

GGZCore Reference Manual

0.0.14

Generated by Doxygen 1.5.1

Fri Nov 30 14:59:02 2007

Contents

1 GGZCore Data Structure Index	1
1.1 GGZCore Data Structures	1
2 GGZCore File Index	3
2.1 GGZCore File List	3
3 GGZCore Page Index	5
3.1 GGZCore Related Pages	5
4 GGZCore Data Structure Documentation	7
4.1 _GGZOptions Struct Reference	7
4.2 GGZChatEventData Struct Reference	8
4.3 GGZErrorEventData Struct Reference	9
4.4 GGZMotdEventData Struct Reference	10
4.5 GGZRoomChangeEventData Struct Reference	11
4.6 GGZTableLeaveEventData Struct Reference	12
5 GGZCore File Documentation	13
5.1 ggzcore.h File Reference	13
6 GGZCore Page Documentation	57
6.1 Todo List	57

Chapter 1

GGZCore Data Structure Index

1.1 GGZCore Data Structures

Here are the data structures with brief descriptions:

_GGZOptions	7
GGZChatEventData (The data associated with a GGZ_CHAT_EVENT room event)	8
GGZErrorEventData (The data describing an error)	9
GGZMotdEventData (The data associated with a GGZ_MOTD_LOADED server event)	10
GGZRoomChangeEventData (The data associated with GGZ_ROOM_ENTER/GGZ_ROOM_LEAVE events)	11
GGZTableLeaveEventData (The data associated with a GGZ_TABLE_LEFT room event)	12

Chapter 2

GGZCore File Index

2.1 GGZCore File List

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

ggzcore.h (The interface for the ggzcore library used by GGZ clients)	13
--	----

Chapter 3

GGZCore Page Index

3.1 GGZCore Related Pages

Here is a list of all related documentation pages:

Todo List	57
---------------------	----

Chapter 4

GGZCore Data Structure Documentation

4.1 `_GGZOptions` Struct Reference

```
#include <ggzcore.h>
```

Data Fields

- `GGZOptionFlags` flags

4.1.1 Detailed Description

Options structure for ggzcore library

4.1.2 Field Documentation

4.1.2.1 `GGZOptionFlags` `_GGZOptions::flags`

Option flags

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

- `ggzcore.h`

4.2 GGZChatEventData Struct Reference

The data associated with a GGZ_CHAT_EVENT room event.

```
#include <ggzcore.h>
```

Data Fields

- GGZChatType **type**
- const char * **sender**
- const char * **message**

4.2.1 Detailed Description

The data associated with a GGZ_CHAT_EVENT room event.

4.2.2 Field Documentation

4.2.2.1 GGZChatType GGZChatEventData::type

The type of chat.

4.2.2.2 const char* GGZChatEventData::sender

The person who sent the message, or NULL

4.2.2.3 const char* GGZChatEventData::message

The message itself (in UTF-8), or NULL

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

- **ggzcore.h**

4.3 GGZErrorEventData Struct Reference

The data describing an error.

```
#include <ggzcore.h>
```

Data Fields

- GGZClientReqError **status**
The type of error that occurred.
- const char * **message**
A default error message.

4.3.1 Detailed Description

The data describing an error.

When an error occurs, a pointer to a struct of this type will be passed as the event data.

4.3.2 Field Documentation

4.3.2.1 GGZClientReqError GGZErrorEventData::status

The type of error that occurred.

Note:

Not all errors are possible with all events.

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

- **ggzcore.h**

4.4 GGZMotdEventData Struct Reference

The data associated with a GGZ_MOTD_LOADED server event.

```
#include <ggzcore.h>
```

Data Fields

- const char * **motd**
- const char * **url**

4.4.1 Detailed Description

The data associated with a GGZ_MOTD_LOADED server event.

4.4.2 Field Documentation

4.4.2.1 const char* GGZMotdEventData::motd

MOTD text message

4.4.2.2 const char* GGZMotdEventData::url

URL of a graphical MOTD webpage, or NULL

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

- **ggzcore.h**

4.5 GGZRoomChangeEventData Struct Reference

The data associated with GGZ_ROOM_ENTER/GGZ_ROOM_LEAVE events.

```
#include <ggzcore.h>
```

Data Fields

- **const char * player_name**
The name of the player entering/leaving.
- **int rooms_known**
- **GGZRoom * to_room**
The room we are entering.
- **GGZRoom * from_room**
The room the player is leaving.

4.5.1 Detailed Description

The data associated with GGZ_ROOM_ENTER/GGZ_ROOM_LEAVE events.

4.5.2 Field Documentation

4.5.2.1 GGZRoom* GGZRoomChangeEventData::to_room

The room we are entering.

This may be NULL if the player is leaving the server or if the info is unknown.

4.5.2.2 GGZRoom* GGZRoomChangeEventData::from_room

The room the player is leaving.

This may be NULL if the player is just entering the server or if the info is unknown.

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

- **ggzcore.h**

4.6 GGZTableLeaveEventData Struct Reference

The data associated with a GGZ_TABLE_LEFT room event.

```
#include <ggzcore.h>
```

Data Fields

- GGZLeaveType **reason**
This is the reason for why we left.
- const char * **player**

4.6.1 Detailed Description

The data associated with a GGZ_TABLE_LEFT room event.

4.6.2 Field Documentation

4.6.2.1 const char* GGZTableLeaveEventData::player

If we were booted from the table (GGZ_LEAVE_BOOT), this is the player who kicked us out.

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

- **ggzcore.h**

Chapter 5

GGZCore File Documentation

5.1 ggzcore.h File Reference

The interface for the ggzcore library used by GGZ clients.

```
#include <stdarg.h>
#include <sys/types.h>
#include <ggz_common.h>
```

Data Structures

- struct **_GGZOptions**
- struct **GGZErrorEventData**
The data describing an error.
- struct **GGZMotdEventData**
The data associated with a GGZ_MOTD_LOADED server event.
- struct **GGZChatEventData**
The data associated with a GGZ_CHAT_EVENT room event.
- struct **GGZTableLeaveEventData**
The data associated with a GGZ_TABLE_LEFT room event.
- struct **GGZRoomChangeEventData**
The data associated with GGZ_ROOM_ENTER/GGZ_ROOM_LEAVE events.

Defines

- #define **GGZCORE_VERSION_MAJOR** 0
- #define **GGZCORE_VERSION_MINOR** 0
- #define **GGZCORE_VERSION_MICRO** 14
- #define **GGZCORE_VERSION_IFACE** "9:0:0"
- #define **GGZCORE_DBG_CONF** "GGZCORE:CONF"

ggz_debug debugging type for configuration system.

- #define **GGZCORE_DBG_GAME** "GGZCORE:GAME"
ggz_debug debugging type for game communication.
- #define **GGZCORE_DBG_HOOK** "GGZCORE:HOOK"
ggz_debug debugging type for hook system.
- #define **GGZCORE_DBG_MODULE** "GGZCORE:MODULE"
ggz_debug debugging type for accessing modules.
- #define **GGZCORE_DBG_NET** "GGZCORE:NET"
ggz_debug debugging type for network interaction.
- #define **GGZCORE_DBG_POLL** "GGZCORE:POLL"
ggz_debug debugging type for debugging while polling.
- #define **GGZCORE_DBG_ROOM** "GGZCORE:ROOM"
ggz_debug debugging type for room events and data.
- #define **GGZCORE_DBG_SERVER** "GGZCORE:SERVER"
ggz_debug debugging type for server events and data.
- #define **GGZCORE_DBG_STATE** "GGZCORE:STATE"
ggz_debug debugging type for state changes.
- #define **GGZCORE_DBG_TABLE** "GGZCORE:TABLE"
ggz_debug debugging type for table data.
- #define **GGZCORE_DBG_XML** "GGZCORE:XML"
ggz_debug debugging type for XML processing.

Typedefs

- typedef **_GGZOptions GGZOptions**
- typedef **_GGZServer GGZServer**
A server object containing all information about a connection.
- typedef **_GGZRoom GGZRoom**
Contains information about a single room on a server.
- typedef **_GGZPlayer GGZPlayer**
Contains information about a single player.
- typedef **_GGZTable GGZTable**
Contains information about a single table.
- typedef **_GGZGameType GGZGameType**
Contains information about a `_game type_`.

- typedef `_GGZModule` **GGZModule**
Contains information about a single module. A game module, on the client, is an executable designed to play a game. Each game type may have many modules that play it.
- typedef `_GGZGame` **GGZGame**
Contains information about a single game table. This contains information about a table we are present at or are about to launch. It is thus associated with both a GGZTable and a GGZModule.
- typedef **GGZHookReturn**(*) **GGZHookFunc** (unsigned int id, const void *event_data, const void *user_data)
- typedef void(*) **GGZDestroyFunc** (void *data)
GGZ object destroy function type.

