a GGZModState*) is passed as the event's data.

See also:

GGZModState (p. 14)

GGZMOD_EVENT_SERVER A new server connection has been made This event occurs when a new connection to the game server has been made, either by the core client or by the game client itself. The fd is passed as the event's data.

See also:

ggzmod_connect (p. 15)

GGZMOD_EVENT_PLAYER The player's seat status has changed.

This event occurs when the player's seat status changes; i.e. he changes seats or starts/stops spectating. The event data is a int[2] pair consisting of the old {is_spectator, seat_num}.

GGZMOD_EVENT_SEAT A seat change.

This event occurs when a seat change occurs. The old seat (a GGZSeat*) is passed as the event's data. The seat information will be updated before the event is invoked.

GGZMOD_EVENT_SPECTATOR_SEAT A spectator seat change.

This event occurs when a spectator seat change occurs. The old spectator (a GGZSpectator*) is passed as the event's data. The spectator information will be updated before the event is invoked.

GGZMOD_EVENT_CHAT A chat message event.

This event occurs when we receive a chat. The chat may have originated in another game client or from the GGZ client; in either case it will be routed to us. The chat information (a GGZChat*) is passed as the event's data. Note that the chat may originate with a player or a spectator, and they may have changed seats or left the table by the time it gets to us.

GGZMOD_EVENT_STATS A players' stats have been updated.

See also:

ggzmod_player_get_record (p. 20)
ggzmod_player_get_rating (p. 20)
ggzmod_player_get_ranking (p. 20)
ggzmod_player_get_highscore (p. 20)

GGZMOD_EVENT_INFO Player information has arrived.

Information has been requested about one or more players and it has now arrived. The event data is a GGZPlayerInfo* structure or NULL if info about all players was requested.

GGZMOD_EVENT_ERROR An error has occurred This event occurs when a GGZMod error has occurred. An error message (a char*) will be passed as the event's data. GGZMod may attempt to recover from the error, but it is not guaranteed that the GGZ connection will continue to work after an error has happened.

4.1.3.2 enum GGZModState

Table states.

Each table has a current "state" that is tracked by ggzmod. First the table is executed and begins running. Then it receives a launch event from GGZ and begins waiting for players. At some point a game will be started and played at the table, after which it may return to waiting. Eventually the table will probably halt and then the program will exit.

More specifically, the game is in the `CREATED` state when it is first executed. It moves to the `CONNECTED` state after GGZ first communicates with it, and to `WAITING` after the connection is established with the game server. After this, the game server may use `ggzmod_set_state` to change between `WAITING`, `PLAYING`, and `DONE` states. A `WAITING` game is considered waiting for players (or whatever), while a `PLAYING` game is actively being played (this information may be, but currently is not, propagated back to GGZ for display purposes). Once the state is changed to `DONE`, the table is considered dead and will exit shortly thereafter.

Each time the game state changes, a `GGZMOD_EVENT_STATE` event will be propagated to the game server.

Enumerator:

`GGZMOD_STATE_CREATED` Initial state. The game starts out in this state. Once the state is changed it should never be changed back.

`GGZMOD_STATE_CONNECTED` Connected state. After the GGZ client and game client get connected, the game changes into this state automatically. Once this happens messages may be sent between these two. Once the game leaves this state it should never be changed back.

`GGZMOD_STATE_WAITING` Waiting state. After the game client and game server are connected, the client enters the waiting state. The game client may now call `ggzmod_set_state` to change between `WAITING`, `PLAYING`, and `DONE` states.

`GGZMOD_STATE_PLAYING` Playing state. This state is only entered after the game client changes state to it via `ggzmod_set_state`. State may be changed back and forth between `WAITING` and `PLAYING` as many times as are wanted.

`GGZMOD_STATE_DONE` Done state. Once the game client is done running, `ggzmod_set_state` should be called to set the state to done. At this point nothing "new" can happen. The state cannot be changed again after this. However the game client will not be terminated by the GGZ client; GGZ just waits for it to exit of its own volition.

4.1.3.3 enum GGZModType

The "type" of ggzmod.

The "flavor" of GGZmod object this is. Affects what operations are allowed.

Enumerator:

`GGZMOD_GGZ` Used by the ggz client ("ggz").