Enumerations

- enum **GGZOptionFlags** {
GGZ_OPT_PARSER = 0x0001, **GGZ_OPT_MODULES** = 0x0002, **GGZ_OPT_THREADED_IO** = 0x0004, **GGZ_OPT_EMBEDDED** = 0x0008,
GGZ_OPT_RECONNECT = 0x0010 }
- enum **GGZHookReturn** { **GGZ_HOOK_OK**, **GGZ_HOOK_REMOVE**, **GGZ_HOOK_ERROR**, **GGZ_HOOK_CRISIS** }
- enum **GGZLoginType** { **GGZ_LOGIN**, **GGZ_LOGIN_GUEST**, **GGZ_LOGIN_NEW** }
- enum **GGZServerEvent** {
GGZ_CONNECTED, **GGZ_CONNECT_FAIL**, **GGZ_NEGOTIATED**,
GGZ_NEGOTIATE_FAIL,
GGZ_LOGGED_IN, **GGZ_LOGIN_FAIL**, **GGZ_MOTD_LOADED**, **GGZ_ROOM_LIST**,
GGZ_TYPE_LIST, **GGZ_SERVER_PLAYERS_CHANGED**, **GGZ_ENTERED**, **GGZ_ENTER_FAIL**,
GGZ_LOGOUT, **GGZ_NET_ERROR**, **GGZ_PROTOCOL_ERROR**, **GGZ_CHAT_FAIL**,
GGZ_STATE_CHANGE, **GGZ_CHANNEL_CONNECTED**, **GGZ_CHANNEL_READY**, **GGZ_CHANNEL_FAIL**,
GGZ_SERVER_ROOMS_CHANGED, **GGZ_NUM_SERVER_EVENTS** }
- enum **GGZRoomEvent** {
GGZ_PLAYER_LIST, **GGZ_TABLE_LIST**, **GGZ_CHAT_EVENT**, **GGZ_ROOM_ENTER**,
GGZ_ROOM_LEAVE, **GGZ_TABLE_UPDATE**, **GGZ_TABLE_LAUNCHED**, **GGZ_TABLE_LAUNCH_FAIL**,
GGZ_TABLE_JOINED, **GGZ_TABLE_JOIN_FAIL**, **GGZ_TABLE_LEFT**, **GGZ_TABLE_LEAVE_FAIL**,
GGZ_PLAYER_LAG, **GGZ_PLAYER_STATS**, **GGZ_PLAYER_COUNT**, **GGZ_PLAYER_PERMS** }

- enum **GGZGameEvent** {
GGZ_GAME_LAUNCHED, **GGZ_GAME_LAUNCH_FAIL**, **GGZ_GAME_NEGOTIATED**, **GGZ_GAME_NEGOTIATE_FAIL**,
GGZ_GAME_PLAYING }
- enum **GGZStateID** {
GGZ_STATE_OFFLINE, **GGZ_STATE_CONNECTING**, **GGZ_STATE_RECONNECTING**, **GGZ_STATE_ONLINE**,
GGZ_STATE_LOGGING_IN, **GGZ_STATE_LOGGED_IN**, **GGZ_STATE_ENTERING_ROOM**, **GGZ_STATE_IN_ROOM**,
GGZ_STATE_BETWEEN_ROOMS, **GGZ_STATE_LAUNCHING_TABLE**, **GGZ_STATE_JOINING_TABLE**, **GGZ_STATE_AT_TABLE**,
GGZ_STATE_LEAVING_TABLE, **GGZ_STATE_LOGGING_OUT** }

The states a server connection may be in.

- enum **GGZModuleEnvironment** {
GGZ_ENVIRONMENT_PASSIVE, **GGZ_ENVIRONMENT_CONSOLE**,
GGZ_ENVIRONMENT_FRAMEBUFFER, **GGZ_ENVIRONMENT_XWINDOW**,
GGZ_ENVIRONMENT_XFULLSCREEN }

The environment a game frontend expects.

Functions

- int **ggzcore_init** (**GGZOptions** options)
- void **ggzcore_reload** (void)
- void **ggzcore_destroy** (void)
- **GGZServer** * **ggzcore_server_new** (void)
Create a new server object.
- int **ggzcore_server_reset** (**GGZServer** *server)
Reset the server object.
- int **ggzcore_server_add_event_hook** (**GGZServer** *server, const **GGZServerEvent** event, const **GGZHookFunc** func)
Register a callback handler for a server event.
- int **ggzcore_server_add_event_hook_full** (**GGZServer** *server, const **GGZServerEvent** event, const **GGZHookFunc** func, const void *data)
Register a callback handler for a server event.
- int **ggzcore_server_remove_event_hook** (**GGZServer** *server, const **GGZServerEvent** event, const **GGZHookFunc** func)
Remove a single hook function from an event's hook list.
- int **ggzcore_server_remove_event_hook_id** (**GGZServer** *server, const **GGZServerEvent** event, const unsigned int hook_id)
Remove a hook function with given ID from the event's hook list.

- `int ggzcore_server_set_hostinfo (GGZServer *server, const char *host, const unsigned int port, const unsigned int use_tls)`
Set host info for connecting to the server.
- `int ggzcore_server_set_logininfo (GGZServer *server, const GGZLoginType type, const char *handle, const char *password, const char *email)`
Set login info for logging in to the server.
- `int ggzcore_server_log_session (GGZServer *server, const char *filename)`
Initiate logging of ggzcore events.
- `const char * ggzcore_server_get_host (const GGZServer *server)`
Get the hostname of the server.
- `int ggzcore_server_get_port (const GGZServer *server)`
Get the port of the server.
- `GGZLoginType ggzcore_server_get_type (const GGZServer *server)`
Get the login type being used for this server.
- `const char * ggzcore_server_get_handle (const GGZServer *server)`
Get the handle being used for this server.
- `const char * ggzcore_server_get_password (const GGZServer *server)`
Get the password being used for this server.
- `int ggzcore_server_get_fd (const GGZServer *server)`
Get the socket used for connection with the server.
- `int ggzcore_server_get_channel (GGZServer *server)`
Get the socket used for direct game connections.
- `GGZStateID ggzcore_server_get_state (const GGZServer *server)`
Get the state of the server connection.
- `int ggzcore_server_get_tls (const GGZServer *server)`
Get the current TLS status of this server.
- `int ggzcore_server_get_num_players (const GGZServer *server)`
Get the total number of players on the server.
- `int ggzcore_server_get_num_rooms (const GGZServer *server)`
Return the number of rooms on the server, or -1 on error.
- `GGZRoom * ggzcore_server_get_nth_room (const GGZServer *server, const unsigned int num)`
Return the nth room on the server, or NULL on error.

- `int ggzcore_server_get_room_num` (const **GGZServer** *server, const **GGZRoom** *room)
Return the number (position in the room list) of the room.
- `GGZRoom * ggzcore_server_get_cur_room` (const **GGZServer** *server)
Return the current room, or NULL if there is none.
- `GGZPlayer * ggzcore_server_get_player` (**GGZServer** *server, const char *name)
Find the player, by name (or NULL).
- `int ggzcore_server_get_num_gametypes` (const **GGZServer** *server)
Return the overall number of game types on the server.
- `GGZGameType * ggzcore_server_get_nth_gametype` (const **GGZServer** *server, const unsigned int num)
Get the nth gametype, or NULL on error.
- `GGZGame * ggzcore_server_get_cur_game` (const **GGZServer** *server)
Return the player's current game.
- `int ggzcore_server_is_online` (const **GGZServer** *server)
Return TRUE iff the server is online (connected?).
- `int ggzcore_server_is_logged_in` (const **GGZServer** *server)
Return TRUE iff we are logged into the server.
- `int ggzcore_server_is_in_room` (const **GGZServer** *server)
Return TRUE iff we are in a room on the server.
- `int ggzcore_server_is_at_table` (const **GGZServer** *server)
Return TRUE iff we are at a table on the server.
- `int ggzcore_server_connect` (**GGZServer** *server)
Connect to the server.
- `int ggzcore_server_create_channel` (**GGZServer** *server)
Establish a direct connection.
- `int ggzcore_server_login` (**GGZServer** *server)
Log in to the server.
- `int ggzcore_server_motd` (**GGZServer** *server)
Request the MOTD from the server.
- `int ggzcore_server_list_rooms` (**GGZServer** *server, const int type, const char verbose)
Request room list.
- `int ggzcore_server_list_gametypes` (**GGZServer** *server, const char verbose)
Request game type list.

- **int ggzcore_server_join_room** (**GGZServer** *server, **GGZRoom** *room)
Join a room on the server.
- **int ggzcore_server_logout** (**GGZServer** *server)
Log out of a server.
- **int ggzcore_server_disconnect** (**GGZServer** *server)
Disconnect from a server after having logged out.
- **int ggzcore_server_data_is_pending** (**GGZServer** *server)
Check for data pending from the server socket.
- **int ggzcore_server_read_data** (**GGZServer** *server, int fd)
Read data for the server on the specified FD.
- **void ggzcore_server_free** (**GGZServer** *server)
Free GGZServer object and accompanying data.
- **GGZRoom * ggzcore_room_new** (void)
Allocate space for a new room object.
- **void ggzcore_room_free** (**GGZRoom** *room)
De-allocate room object and its children.
- **GGZServer * ggzcore_room_get_server** (const **GGZRoom** *room)
Return the server for this room (or NULL on error).
- **const char * ggzcore_room_get_name** (const **GGZRoom** *room)
Return the name of the room (or NULL on error).
- **const char * ggzcore_room_get_desc** (const **GGZRoom** *room)
Return the description of the room (or NULL on error).
- **GGZGameType * ggzcore_room_get_gametype** (const **GGZRoom** *room)
Return the type of game played in this room (or NULL on error).
- **int ggzcore_room_get_num_players** (const **GGZRoom** *room)
Return the number of players in the room (or negative on error).
- **GGZPlayer * ggzcore_room_get_nth_player** (const **GGZRoom** *room, const unsigned int num)
Return the nth player in the room (or NULL on error).
- **int ggzcore_room_get_num_tables** (const **GGZRoom** *room)
Return the number of tables in the room (or negative on error).
- **GGZTable * ggzcore_room_get_nth_table** (const **GGZRoom** *room, const unsigned int num)
Return the nth table in the room (or NULL on error).

- **GGZTable * ggzcore_room_get_table_by_id** (const **GGZRoom** *room, const unsigned int id)
Return the table in this room with matching ID (NULL on error).
- **int ggzcore_room_get_closed** (const **GGZRoom** *room)
Return whether this room is closed (1), or open as usual (0).
- **int ggzcore_room_add_event_hook** (**GGZRoom** *room, const **GGZRoomEvent** event, const **GGZHookFunc** func)
Register a handler (hook) for the room event.
- **int ggzcore_room_add_event_hook_full** (**GGZRoom** *room, const **GGZRoomEvent** event, const **GGZHookFunc** func, const void *data)
Register a handler (hook) for the room event, with data.
- **int ggzcore_room_remove_event_hook** (**GGZRoom** *room, const **GGZRoomEvent** event, const **GGZHookFunc** func)
Remove a hook from an event.
- **int ggzcore_room_remove_event_hook_id** (**GGZRoom** *room, const **GGZRoomEvent** event, const unsigned int hook_id)
Remove a hook from an event, by ID.
- **int ggzcore_room_list_players** (**GGZRoom** *room)
Call to request a list of players in the room.
- **int ggzcore_room_list_tables** (**GGZRoom** *room, const int type, const char global)
Call to request a list of tables in the room.
- **int ggzcore_room_chat** (**GGZRoom** *room, const GGZChatType opcode, const char *player, const char *msg)
Chat!
- **int ggzcore_room_admin** (**GGZRoom** *room, GGZAdminType type, const char *player, const char *reason)
Server- and room-level administrative actions.
- **int ggzcore_room_launch_table** (**GGZRoom** *room, **GGZTable** *table)
Launch a table in the room.
- **int ggzcore_room_join_table** (**GGZRoom** *room, const unsigned int table_id, int spectator)
Join a table in the room, so that you can then play at it.
- **int ggzcore_room_leave_table** (**GGZRoom** *room, int force)
Leave the table you are currently playing at.
- **char * ggzcore_player_get_name** (const **GGZPlayer** *player)
Return the name of the player.

- **GGZPlayerType ggzcore_player_get_type** (const **GGZPlayer** *player)
Return the type of the player (admin/registered/guest).
- **GGZRoom * ggzcore_player_get_room** (const **GGZPlayer** *player)
Return the player's room, or NULL if none.
- **GGZTable * ggzcore_player_get_table** (const **GGZPlayer** *player)
Return the player's table, or NULL if none.
- **bool ggzcore_player_has_perm** (const **GGZPlayer** *player, **GGZPerm** perm)
Return true iff the player has this permission.
- **int ggzcore_player_set_perm** (**GGZPlayer** *player, **GGZPerm** perm, bool set)
Administer player permissions.
- **int ggzcore_player_get_lag** (const **GGZPlayer** *player)
Return the player's lag class (1..5).
- **int ggzcore_player_get_record** (const **GGZPlayer** *player, int *wins, int *losses, int *ties, int *forfeits)
Get the player's win-loss record.
- **int ggzcore_player_get_rating** (const **GGZPlayer** *player, int *rating)
Get the player's rating.
- **int ggzcore_player_get_ranking** (const **GGZPlayer** *player, int *ranking)
Get the player's ranking.
- **int ggzcore_player_get_highscore** (const **GGZPlayer** *player, int *highscore)
Get the player's highscore.
- **GGZTable * ggzcore_table_new** (void)
Create a new table object.
- **int ggzcore_table_init** (**GGZTable** *table, const **GGZGameType** *gametype, const char *desc, const unsigned int num_seats)
Set data on a table object.
- **void ggzcore_table_free** (**GGZTable** *table)
Free the table object.
- **int ggzcore_table_set_seat** (**GGZTable** *table, const unsigned int seat, **GGZSeatType** type, const char *name)
Set a seat type at a table, pre-launch.
- **int ggzcore_table_remove_player** (**GGZTable** *table, const char *name)
Find and remove the player from the table.
- **int ggzcore_table_get_id** (const **GGZTable** *table)

Return the ID of the table.

- `const GGZRoom * ggzcore_table_get_room (const GGZTable *table)`
Return the room this table is in.
- `const GGZGameType * ggzcore_table_get_type (const GGZTable *table)`
Return the game type of the table.
- `const char * ggzcore_table_get_desc (const GGZTable *table)`
Return the table's description (or NULL).
- `GGZTableState ggzcore_table_get_state (const GGZTable *table)`
Return the state of the table.
- `int ggzcore_table_get_num_seats (const GGZTable *table)`
Return the number of seats at the table.
- `int ggzcore_table_set_desc (GGZTable *table, const char *desc)`
Set the table description.
- `int ggzcore_table_get_seat_count (const GGZTable *table, GGZSeatType type)`
Count the seats of the given type.
- `const char * ggzcore_table_get_nth_player_name (const GGZTable *table, const unsigned int num)`
Return the name of a player at the table, or NULL on error.
- `int ggzcore_table_get_num_spectator_seats (const GGZTable *table)`
Return the number of spectator seats at the table, or -1.
- `const char * ggzcore_table_get_nth_spectator_name (const GGZTable *table, const unsigned int num)`
Return the name of the nth spectator, or NULL if seat is empty.
- `GGZSeatType ggzcore_table_get_nth_player_type (const GGZTable *table, const unsigned int num)`
Return the type of a player at the table, or GGZ_PLAYER_NONE on error.
- `unsigned int ggzcore_gametype_get_id (const GGZGameType *type)`
Get the ID of this gametype.
- `const char * ggzcore_gametype_get_name (const GGZGameType *type)`
Get the name of the game type.
- `const char * ggzcore_gametype_get_prot_engine (const GGZGameType *type)`
Get the protocol "engine" used by the game type.
- `const char * ggzcore_gametype_get_prot_version (const GGZGameType *type)`
Get the version of the protocol the game uses.

- `const char * ggzcore_gametype_get_version (const GGZGameType *type)`
Get the version of the game itself.
- `const char * ggzcore_gametype_get_author (const GGZGameType *type)`
Get the author of the game.
- `const char * ggzcore_gametype_get_url (const GGZGameType *type)`
Get a URL for more info about the game.
- `const char * ggzcore_gametype_get_desc (const GGZGameType *type)`
Get a description of the game.
- `int ggzcore_gametype_get_max_players (const GGZGameType *type)`
Get the maximum number of players the game can support.
- `int ggzcore_gametype_get_max_bots (const GGZGameType *type)`
Get the maximum number of bots the game can support.
- `int ggzcore_gametype_get_spectators_allowed (const GGZGameType *type)`
Return TRUE iff spectators are allowed for this game type.
- `int ggzcore_gametype_num_players_is_valid (const GGZGameType *type, unsigned int num)`
Return TRUE iff the given number of players is valid.
- `int ggzcore_gametype_num_bots_is_valid (const GGZGameType *type, unsigned int num)`
Return TRUE iff the given number of bots is valid.
- `int ggzcore_gametype_get_num_namedbots (const GGZGameType *type)`
Return the number of named bots for this gametype.
- `const char * ggzcore_gametype_get_namedbot_name (const GGZGameType *type, unsigned int num)`
- `const char * ggzcore_gametype_get_namedbot_class (const GGZGameType *type, unsigned int num)`
- `int ggzcore_gametype_get_peers_allowed (const GGZGameType *type)`
Return TRUE iff this game may disclose the player's hostname.
- `int ggzcore_conf_initialize (const char *g_path, const char *u_path)`
- `int ggzcore_conf_write_string (const char *section, const char *key, const char *value)`
- `int ggzcore_conf_write_int (const char *section, const char *key, int value)`
- `int ggzcore_conf_write_list (const char *section, const char *key, int argc, char **argv)`
- `char * ggzcore_conf_read_string (const char *section, const char *key, const char *def)`
- `int ggzcore_conf_read_int (const char *section, const char *key, int def)`
- `int ggzcore_conf_read_list (const char *section, const char *key, int *argc, char ***argvp)`
- `int ggzcore_conf_remove_section (const char *section)`
- `int ggzcore_conf_remove_key (const char *section, const char *key)`
- `int ggzcore_conf_commit (void)`

- unsigned int **ggzcore_module_get_num** (void)
This returns the number of registered modules.
- int **ggzcore_module_add** (const char *name, const char *version, const char *prot_engine, const char *prot_version, const char *author, const char *frontend, const char *url, const char *exe_path, const char *icon_path, const char *help_path, **GGZModuleEnvironment** environment)
- int **ggzcore_module_get_num_by_type** (const char *game, const char *engine, const char *version)
Returns how many modules support this game and protocol.
- **GGZModule** * **ggzcore_module_get_nth_by_type** (const char *game, const char *engine, const char *version, const unsigned int num)
Returns n-th module that supports this game and protocol.
- const char * **ggzcore_module_get_name** (**GGZModule** *module)
Return the name of the module.
- const char * **ggzcore_module_get_version** (**GGZModule** *module)
Return the (game?) version of the module.
- const char * **ggzcore_module_get_prot_engine** (**GGZModule** *module)
Return the name of the module's protocol engine.
- const char * **ggzcore_module_get_prot_version** (**GGZModule** *module)
Return the version of the module's protocol engine.
- const char * **ggzcore_module_get_author** (**GGZModule** *module)
Return the author of the module.
- const char * **ggzcore_module_get_frontend** (**GGZModule** *module)
Return the module's frontend type.
- const char * **ggzcore_module_get_url** (**GGZModule** *module)
Return the URL associated with the module.
- const char * **ggzcore_module_get_icon_path** (**GGZModule** *module)
- const char * **ggzcore_module_get_help_path** (**GGZModule** *module)
Return the help path of the module (?).
- char ** **ggzcore_module_get_argv** (**GGZModule** *module)
Return the executable arguments for the module. See `exec()`.
- **GGZModuleEnvironment** **ggzcore_module_get_environment** (**GGZModule** *module)
Return the preferred environment type.
- **GGZGame** * **ggzcore_game_new** (void)
Make a new game object.