`GGZMOD_GAME` Used by the game client ("table").

4.1.4 Function Documentation

4.1.4.1 int ggzmod_connect (GGZMod * ggzmod)

Connect to ggz.

Call this function to make an initial GGZ core client <-> game client connection. Afterwards

Parameters:

ggzmod The ggzmod object.

Returns:

0 on success, -1 on failure.

4.1.4.2 int ggzmod_disconnect (GGZMod * *ggzmod*)

Disconnect from ggz.

This terminates the link between the game client and the GGZ core client.

Parameters:

ggzmod The ggzmod object.

Returns:

0 on success, -1 on failure.

4.1.4.3 int ggzmod_dispatch (GGZMod * *ggzmod*)

Check for and handle input.

This function handles input from the communications sockets:

- It will check for input, but will not block.
- It will monitor input from the GGZmod socket.
- It will monitor input from player sockets only if a handler is registered for the PLAYER_DATA event.
- It will call an event handler as necessary.

Parameters:

ggzmod The ggzmod object.

Returns:

-1 on error, the number of events handled (0-1) on success.

4.1.4.4 void ggzmod_free (GGZMod * *ggzmod*)

Destroy a finished ggzmod object.

After the connection is through, the object may be freed.

Parameters:

ggzmod The GGZMod object.

4.1.4.5 int ggzmod_get_fd (GGZMod * *ggzmod*)

Get the file descriptor for the GGZMod socket.

Parameters:

ggzmod The GGZMod object.

Returns:

GGZMod's main ggz <-> table socket FD.

4.1.4.6 void* ggzmod_get_gamedata (GGZMod * *ggzmod*)

Return gamedata pointer.

Each GGZMod object can be given a "gamedata" pointer that is returned by this function. This is useful for when a single process serves multiple GGZmod's.

Parameters:

ggzmod The GGZMod object.

Returns:

A pointer to the gamedata block (or NULL if none).

See also:

[ggzmod_set_gamedata](#) (p. 22)

4.1.4.7 int ggzmod_get_num_seats (GGZMod * *ggzmod*)

Get the total number of seats at the table.

Returns:

The number of seats, or -1 on error.

Note:

If no connection is present, -1 will be returned.

While in GGZMOD_STATE_CREATED, we don't know the number of seats.

4.1.4.8 int ggzmod_get_num_spectator_seats (GGZMod * *ggzmod*)

Get the maximum number of spectators. This function returns the maximum number of spectator seats available. A game can use this to iterate over the spectator seats to look for spectators occupying them. Since spectators may come and go at any point and there is no limit on the number of spectators, you should consider this value to be dynamic and call this function again each time you're looking for spectators.

Returns:

The number of available spectator seats, or -1 on error.

4.1.4.9 `const char* ggzmod_get_player (GGZMod * ggzmod, int * is_spectator, int * seat_num)`

Get data about this player.

Call this function to find out where at the table this player is sitting.

Parameters:

ggzmod The GGZMod object.

is_spectator Will be set to TRUE iff player is spectating.

seat_num Will be set to the number of our (spectator) seat.

Returns:

The name of the player (or NULL on error).

4.1.4.10 `GGZSeat ggzmod_get_seat (GGZMod * ggzmod, int seat)`

Get all data for the specified seat.

Parameters:

ggzmod The GGZMod object.

seat The seat number (0..(number of seats - 1)).

Returns:

A valid **GGZSeat** (p. 5) structure, if seat is a valid seat.

4.1.4.11 `int ggzmod_get_server_fd (GGZMod * ggzmod)`

Get the fd of the game server connection.

Parameters:

ggzmod The GGZMod object.

Returns:

The server connection fd

4.1.4.12 `GGZSpectatorSeat ggzmod_get_spectator_seat (GGZMod * ggzmod, int seat)`

Get a spectator's data.

Parameters:

ggzmod The GGZMod object.

seat The number, between 0 and (number of spectators - 1).

Returns:

A valid GGZSpectator structure, if given a valid seat.

4.1.4.13 GGZModState ggzmod_get_state (GGZMod * *ggzmod*)

Get the current state of the table.

Parameters:

ggzmod The GGZMod object.

Returns:

The state of the table.