- `int ggzcore_game_init (GGZGame *game, GGZServer *server, GGZModule *module)`
Initialize the game object.
- `void ggzcore_game_free (GGZGame *game)`
Free the game object.
- `int ggzcore_game_add_event_hook (GGZGame *game, const GGZGameEvent event, const GGZHookFunc func)`
Register a hook for a game event.
- `int ggzcore_game_add_event_hook_full (GGZGame *game, const GGZGameEvent event, const GGZHookFunc func, const void *data)`
Register a hook for a game event.
- `int ggzcore_game_remove_event_hook (GGZGame *game, const GGZGameEvent event, const GGZHookFunc func)`
Remove a hook from a game event.
- `int ggzcore_game_remove_event_hook_id (GGZGame *game, const GGZGameEvent event, const unsigned int hook_id)`
Remove a specified hook from a game event.
- `int ggzcore_game_get_control_fd (GGZGame *game)`
Return the control (ggzmod) socket for the game.
- `void ggzcore_game_set_server_fd (GGZGame *game, unsigned int fd)`
Return the game's server socket. Needed only for channels set up by ggzcore.
- `GGZModule * ggzcore_game_get_module (GGZGame *game)`
Return the module set for the game.
- `int ggzcore_game_launch (GGZGame *game)`
Launch thee game!
- `int ggzcore_game_read_data (GGZGame *game)`
Read data from the game. When data is pending on the control socket, call this function.

5.1.1 Detailed Description

The interface for the ggzcore library used by GGZ clients.

5.1.2 Typedef Documentation

5.1.2.1 `typedef void(*) GGZDestroyFunc(void *data)`

GGZ object destroy function type.

Todo

This is not currently used.

5.1.2.2 typedef struct _GGZGameType GGZGameType

Contains information about a `_game type_`.

Note:

Each room has one game type; a game may be used in multiple rooms.

5.1.2.3 typedef GGZHookReturn(*) GGZHookFunc(unsigned int id, const void *event_data, const void *user_data)

GGZ Event hook function type, used as a vallback for events

5.1.2.4 typedef struct _GGZOptions GGZOptions

Options structure for ggzcore library

5.1.3 Enumeration Type Documentation**5.1.3.1 enum GGZGameEvent**

A `GGZGameEvent` is an event associated with the game, that is triggered by a communication from the server or from the game. When a game event occurs, the associated event handle will be called, and will be passed the event data (a `void*`) along with the (optional) user data. All game events apply to the current game. Game events are usually triggered by calling `ggzcore_server_read_data` or `ggzcore_game_read_data`.

See also:

`ggzcore_game_add_event_hook` (p. 38)

`ggzcore_server_read_data` (p. 52)

Enumerator:

GGZ_GAME_LAUNCHED A game was launched by the player (you). After this the core client should call `ggzcore_game_get_control_fd`, monitor the socket that function returns, and call `ggzcore_game_read_data` when there is data pending. This event is triggered inside of `ggzcore_game_launch`.

Parameters:

data NULL

See also:

`ggzcore_game_launch` (p. 25)

GGZ_GAME_LAUNCH_FAIL Your game launch has failed. Triggered instead of `GGZ_GAME_LAUNCHED` when there's a failure somewhere.

Parameters:

data NULL

See also:

GGZ_GAME_LAUNCHED (p. 26)

GGZ_GAME_NEGOTIATED Negotiation with server was successful. This should happen some time after the launch succeeds. The core client need do nothing at this point.

Parameters:

data NULL

GGZ_GAME_NEGOTIATE_FAIL Negotiation was not successful, game launch failed.

Todo

Currently this can't actually happen...

GGZ_GAME_PLAYING Game reached the 'playing' state. When this happens the core client should call `ggzcore_room_launch_table` or `ggzcore_room_join_table` to finalize the game join.

Parameters:

data NULL

5.1.3.2 enum GGZHookReturn

GGZ Hook function return types

Enumerator:

GGZ_HOOK_OK Success!

GGZ_HOOK_REMOVE Remove this hook immediately.

GGZ_HOOK_ERROR A localized error.

GGZ_HOOK_CRISIS A major error; stop processing the event.

5.1.3.3 enum GGZLoginType

This controls the type of login a user chooses. A different value will require different information to be sent to the server.

Enumerator:

GGZ_LOGIN Standard login; uname and correct passwd needed.

GGZ_LOGIN_GUEST Guest login; only a uname is required.

GGZ_LOGIN_NEW New user login; only a uname is required. Password will be assigned by the server (but can be passed along).

5.1.3.4 enum GGZModuleEnvironment

The environment a game frontend expects.

Core clients should offer those game modules which fit their own environment.

Enumerator:

GGZ_ENVIRONMENT_PASSIVE No GUI, no interaction with user
GGZ_ENVIRONMENT_CONSOLE Text console
GGZ_ENVIRONMENT_FRAMEBUFFER VESA or framebuffer
GGZ_ENVIRONMENT_XWINDOW X11 windowed mode (default)
GGZ_ENVIRONMENT_XFULLSCREEN X11 fullscreen mode

5.1.3.5 enum GGZOptionFlags**Enumerator:**

GGZ_OPT_MODULES Load the default configuration file (unused).
GGZ_OPT_THREADED_IO Load the game module list.
GGZ_OPT_EMBEDDED Provide multi-threaded network IO (for asynchronous lookups).
GGZ_OPT_RECONNECT Run game with integrated core client. Reconnect automatically after connection loss.

5.1.3.6 enum GGZRoomEvent

A GGZRoomEvent is an event associated with the room, that is triggered by a communication from the server. When a room event occurs, the associated event handler will be called, and will be passed the event data (a void*), along with the (optional) user data. All room events apply to the current room unless a room number is given. Room events are almost always triggered by calling `ggzcore_server_read_data`.

See also:

`ggzcore_room_add_event_hook` (p. 41)
`ggzcore_server_read_data` (p. 52)

Enumerator:

GGZ_PLAYER_LIST The list of players in a room has arrived.

Parameters:

data The room (GGZRoom*)

Note:

This will only be issued for the current room.

See also:

`ggzcore_room_list_players` (p. 44)

GGZ_TABLE_LIST Received the list of active tables.

Parameters:

data NULL

See also:

`ggzcore_room_list_tables` (p. 44)

GGZ_CHAT_EVENT Received a chat message of any kind. This can happen at any time when you're in a room.

Parameters:

data The **GGZChatEventData** (p. 8) associated with the chat.

See also:

GGZChatEventData (p. 8)

GGZ_ROOM_ENTER A player has entered the room with you.

Parameters:

data A **GGZRoomChangeEventData** (p. 11) structure.

GGZ_ROOM_LEAVE A player has left your room.

Parameters:

data A **GGZRoomChangeEventData** (p. 11) structure.

GGZ_TABLE_UPDATE One of the tables in the current room has changed.

Todo

How are you supposed to know which table has changed?

Parameters:

data NULL

GGZ_TABLE_LAUNCHED The table you tried to launch has launched!

See also:

ggzcore_room_launch_table (p. 43)

Parameters:

data NULL

GGZ_TABLE_LAUNCH_FAIL The table you tried to launch couldn't be launched

See also:

GGZ_TABLE_LAUNCHED (p. 29)

Parameters:

data A pointer to a **GGZErrorEventData** (p. 9)

GGZ_TABLE_JOINED Your table join attempt has succeeded.

See also:

ggzcore_room_join_table (p. 43)

Parameters:

data The table index (int*) of the table we joined.

GGZ_TABLE_JOIN_FAIL Joining a table did not succeed.

See also:

GGZ_TABLE_JOINED (p. 29)

Parameters:

data A helpful error string.

GGZ_TABLE_LEFT You have successfully left the table you were at.

See also:

`ggzcore_room_leave_table` (p. 43)

Parameters:

data The `GGZTableLeaveEventData` (p. 12) associated with the leave.

See also:

`GGZTableLeaveEventData` (p. 12)

`GGZ_TABLE_LEAVE_FAIL` Your attempt to leave the table has failed.

See also:

`GGZ_TABLE_LEFT` (p. 29)

Parameters:

data A helpful error string.

`GGZ_PLAYER_LAG` A player's lag (measure of connection speed) has been updated

See also:

`ggzcore_player_get_lag` (p. 21)

Parameters:

data The name of the player whose lag has changed.

`GGZ_PLAYER_STATS` A player's stats have been updated.

See also:

`GGZ_PLAYER_LIST` (p. 28)

`ggzcore_player_get_record` (p. 41)

`ggzcore_player_get_rating` (p. 40)

`ggzcore_player_get_ranking` (p. 40)

`ggzcore_player_get_highscore` (p. 40)

Parameters:

data The name of the player whose stats have changed.

`GGZ_PLAYER_COUNT` The number of players in a room has arrived.

Parameters:

data The room (`GGZRoom*`)

`GGZ_PLAYER_PERMS` A player's permissions have been updated.

See also:

`ggzcore_player_get_perm`

Parameters:

data The name of the player whose stats have changed.

5.1.3.7 enum GGZServerEvent

A `GGZServerEvent` is an event triggered by a communication from the server. Each time an event occurs, the associated event handler will be called, and will be passed the event data (a `void*`). Most events are generated as a result of `ggzcore_server_read_data`.

See also:

`ggzcore_server_add_event_hook` (p. 45)

Enumerator:

GGZ_CONNECTED We have just made a connection to the server. After this point the server's socket should be accessible and should be monitored for data. It happens in direct response to `ggzcore_server_connect`. Note that most events after this will only happen by calling `ggzcore_server_read_data` on the server's FD!

Parameters:

data NULL

See also:

`ggzcore_server_connect` (p. 45)

GGZ_CONNECT_FAIL Error: we have failed to connect to the server. This is generated in place of `GGZ_CONNECTED` if the connection could not be made. The server object is otherwise unaffected.

Parameters:

data An error string (created by `strerror`)

See also:

`ggzcore_server_connect` (p. 45)

GGZ_NEGOTIATED We have negotiated a connection to the server. This will happen automatically once a connection has been established, if the server socket is monitored.

Note:

This just means we've determined `ggzd` is at the other end.