4.1.4.14 GGZModType ggzmod_get_type (GGZMod * *ggzmod*)

Get the type of the ggzmod object.

Parameters:

ggzmod The GGZMod object.

Returns:

The type of the GGZMod object (GGZ or GAME).

4.1.4.15 int ggzmod_is_ggz_mode (void)

Is the program running in GGZ mode?

Call this function to see if the program was actually launched by GGZ. This can be used to give an error message if the executable is run outside of the GGZ environment, or for games that will run both inside and outside of GGZ.

Returns:

A boolean value indicating whether the program is running in GGZ.

Note:

Should only be called by game clients, not by GGZ itself.

4.1.4.16 GGZMod* ggzmod_new (GGZModType *type*)

Create a new ggzmod object.

Before connecting through ggzmod, a new ggzmod object is needed.

Parameters:

type The type of ggzmod. Should be GGZMOD_GAME for game servers.

See also:

GGZModType (p. 15)

4.1.4.17 `int ggzmod_player_get_highscore (GGZMod * ggzmod, GGZSeat * seat, int * highscore)`

Get the player's highscore.

Returns:

TRUE if there is a highscore; FALSE if not or on error.

4.1.4.18 `GGZPlayerInfo* ggzmod_player_get_info (GGZMod * ggzmod, int seat)`

Get the extended information for the specified seat.

Parameters:

ggzmod The GGZMod object.

seat The seat number (0..(number of seats - 1)).

Returns:

A valid GGZPlayerInfo structure, if seat is valid and has info.

4.1.4.19 `int ggzmod_player_get_ranking (GGZMod * ggzmod, GGZSeat * seat, int * ranking)`

Get the player's ranking.

Returns:

TRUE if there is a ranking; FALSE if not or on error.

4.1.4.20 `int ggzmod_player_get_rating (GGZMod * ggzmod, GGZSeat * seat, int * rating)`

Get the player's rating.

Returns:

TRUE if there is a rating; FALSE if not or on error.

4.1.4.21 `int ggzmod_player_get_record (GGZMod * ggzmod, GGZSeat * seat, int * wins, int * losses, int * ties, int * forfeits)`

Get the player's win-loss record.

Returns:

TRUE if there is a record; FALSE if not or on error.

4.1.4.22 int ggzmod_player_request_info (GGZMod * *ggzmod*, int *seat_num*)

Request extended player information for one or more players.

Depending on the seat parameter (-1 or valid number), this function asynchronously requests information about player(s), which will arrive with a GGZMOD_EVENT_INFO event.

Parameters:

ggzmod The ggzmod object.

seat_num The seat number to request info for, or -1 to select all.

Returns:

TRUE if seat is -1 or valid number, FALSE for non-player seats.

4.1.4.23 void ggzmod_request_boot (GGZMod * *ggzmod*, const char * *name*)

Boot a player. Only the game host may do this.

Parameters:

ggzmod The ggzmod object.

name The name of the player to boot.

4.1.4.24 void ggzmod_request_bot (GGZMod * *ggzmod*, int *seat_num*)

Change the requested seat from an open seat to a bot.

Parameters:

ggzmod The ggzmod object.

seat_num The number of the seat to toggle.

4.1.4.25 void ggzmod_request_chat (GGZMod * *ggzmod*, const char * *chat_msg*)

Chat! This initiates a table chat.

Parameters:

ggzmod The ggzmod object.

chat_msg The chat message.

Note:

The chat message should be in UTF-8.

4.1.4.26 void ggzmod_request_open (GGZMod * *ggzmod*, int *seat_num*)

Change the requested seat from a bot to an open seat.

Parameters:

ggzmod The ggzmod object.

seat_num The number of the seat to toggle.

4.1.4.27 void ggzmod_request_sit (GGZMod * *ggzmod*, int *seat_num*)

Sit down (move from a spectator seat into a player seat).

Parameters:

ggzmod The ggzmod object.

seat_num The seat to sit in.

4.1.4.28 void ggzmod_request_stand (GGZMod * *ggzmod*)

Stand up (move from your seat into a spectator seat).

Parameters:

ggzmod The ggzmod object.

4.1.4.29 void ggzmod_set_gamedata (GGZMod * *ggzmod*, void * *data*)

Set gamedata pointer.