Parameters:

data NULL

See also:

`ggzcore_server_read_data` (p. 52)

GGZ_NEGOTIATE_FAIL Error: negotiation failure. Could be the wrong version. This will happen in place of a `GGZ_NEGOTIATED` if the server could not be negotiated with.

Parameters:

data A useless error string.

See also:

`ggzcore_server_read_data` (p. 52)

GGZ_LOGGED_IN We have successfully logged in. We can now start doing stuff. This will not happen until the client sends their login information.

See also:

`ggzcore_server_login` (p. 51)

Parameters:

data NULL

See also:

`ggzcore_server_read_data` (p. 52)

GGZ_LOGIN_FAIL Error: login failure. This will happen in place of `GGZ_LOGGED_IN` if the login failed. The server object will be otherwise unaffected.

Parameters:

data A pointer to a **GGZErrorEventData** (p. 9).

See also:

GGZErrorEventData (p. 9)

ggzcore_server_read_data (p. 52)

GGZ_MOTD_LOADED The MOTD has been read from the server and can be displayed. The server will send us the MOTD automatically after login; it can also be requested by **ggzcore_server_motd**. It is up to the client whether or not to display it. See the online documentation (somewhere?) about the MOTD markup format.

Parameters:

data Pointer to a **GGZMotdEventData** (p. 10) including the full MOTD text.

See also:

ggzcore_server_motd (p. 18)

Todo

The MOTD cannot be accessed outside of this event

See also:

ggzcore_server_read_data!

GGZ_ROOM_LIST The room list arrived. This will only happen after the list is requested by **ggzcore_server_list_rooms()** (p. 51). The list may be accessed through **ggzcore_server_get_num_rooms()** (p. 48) and **ggzcore_server_get_nth_room()** (p. 17). Until this event arrives these functions will be useless!

Parameters:

data NULL

See also:

ggzcore_server_read_data (p. 52)

GGZ_TYPE_LIST The list of game types is available. This will only happen after the list is requested by **ggzcore_server_list_types()**. The list may be accessed through **ggzcore_server_get_num_gametypes()** (p. 48) and **ggzcore_server_get_nth_gametype()** (p. 18). Until this event arrives these functions will be useless!

Parameters:

data NULL

See also:

ggzcore_server_read_data (p. 52)

GGZ_SERVER_PLAYERS_CHANGED The number of players on the server has changed. This event is issued rather frequently every time players enter or leave.

Parameters:

data NULL

See also:

ggzcore_server_get_num_players (p. 48)

ggzcore_server_read_data (p. 52)

GGZ_ENTERED We have successfully entered a room. This will be issued to tell us a room join has succeeded, after it has been requested.

Parameters:

data NULL

See also:

`ggzcore_server_join_room` (p. 50)
`ggzcore_server_read_data` (p. 52)

GGZ_ENTER_FAIL Error: we have tried to enter a room and failed. This will be issued to tell us a room join has failed.

Parameters:

data A pointer to a **GGZErrorEventData** (p. 9).

See also:

GGZErrorEventData (p. 9)
`ggzcore_server_join_room` (p. 50)
`ggzcore_server_read_data` (p. 52)

GGZ_LOGOUT Logged out of the server. This will happen when the server completes the communication; usually after `ggzcore_net_send_logout` is called.

Parameters:

data NULL

See also:

`ggzcore_server_read_data` (p. 52)

GGZ_NET_ERROR Error: a network (transmission) error occurred. The server will automatically disconnect.

Parameters:

data A generally unhelpful error string.

See also:

`ggzcore_server_read_data` (p. 52)

GGZ_PROTOCOL_ERROR Error: a communication protocol error occurred. This can happen in a variety of situations when the server sends us something we can't handle. The server will be automatically disconnected.

Parameters:

data A pointer to a **GGZErrorEventData** (p. 9).

See also:

GGZErrorEventData (p. 9)
`ggzcore_server_read_data` (p. 52)

GGZ_CHAT_FAIL Error: A chat message could not be sent. This will happen when we try to send a chat and the server rejects it.

Parameters:

data A pointer to a **GGZErrorEventData** (p. 9).

See also:

GGZErrorEventData (p. 9)
`ggzcore_server_read_data` (p. 52)

GGZ_STATE_CHANGE The internal state of `ggzcore` has changed. This may happen at any time.

Parameters:*data* NULL**See also:**

GGZStateID (p. 35)
ggzcore_server_get_state (p. 49)

GGZ_CHANNEL_CONNECTED Status event: a requested direct game connection has been established. To start a game (table), a channel must be created. This event will alert that the channel has been established. The channel's FD should then be monitored for input, which should then be passed back to the server object for handling.

Parameters:*data* NULL**Note:**

This event is deprecated and should not be used.

See also:

ggzcore_server_get_channel (p. 46)
ggzcore_server_read_data (p. 52)

GGZ_CHANNEL_READY Game channel is ready for read/write operations. After the channel has been connected, if we continue to monitor the socket eventually it will be negotiated and ready to use. At this point it is ready for the game client to use.

Parameters:*data* NULL**Note:**

This event is deprecated and should not be used.

See also:

ggzcore_server_read_data (p. 52)

GGZ_CHANNEL_FAIL Error: Failure during setup of direct connection to game server. If the channel could not be prepared, this event will happen instead of **GGZ_CHANNEL_READY** or **GGZ_CHANNEL_CONNECTED** event. At this point the channel is no longer useful (I think).

Parameters:*data* An unhelpful error string**Note:**

This event is deprecated and should not be used.

See also:

ggzcore_server_read_data (p. 52)

GGZ_SERVER_ROOMS_CHANGED The room configuration on the server changed. A room was either added or removed, or scheduled for removing (closed).

Parameters:*data* NULL**See also:**

ggzcore_room_get_closed() (p. 20)

GGZ_NUM_SERVER_EVENTS Terminator. Do not use.

5.1.3.8 enum GGZStateID

The states a server connection may be in.

On the client side, a simplistic state maching is used to tell what's going on. A game client should usually consult the current state when determining what actions are possible.

Enumerator:

- GGZ_STATE_OFFLINE** Not connected (at all)
- GGZ_STATE_CONNECTING** In the process of connecting.
- GGZ_STATE_RECONNECTING** Continuous reconnection attempts.
- GGZ_STATE_ONLINE** Connected, but not doing anything.
- GGZ_STATE_LOGGING_IN** In the process of logging in.
- GGZ_STATE_LOGGED_IN** Online and logged in!
- GGZ_STATE_ENTERING_ROOM** Moving into a room.
- GGZ_STATE_IN_ROOM** Online, logged in, and in a room.
- GGZ_STATE_BETWEEN_ROOMS** Moving between rooms.
- GGZ_STATE_LAUNCHING_TABLE** Trying to launch a table.
- GGZ_STATE_JOINING_TABLE** Trying to join a table.
- GGZ_STATE_AT_TABLE** Online, logged in, in a room, at a table.
- GGZ_STATE_LEAVING_TABLE** Waiting to leave a table.
- GGZ_STATE_LOGGING_OUT** In the process of logging out.

5.1.4 Function Documentation

5.1.4.1 int ggzcore_conf_commit (void)

ggzcore_conf_commit() (p. 35) - Commits the core user config file to disk

Returns:

: int : 0 if successful, -1 on error

5.1.4.2 int ggzcore_conf_initialize (const char * g_path, const char * u_path)

ggzcore_conf_initialize() (p. 35) Opens the global and/or user configuration files for the frontend. Either g_path or u_path can be NULL if the file is not to be used. The user config file will be created if it does not exist.

Returns:

: 0 on success, negative on failure

5.1.4.3 `int ggzcore_conf_read_int (const char * section, const char * key, int def)`

`ggzcore_conf_read_int()` (p. 36) - Read a integer from the configuration file(s)

Parameters:

section section to get value from
key key value was stored under
def default value to return if none is found

Returns:

an integer from the configuration file OR the default value

Note:

There is no guaranteed way to find if the call failed. If you must know, call `ggzcore_conf_read_string` with a NULL default value and check for the NULL return.

5.1.4.4 `int ggzcore_conf_read_list (const char * section, const char * key, int * argcp, char *** argvp)`

`ggzcore_conf_read_list()` (p. 36) - Read a list from the configuration file(s)

Parameters:

section section to get value from
key key value was stored under
argcp ptr to int which will receive the list entry count
argvp a pointer to a dynamically allocated array that `ggzcore_conf_read_list()` (p. 36) will build

Returns:

int : 0 if successful, -1 on error

5.1.4.5 `char* ggzcore_conf_read_string (const char * section, const char * key, const char * def)`

`ggzcore_conf_read_string()` (p. 36) - Read a string from the configuration file(s)

Parameters:

section section to get value from
key key value was stored under
def default value to return if none is found

Returns:

a dynamically allocated string from the configuration file OR a dynamically allocated copy of the default string

Note:

The default may be set to NULL, in which case a NULL will be returned if the value could not be found in either configuration file.

5.1.4.6 int ggzcore_conf_remove_key (const char * *section*, const char * *key*)

ggzcore_conf_remove_key() (p. 37) - Removes a key entry from the user config file

Parameters:

section section to remove

key key entry to remove

Returns:

int : 0 if success, -1 on error, 1 on soft error (section/key didn't exist)

5.1.4.7 int ggzcore_conf_remove_section (const char * *section*)

ggzcore_conf_remove_section() (p. 37) - Removes a section from the user config file

Parameters:

section section to remove

Returns:

int : 0 if successful, -1 on error, 1 on soft error (section did not exist)

5.1.4.8 int ggzcore_conf_write_int (const char * *section*, const char * *key*, int *value*)

ggzcore_conf_write_int() (p. 37) - Write a integer to the user config file

Parameters:

section section to store value in

key key to store value under

value value to store

Returns:

int : 0 if successful, -1 on error

5.1.4.9 int ggzcore_conf_write_list (const char * *section*, const char * *key*, int *argc*, char ** *argv*)

ggzcore_conf_write_list() (p. 37) - Write a list to the user config file

Parameters:

section section to store value in

key key to store value under

argc count of string arguments in list

argv array of NULL terminated strings

Returns:

int : 0 if successful, -1 on error

5.1.4.10 `int ggzcore_conf_write_string (const char * section, const char * key, const char * value)`

`ggzcore_conf_write_string()` (p. 38) - Write a string to the user config file

Parameters:

section section to store value in

key key to store value under

value value to store

Returns:

int : 0 if successful, -1 on error

5.1.4.11 `void ggzcore_destroy (void)`

`ggzcore_destroy()` (p. 38) - Cleanup function for ggzcore lib.

5.1.4.12 `int ggzcore_game_add_event_hook (GGZGame * game, const GGZGameEvent event, const GGZHookFunc func)`

Register a hook for a game event.

See also:

`ggzcore_server_add_event_hook` (p. 45)

`ggzcore_room_add_event_hook` (p. 41)

5.1.4.13 `int ggzcore_game_add_event_hook_full (GGZGame * game, const GGZGameEvent event, const GGZHookFunc func, const void * data)`

Register a hook for a game event.

See also:

`ggzcore_server_add_event_hook_full` (p. 45)

`ggzcore_room_add_event_hook_full` (p. 42)

5.1.4.14 `int ggzcore_game_read_data (GGZGame * game)`

Read data from the game. When data is pending on the control socket, call this function.

Returns:

negative on error

See also:

`ggzcore_game_get_control_fd` (p. 25)

5.1.4.15 `int ggzcore_game_remove_event_hook (GGZGame * game, const GGZGameEvent event, const GGZHookFunc func)`

Remove a hook from a game event.

See also:

`ggzcore_server_remove_event_hook` (p. 52)
`ggzcore_room_remove_event_hook` (p. 44)

5.1.4.16 `int ggzcore_game_remove_event_hook_id (GGZGame * game, const GGZGameEvent event, const unsigned int hook_id)`

Remove a specified hook from a game event.

See also:

`ggzcore_server_remove_event_hook_id` (p. 52)
`ggzcore_room_remove_event_hook_id` (p. 44)

5.1.4.17 `unsigned int ggzcore_gametype_get_id (const GGZGameType * type)`

Get the ID of this gametype.

Note:

This is not normally useful for a GGZ client to know.

5.1.4.18 `int ggzcore_gametype_get_max_bots (const GGZGameType * type)`

Get the maximum number of bots the game can support.

See also:

`ggzcore_gametype_bots_is_valid`

5.1.4.19 `int ggzcore_gametype_get_max_players (const GGZGameType * type)`

Get the maximum number of players the game can support.

See also:

`ggzcore_gametype_num_players_is_valid` (p. 23)

5.1.4.20 int ggzcore_init (GGZOptions options)

ggzcore_init() (p. 40) - Initialization function for ggzcore lib.

Parameters:

options options structure

Returns:

int : 0 if successful, -1 on failure

5.1.4.21 int ggzcore_module_add (const char * name, const char * version, const char * prot_engine, const char * prot_version, const char * author, const char * frontend, const char * url, const char * exe_path, const char * icon_path, const char * help_path, GGZModuleEnvironment environment)

This adds a local module to the list. It returns 0 if successful or -1 on failure.

5.1.4.22 const char* ggzcore_module_get_icon_path (GGZModule * module)

This is (intended to be) an optional xpm file that the module can provide to use for representing the game graphically.

5.1.4.23 int ggzcore_player_get_highscore (const GGZPlayer * player, int * highscore)

Get the player's highscore.

Returns:

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

5.1.4.24 int ggzcore_player_get_ranking (const GGZPlayer * player, int * ranking)

Get the player's ranking.

Returns:

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

5.1.4.25 int ggzcore_player_get_rating (const GGZPlayer * player, int * rating)

Get the player's rating.

Returns:

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

5.1.4.26 `int ggzcore_player_get_record (const GGZPlayer * player, 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.

5.1.4.27 `int ggzcore_player_set_perm (GGZPlayer * player, GGZPerm perm, bool set)`

Administer player permissions.

Parameters:

player Name of the target player

perm The permission to (un)set.

set Whether to set or clear the permission.

Returns:

0 on success, negative on failure

5.1.4.28 `void ggzcore_reload (void)`

`ggzcore_reload()` (p. 41) - Reload game module database

5.1.4.29 `int ggzcore_room_add_event_hook (GGZRoom * room, const GGZRoomEvent event, const GGZHookFunc func)`

Register a handler (hook) for the room event.

A room event will happen when data is received from the server. To make updates to the frontend, the client will need to register a hook function to handle the event. This hook function will be called each time the room event occurs. More than one hook function may be specified, in which case they will all be called (in FIFO order).

Parameters:

room The room object to associate the hook with.

event The event the handler is going to be "hooked" onto.

func The event handler itself. This is called during the event.

Returns:

The hook ID, or negative on error.

See also:

`ggzcore_room_add_event_hook_full` (p. 42)

`ggzcore_room_remove_event_hook` (p. 44)

`ggzcore_room_remove_event_hook_id` (p. 44)

5.1.4.30 `int ggzcore_room_add_event_hook_full (GGZRoom * room, const GGZRoomEvent event, const GGZHookFunc func, const void * data)`

Register a handler (hook) for thee room event, with data.

This function is similar to `ggzcore_room_add_event_hook`, except that user data will be associated with the hook. This data will be passed back to the function each time it is invoked on this event.

Parameters:

- room* The room object to associate the hook with.
- event* The event the handler is going to be "hooked" onto.
- func* The event handler itself. This is called during the event.
- data* The user data associated with the hook.

Returns:

The hook ID, or negative on error.

See also:

`ggzcore_room_add_event_hook` (p. 41)

5.1.4.31 `int ggzcore_room_admin (GGZRoom * room, GGZAdminType type, const char * player, const char * reason)`

Server- and room-level administrative actions.

Parameters:

- room* Your current room.
- type* Type of action (gag, ungag, kick, ...)
- player* Name of the target player
- reason* The reason for the action (only for kicking)

Returns:

0 on success, negative on failure

5.1.4.32 `int ggzcore_room_chat (GGZRoom * room, const GGZChatType opcode, const char * player, const char * msg)`

Chat!

Parameters:

- room* Your current room.
- opcode* The chat type.
- player* The name of the target player (only for certain chat types)
- msg* The text of the chat message (some chat types don't need it)

Returns:

0 on success, negative on (any) failure

Note:

The chat message should be in UTF-8.

5.1.4.33 int ggzcore_room_join_table (GGZRoom * room, const unsigned int table_id, int spectator)

Join a table in the room, so that you can then play at it.

Parameters:

room Your current room.

table_id The table to join.

spectator TRUE if you wish to spectate, FALSE if you want to play

Returns:

0 on success, negative on (any) failure

5.1.4.34 int ggzcore_room_launch_table (GGZRoom * room, GGZTable * table)

Launch a table in the room.

When a player wants to launch a new table, this is the function to do it. You must first create the table and set up the number and type of seats. Then call this function to initiate the launch.

Parameters:

room Your current room.

table The table to launch.

Returns:

0 on success, negative on (any) failure

5.1.4.35 int ggzcore_room_leave_table (GGZRoom * room, int force)

Leave the table you are currently playing at.

This function tries to leave your current table. You should "force" the leave only if the game client is inoperable, since for some games this will destroy the game server as well.

Parameters:

room Your current room.

force TRUE to force the leave, FALSE to leave it up to ggzd

Returns:

0 on success, negative on (any) failure

5.1.4.36 `int ggzcore_room_list_players (GGZRoom * room)`

Call to request a list of players in the room.

See also:

`GGZ_PLAYER_LIST` (p.28)

5.1.4.37 `int ggzcore_room_list_tables (GGZRoom * room, const int type, const char global)`

Call to request a list of tables in the room.

Parameters:

room Your current room

type currently ignored (???)

global currently ignored (???)

See also:

`GGZ_TABLE_LIST` (p. 28)

5.1.4.38 `int ggzcore_room_remove_event_hook (GGZRoom * room, const GGZRoomEvent event, const GGZHookFunc func)`

Remove a hook from an event.

Removes a specific hook function from the hook list for the given room event. If more than one copy of the function exists in the list, the oldest one will be removed.

Parameters:

room The room object to associate the hook with.

event The event the handler is to be unhooked from.

func The event handler function to remove.

Returns:

0 on success, negative on failure.

See also:

`ggzcore_room_add_event_hook` (p. 41)

5.1.4.39 `int ggzcore_room_remove_event_hook_id (GGZRoom * room, const GGZRoomEvent event, const unsigned int hook_id)`

Remove a hook from an event, by ID.

Removes a specific hook from the hook list for the given room. The "ID" should be the same as that returned when the hook was added.

Parameters:

room The room object to associate the hook with.

event The event the handler is to be unhooked from.

id The ID of the hook to remove, as returned by the add function return 0 on success, negative on failure

See also:

`ggzcore_room_add_event_hook` (p. 41)

5.1.4.40 int ggzcore_server_add_event_hook (GGZServer * server, const GGZServerEvent event, const GGZHookFunc func)

Register a callback handler for a server event.

Call this function to register the given GGZHookFunc as a handler for the given event. Then any time that event happens the handler function will be called.

Parameters:

server The GGZ server object.

event The server event to be handled.

func The handler function to be called when the event occurs.

Returns:

A hook ID value to identify this handler.

Note:

Equivalent to `ggzcore_server_add_event_hook_full` with `data==NULL`.
More than one handler can be registered for each event.

5.1.4.41 int ggzcore_server_add_event_hook_full (GGZServer * server, const GGZServerEvent event, const GGZHookFunc func, const void * data)

Register a callback handler for a server event.

See also:

`ggzcore_server_add_event_hook` (p. 45)

Parameters:

data An arbitrary pointer that will be passed to the hook function.