Parameters:

ggzmod The GGZMod object.

data The gamedata block (or NULL for none).

See also:

[ggzmod_get_gamedata](#) (p. 17)

4.1.4.30 void ggzmod_set_handler (GGZMod * *ggzmod*, GGZModEvent *e*, GGZModHandler *func*)

Set a handler for the given event.

As described above, GGZmod uses an event-driven structure. Each time an event is called, the event handler (there can be only one) for that event will be called. This function registers such an event handler.

Parameters:

ggzmod The GGZmod object.

e The GGZmod event.

func The handler function being registered.

See also:

`ggzmod_get_gamedata` (p. 17)

4.1.4.31 `int ggzmod_set_state (GGZMod * ggzmod, GGZModState state)`

Change the table's state.

This function should be called to change the state of a table. A game can use this function to change state between WAITING and PLAYING, or to set it to DONE.

Parameters:

ggzmod The ggzmod object.

state The new state.

Returns:

0 on success, -1 on failure/error.

4.1.4.32 `int ggzmod_spectator_get_highscore (GGZMod * ggzmod, GGZSpectatorSeat * seat, int * highscore)`

Get the spectator's highscore.

Returns:

TRUE if there is a highscore; FALSE if not or on error.

4.1.4.33 `int ggzmod_spectator_get_ranking (GGZMod * ggzmod, GGZSpectatorSeat * seat, int * ranking)`

Get the spectator's ranking.

Returns:

TRUE if there is a ranking; FALSE if not or on error.

4.1.4.34 `int ggzmod_spectator_get_rating (GGZMod * ggzmod, GGZSpectatorSeat * seat, int * rating)`

Get the spectator's rating.

Returns:

TRUE if there is a rating; FALSE if not or on error.

4.1.4.35 `int ggzmod_spectator_get_record (GGZMod * ggzmod,
GGZSpectatorSeat * seat, int * wins, int * losses, int * ties, int * forfeits)`

Get the spectator's win-loss record.

Returns:

TRUE if there is a record; FALSE if not or on error.

Index

GGZMod
 ggzmod.h, 13
ggzmod.h, 9
 GGZMod, 13
 ggzmod_connect, 15
 ggzmod_disconnect, 16
 ggzmod_dispatch, 16
 GGZMOD_EVENT_CHAT, 14
 GGZMOD_EVENT_ERROR, 14
 GGZMOD_EVENT_INFO, 14
 GGZMOD_EVENT_PLAYER, 14
 GGZMOD_EVENT_SEAT, 14
 GGZMOD_EVENT_SERVER, 14
 GGZMOD_EVENT_SPECTATOR_-
 SEAT, 14
 GGZMOD_EVENT_STATE, 14
 GGZMOD_EVENT_STATS, 14
 ggzmod_free, 16
 GGZMOD_GAME, 15
 ggzmod_get_fd, 16
 ggzmod_get_gamedata, 17
 ggzmod_get_num_seats, 17
 ggzmod_get_num_spectator_seats, 17
 ggzmod_get_player, 17
 ggzmod_get_seat, 18
 ggzmod_get_server_fd, 18
 ggzmod_get_spectator_seat, 18
 ggzmod_get_state, 18
 ggzmod_get_type, 19
 GGZMOD_GGZ, 15
 ggzmod_is_ggz_mode, 19
 ggzmod_new, 19
 ggzmod_player_get_highscore, 19
 ggzmod_player_get_info, 20
 ggzmod_player_get_ranking, 20
 ggzmod_player_get_rating, 20
 ggzmod_player_get_record, 20
 ggzmod_player_request_info, 20
 ggzmod_request_boot, 21
 ggzmod_request_bot, 21
 ggzmod_request_chat, 21
 ggzmod_request_open, 21
 ggzmod_request_sit, 22
 ggzmod_request_stand, 22
 ggzmod_set_gamedata, 22
 ggzmod_set_handler, 22
 ggzmod_set_state, 23
 ggzmod_spectator_get_highscore, 23
 ggzmod_spectator_get_ranking, 23
 ggzmod_spectator_get_rating, 23
 ggzmod_spectator_get_record, 23
 GGZMOD_STATE_CONNECTED, 15
 GGZMOD_STATE_CREATED, 15
 GGZMOD_STATE_DONE, 15
 GGZMOD_STATE_PLAYING, 15
 GGZMOD_STATE_WAITING, 15
 GGZModEvent, 13
 GGZModHandler, 13
 GGZModState, 14
 GGZModType, 15
ggzmod_connect
 ggzmod.h, 15
ggzmod_disconnect
 ggzmod.h, 16
ggzmod_dispatch
 ggzmod.h, 16
GGZMOD_EVENT_CHAT
 ggzmod.h, 14
GGZMOD_EVENT_ERROR
 ggzmod.h, 14
GGZMOD_EVENT_INFO
 ggzmod.h, 14
GGZMOD_EVENT_PLAYER
 ggzmod.h, 14
GGZMOD_EVENT_SEAT
 ggzmod.h, 14
GGZMOD_EVENT_SERVER
 ggzmod.h, 14
GGZMOD_EVENT_SPECTATOR_SEAT
 ggzmod.h, 14
GGZMOD_EVENT_STATE
 ggzmod.h, 14
GGZMOD_EVENT_STATS
 ggzmod.h, 14
ggzmod_free
 ggzmod.h, 16
GGZMOD_GAME
 ggzmod.h, 15
ggzmod_get_fd
 ggzmod.h, 16