5.1.4.42 int ggzcore_server_connect (GGZServer * server)

Connect to the server.

Call this function to initially connect to a GGZ server. Connection info is set using the `ggzcore_server_set_hostinfo` function.

The function is asynchronous and will return very quickly. After the connection is (hopefully) established we will receive either a GGZ_CONNECTED or GGZ_CONNECT_FAIL server event. If the connection succeeds, negotiations with the GGZ server will begin automatically. Once this is complete, we will receive either a GGZ_NEGOTIATED or GGZ_NEGOTIATE_FAIL event.

Parameters:

server The GGZ server object.

Returns:

0 on success, -1 on failure.

Note:

On success a GGZ_CONNECTED event will be generated.

On failure a GGZ_CONNECT_FAIL event may or may not be generated.

5.1.4.43 int ggzcore_server_create_channel (GGZServer * server)

Establish a direct connection.

Direct connections are requested for games. They are similar to connections, instead of that no login takes place, but a channel for arbitrary game data is created. Needed only for channels set up by ggzcore.

Parameters:

server The GGZ server object.

Returns:

0 on success, -1 on failure.

5.1.4.44 int ggzcore_server_get_channel (GGZServer * server)

Get the socket used for direct game connections.

This returns the file descriptor of the socket for the TCP game connection. This will be handed off to a game module when it is ready. Needed only for channels set up by ggzcore.

Parameters:

server The GGZ server object.

Returns:

The file descriptor of the connection socket.

See also:

[ggzcore_server_create_channel](#) (p. 46)

5.1.4.45 `int ggzcore_server_get_fd (const GGZServer * server)`

Get the socket used for connection with the server.

This returns the file descriptor of the primary socket for the TCP connection to the server. All GGZ data goes across this socket.

Parameters:

server The GGZ server object.

Returns:

The file descriptor of the connection socket.

See also:

`ggzcore_server_connect` (p. 45)

5.1.4.46 `const char* ggzcore_server_get_handle (const GGZServer * server)`

Get the handle being used for this server.

Parameters:

server The GGZ server object.

Returns:

A string containing the handle, or NULL on error.

See also:

`ggzcore_server_set_logininfo` (p. 53)

5.1.4.47 `const char* ggzcore_server_get_host (const GGZServer * server)`

Get the hostname of the server.

Parameters:

server The GGZ server object.

Returns:

A string containing the host name, or NULL on error.

See also:

`ggzcore_server_set_hostinfo` (p. 53)

5.1.4.48 `int ggzcore_server_get_num_gametypes (const GGZServer * server)`

Return the overall number of game types on the server.

Parameters:

server The GGZ server object.

Returns:

The number of game types on this server, or -1 on error.

Note:

This number is 0 until

See also:

`GGZ_TYPE_LIST` (p. 32).

5.1.4.49 `int ggzcore_server_get_num_players (const GGZServer * server)`

Get the total number of players on the server.

Parameters:

server The GGZ server object return An approximate number of players in all rooms of the server.

5.1.4.50 `int ggzcore_server_get_num_rooms (const GGZServer * server)`

Return the number of rooms on the server, or -1 on error.

Note:

Until we retrieve the list of rooms, this will return 0.

5.1.4.51 `const char* ggzcore_server_get_password (const GGZServer * server)`

Get the password being used for this server.

Parameters:

server The GGZ server object.

Returns:

A string containing the password, or NULL on error.

See also:

`ggzcore_server_set_logininfo` (p. 53)

5.1.4.52 GGZPlayer* ggzcore_server_get_player (GGZServer * *server*, const char * *name*)

Find the player, by name (or NULL).

Note:

Only players in the current room can currently be found.
This function is inefficient.

5.1.4.53 int ggzcore_server_get_port (const GGZServer * *server*)

Get the port of the server.

Parameters:

server The GGZ server object.

Returns:

The port number of the server, or -1 on error.

See also:

ggzcore_server_set_hostinfo (p. 53)

5.1.4.54 int ggzcore_server_get_room_num (const GGZServer * *server*, const GGZRoom * *room*)

Return the number (position in the room list) of the room.

See also:

ggzcore_server_get_nth_room (p. 17)

5.1.4.55 GGZStateID ggzcore_server_get_state (const GGZServer * *server*)

Get the state of the server connection.

Parameters:

server The GGZ server object.

Returns:

The state of the connection, or -1 on error.

5.1.4.56 `int ggzcore_server_get_tls (const GGZServer * server)`

Get the current TLS status of this server.

Parameters:

server The GGZ server object

Returns:

Whether TLS is active or not

5.1.4.57 `GGZLoginType ggzcore_server_get_type (const GGZServer * server)`

Get the login type being used for this server.

Parameters:

server The GGZ server object.

Returns:

The login type set for the server, or -1 on error.

See also:

`ggzcore_server_set_logininfo` (p. 53)

5.1.4.58 `int ggzcore_server_join_room (GGZServer * server, GGZRoom * room)`

Join a room on the server.

Parameters:

server The GGZ server object.

room The number of the room to join.

Returns:

0 on success, -1 on failure (e.g. non-existing room number).

5.1.4.59 `int ggzcore_server_list_gametypes (GGZServer * server, const char
verbose)`

Request game type list.

Parameters:

server The GGZ server object.

verbose Receive detailed gametype information or not.

Returns:

0 on success, -1 on failure.

Note:

A `GGZ_TYPE_LIST` event will be the asynchronous response on success.

5.1.4.60 `int ggzcore_server_list_rooms (GGZServer * server, const int type, const char verbose)`

Request room list.

Parameters:

server The GGZ server object.

type Not used yet.

verbose Receive all information about a room or only the essentials.

Returns:

0 on success, -1 on failure.

Note:

A GGZ_ROOM_LIST might be generated thereafter.

5.1.4.61 `int ggzcore_server_log_session (GGZServer * server, const char * filename)`

Initiate logging of ggzcore events.

Normally, ggzcore traffic is not logged anywhere. With this functions, such output can be directed into a file. It contains all the network messages received from the server.

Parameters:

server The GGZ server object.

filename The file the messages are written to.

Returns:

0 on success, -1 on error.

5.1.4.62 `int ggzcore_server_login (GGZServer * server)`

Log in to the server.

Call this function to log in to the server once a connection has been established. Typically you must first connect to the server, then wait to receive the GGZ_CONNECTED and GGZ_NEGOTIATED events before attempting to log in. Login info is set using the ggzcore_server_set_logininfo function.

The function is asynchronous and will return immediately. After the login request is sent, we will wait to receive either a GGZ_LOGGED_IN or GGZ_LOGIN_FAIL server event.

Parameters:

server The GGZ server object.

Returns:

0 on success, -1 on failure.

Note:

On failure no events will be generated.

5.1.4.63 GGZServer* ggzcore_server_new (void)

Create a new server object.

Call this function to create a server object. This object holds all state data for communicating with a ggz server. It is necessary for any kind of connection.

5.1.4.64 int ggzcore_server_read_data (GGZServer * server, int fd)

Read data for the server on the specified FD.

Returns:

negative on error

5.1.4.65 int ggzcore_server_remove_event_hook (GGZServer * server, const GGZServerEvent event, const GGZHookFunc func)

Remove a single hook function from an event's hook list.

Parameters:

server The GGZ server object.

event The server event the hook is associated with.

func The function to be removed from the hook list.

Returns:

0 on success (hook removed); -1 on failure (no hook removed)

Note:

At most one copy of the function will be removed.

See also:

`ggzcore_server_remove_event_hook_id` (p. 52)

5.1.4.66 int ggzcore_server_remove_event_hook_id (GGZServer * server, const GGZServerEvent event, const unsigned int hook_id)

Remove a hook function with given ID from the event's hook list.

Parameters:

server The GGZ server object.

event The server event the hook is associated with.

hook_id The ID of the hook event.

Returns:

0 on success (hook removed); -1 on failure (no hook removed)

Note:

The hook ID is given by `ggzcore_server_add_event_hook`

5.1.4.67 int ggzcore_server_reset (GGZServer * server)

Reset the server object.

After you've disconnected, call this function to discard all state data and reset the state of the server object. You can then connect again.

Note:

You should disconnect before resetting.

5.1.4.68 int ggzcore_server_set_hostinfo (GGZServer * server, const char * host, const unsigned int port, const unsigned int use_tls)

Set host info for connecting to the server.

Call this function to set host info for the GGZ server before trying to connect to it.

Parameters:

server The GGZ server object.

host A string containing the hostname.

port The port to connect to.

use_tls If set, the connection will be encrypted.

Returns:

0 on success, -1 on error.

Note:

Should never fail when given valid input.

See also:

`ggzcore_server_connect` (p. 45)

5.1.4.69 int ggzcore_server_set_logininfo (GGZServer * server, const GGZLoginType type, const char * handle, const char * password, const char * email)

Set login info for logging in to the server.

Call this function to set login info for the GGZ server before trying to login. This should only be called when the server is in certain states.

Parameters:

server The GGZ server object.

type The type of login to attempt.

handle The username to use with the server.

password The password to use (may be NULL with some login types).

email The email address to use (may be NULL with some login types).

Returns:

0 on success, -1 on error.

5.1.4.70 `int ggzcore_table_get_seat_count (const GGZTable * table, GGZSeatType type)`

Count the seats of the given type.

Given a table and a seat type, this function returns the number of seats at the table that match the type.

Parameters:

table A GGZ table.

type A GGZSeatType.

Returns:

The number of seats matching the type, or -1 on error.

5.1.4.71 `int ggzcore_table_init (GGZTable * table, const GGZGameType * gametype, const char * desc, const unsigned int num_seats)`

Set data on a table object.

Note:

Useful when launching a game.

5.1.4.72 `GGZTable* ggzcore_table_new (void)`

Create a new table object.

Note:

Useful when launching a game.