ggzmod_get_gamedata
 ggzmod.h, 17
 ggzmod_get_num_seats
 ggzmod.h, 17
 ggzmod_get_num_spectator_seats
 ggzmod.h, 17
 ggzmod_get_player
 ggzmod.h, 17
 ggzmod_get_seat
 ggzmod.h, 18
 ggzmod_get_server_fd
 ggzmod.h, 18
 ggzmod_get_spectator_seat
 ggzmod.h, 18
 ggzmod_get_state
 ggzmod.h, 18
 ggzmod_get_type
 ggzmod.h, 19
 GGZMOD_GGZ
 ggzmod.h, 15
 ggzmod_is_ggz_mode
 ggzmod.h, 19
 ggzmod_new
 ggzmod.h, 19
 ggzmod_player_get_highscore
 ggzmod.h, 19
 ggzmod_player_get_info
 ggzmod.h, 20
 ggzmod_player_get_ranking
 ggzmod.h, 20
 ggzmod_player_get_rating
 ggzmod.h, 20
 ggzmod_player_get_record
 ggzmod.h, 20
 ggzmod_player_request_info
 ggzmod.h, 20
 ggzmod_request_boot
 ggzmod.h, 21
 ggzmod_request_bot
 ggzmod.h, 21
 ggzmod_request_chat
 ggzmod.h, 21
 ggzmod_request_open
 ggzmod.h, 21
 ggzmod_request_sit
 ggzmod.h, 22
 ggzmod_request_stand
 ggzmod.h, 22
 ggzmod_set_gamedata
 ggzmod.h, 22
 ggzmod_set_handler
 ggzmod.h, 22
 ggzmod_set_state
 ggzmod.h, 23
 ggzmod_spectator_get_highscore
 ggzmod.h, 23
 ggzmod_spectator_get_ranking
 ggzmod.h, 23
 ggzmod_spectator_get_rating
 ggzmod.h, 23
 ggzmod_spectator_get_record
 ggzmod.h, 23
 GGZMOD_STATE_CONNECTED
 ggzmod.h, 15
 GGZMOD_STATE_CREATED
 ggzmod.h, 15
 GGZMOD_STATE_DONE
 ggzmod.h, 15
 GGZMOD_STATE_PLAYING
 ggzmod.h, 15
 GGZMOD_STATE_WAITING
 ggzmod.h, 15
 GGZModEvent
 ggzmod.h, 13
 GGZModHandler
 ggzmod.h, 13
 GGZModState
 ggzmod.h, 14
 GGZModType
 ggzmod.h, 15
 GGZSeat, 5
 name, 5
 num, 5
 type, 5
 GGZSpectatorSeat, 7
 GGZSpectatorSeat
 name, 7
 num, 7
 name
 GGZSeat, 5
 GGZSpectatorSeat, 7
 num
 GGZSeat, 5
 GGZSpectatorSeat, 7
 type
 GGZSeat, 5