5.1.4.73 `int ggzcore_table_set_seat (GGZTable * table, const unsigned int seat, GGZSeatType type, const char * name)`

Set a seat type at a table, pre-launch.

When launching a table, call this function to set up a particular seat at the table. It can also be used to fiddle with already existing tables, but that would be extremely unwise.

Parameters:

table The table object to change.

seat The seat number at the table to change.

type The type of seat to make it (open, reserved, or bot).

name The name of the seat (must be valid for reserved seats).

Returns:

0 on success, -1 on error.

Todo

How do we stop the GGZ client from fiddling with random tables?

Chapter 6

GGZCore Page Documentation

6.1 Todo List

Global GGZ_GAME_NEGOTIATE_FAIL (p. 27) Currently this can't actually happen...

Global GGZ_TABLE_UPDATE (p. 29) How are you supposed to know which table has changed?

Parameters:

data NULL

Global GGZ_MOTD_LOADED (p. 32) The MOTD cannot be accessed outside of this event

See also:

ggzcore_server_read_data!

Global GGZDestroyFunc (p. 25) This is not currently used.

Global ggzcore_table_set_seat (p. 55) How do we stop the GGZ client from fiddling with random tables?

GGZ_PLAYER_PERMS
ggzcore.h, 30

GGZ_PLAYER_STATS
ggzcore.h, 30

GGZ_PROTOCOL_ERROR
ggzcore.h, 33

GGZ_ROOM_ENTER
ggzcore.h, 29

GGZ_ROOM_LEAVE
ggzcore.h, 29

GGZ_ROOM_LIST
ggzcore.h, 32

GGZ_SERVER_PLAYERS_CHANGED
ggzcore.h, 32

GGZ_SERVER_ROOMS_CHANGED
ggzcore.h, 34

GGZ_STATE_AT_TABLE
ggzcore.h, 35

GGZ_STATE_BETWEEN_ROOMS
ggzcore.h, 35

GGZ_STATE_CHANGE
ggzcore.h, 33

GGZ_STATE_CONNECTING
ggzcore.h, 35

GGZ_STATE_ENTERING_ROOM
ggzcore.h, 35

GGZ_STATE_IN_ROOM
ggzcore.h, 35

GGZ_STATE_JOINING_TABLE
ggzcore.h, 35

GGZ_STATE_LAUNCHING_TABLE
ggzcore.h, 35

GGZ_STATE_LEAVING_TABLE
ggzcore.h, 35

GGZ_STATE_LOGGED_IN
ggzcore.h, 35

GGZ_STATE_LOGGING_IN
ggzcore.h, 35

GGZ_STATE_LOGGING_OUT
ggzcore.h, 35

GGZ_STATE_OFFLINE
ggzcore.h, 35

GGZ_STATE_ONLINE
ggzcore.h, 35

GGZ_STATE_RECONNECTING
ggzcore.h, 35

GGZ_TABLE_JOIN_FAIL
ggzcore.h, 29

GGZ_TABLE_JOINED
ggzcore.h, 29

GGZ_TABLE_LAUNCH_FAIL
ggzcore.h, 29

GGZ_TABLE_LAUNCHED
ggzcore.h, 29

GGZ_TABLE_LEAVE_FAIL
ggzcore.h, 30

GGZ_TABLE_LEFT
ggzcore.h, 29

GGZ_TABLE_LIST
ggzcore.h, 28

GGZ_TABLE_UPDATE
ggzcore.h, 29

GGZ_TYPE_LIST
ggzcore.h, 32

GGZChatEventData, 8

GGZChatEventData
message, 8
sender, 8
type, 8
ggzcore.h, 13

GGZ_CHANNEL_CONNECTED, 34

GGZ_CHANNEL_FAIL, 34

GGZ_CHANNEL_READY, 34

GGZ_CHAT_EVENT, 28

GGZ_CHAT_FAIL, 33

GGZ_CONNECT_FAIL, 31

GGZ_CONNECTED, 31

GGZ_ENTER_FAIL, 33

GGZ_ENTERED, 32

GGZ_ENVIRONMENT_CONSOLE, 28

GGZ_ENVIRONMENT_-
FRAMEBUFFER, 28

GGZ_ENVIRONMENT_PASSIVE, 28

GGZ_ENVIRONMENT_-
XFULLSCREEN, 28

GGZ_ENVIRONMENT_XWINDOW, 28

GGZ_GAME_LAUNCH_FAIL, 26

GGZ_GAME_LAUNCHED, 26

GGZ_GAME_NEGOTIATE_FAIL, 27

GGZ_GAME_NEGOTIATED, 27

GGZ_GAME_PLAYING, 27

GGZ_HOOK_CRISIS, 27

GGZ_HOOK_ERROR, 27

GGZ_HOOK_OK, 27

GGZ_HOOK_REMOVE, 27

GGZ_LOGGED_IN, 31

GGZ_LOGIN, 27

GGZ_LOGIN_FAIL, 31

GGZ_LOGIN_GUEST, 27

GGZ_LOGIN_NEW, 27

GGZ_LOGOUT, 33

GGZ_MOTD_LOADED, 32

GGZ_NEGOTIATE_FAIL, 31

GGZ_NEGOTIATED, 31

GGZ_NET_ERROR, 33

GGZ_NUM_SERVER_EVENTS, 34

GGZ_OPT_EMBEDDED, 28

GGZ_OPT_MODULES, 28

- GGZ_OPT_RECONNECT, 28
- GGZ_OPT_THREADED_IO, 28
- GGZ_PLAYER_COUNT, 30
- GGZ_PLAYER_LAG, 30
- GGZ_PLAYER_LIST, 28
- GGZ_PLAYER_PERMS, 30
- GGZ_PLAYER_STATS, 30
- GGZ_PROTOCOL_ERROR, 33
- GGZ_ROOM_ENTER, 29
- GGZ_ROOM_LEAVE, 29
- GGZ_ROOM_LIST, 32
- GGZ_SERVER_PLAYERS_CHANGED, 32
- GGZ_SERVER_ROOMS_CHANGED, 34
- GGZ_STATE_AT_TABLE, 35
- GGZ_STATE_BETWEEN_ROOMS, 35
- GGZ_STATE_CHANGE, 33
- GGZ_STATE_CONNECTING, 35
- GGZ_STATE_ENTERING_ROOM, 35
- GGZ_STATE_IN_ROOM, 35
- GGZ_STATE_JOINING_TABLE, 35
- GGZ_STATE_LAUNCHING_TABLE, 35
- GGZ_STATE_LEAVING_TABLE, 35
- GGZ_STATE_LOGGED_IN, 35
- GGZ_STATE_LOGGING_IN, 35
- GGZ_STATE_LOGGING_OUT, 35
- GGZ_STATE_OFFLINE, 35
- GGZ_STATE_ONLINE, 35
- GGZ_STATE_RECONNECTING, 35
- GGZ_TABLE_JOIN_FAIL, 29
- GGZ_TABLE_JOINED, 29
- GGZ_TABLE_LAUNCH_FAIL, 29
- GGZ_TABLE_LAUNCHED, 29
- GGZ_TABLE_LEAVE_FAIL, 30
- GGZ_TABLE_LEFT, 29
- GGZ_TABLE_LIST, 28
- GGZ_TABLE_UPDATE, 29
- GGZ_TYPE_LIST, 32
- ggzcore_conf_commit, 35
- ggzcore_conf_initialize, 35
- ggzcore_conf_read_int, 35
- ggzcore_conf_read_list, 36
- ggzcore_conf_read_string, 36
- ggzcore_conf_remove_key, 36
- ggzcore_conf_remove_section, 37
- ggzcore_conf_write_int, 37
- ggzcore_conf_write_list, 37
- ggzcore_conf_write_string, 37
- ggzcore_destroy, 38
- ggzcore_game_add_event_hook, 38
- ggzcore_game_add_event_hook_full, 38
- ggzcore_game_read_data, 38
- ggzcore_game_remove_event_hook, 38
- ggzcore_game_remove_event_hook_id, 39
- ggzcore_gametype_get_id, 39
- ggzcore_gametype_get_max_bots, 39
- ggzcore_gametype_get_max_players, 39
- ggzcore_init, 39
- ggzcore_module_add, 40
- ggzcore_module_get_icon_path, 40
- ggzcore_player_get_highscore, 40
- ggzcore_player_get_ranking, 40
- ggzcore_player_get_rating, 40
- ggzcore_player_get_record, 40
- ggzcore_player_set_perm, 41
- ggzcore_reload, 41
- ggzcore_room_add_event_hook, 41
- ggzcore_room_add_event_hook_full, 41
- ggzcore_room_admin, 42
- ggzcore_room_chat, 42
- ggzcore_room_join_table, 43
- ggzcore_room_launch_table, 43
- ggzcore_room_leave_table, 43
- ggzcore_room_list_players, 43
- ggzcore_room_list_tables, 44
- ggzcore_room_remove_event_hook, 44
- ggzcore_room_remove_event_hook_id, 44
- ggzcore_server_add_event_hook, 45
- ggzcore_server_add_event_hook_full, 45
- ggzcore_server_connect, 45
- ggzcore_server_create_channel, 46
- ggzcore_server_get_channel, 46
- ggzcore_server_get_fd, 46
- ggzcore_server_get_handle, 47
- ggzcore_server_get_host, 47
- ggzcore_server_get_num_gametypes, 47
- ggzcore_server_get_num_players, 48
- ggzcore_server_get_num_rooms, 48
- ggzcore_server_get_password, 48
- ggzcore_server_get_player, 48
- ggzcore_server_get_port, 49
- ggzcore_server_get_room_num, 49
- ggzcore_server_get_state, 49
- ggzcore_server_get_tls, 49
- ggzcore_server_get_type, 50
- ggzcore_server_join_room, 50
- ggzcore_server_list_gametypes, 50
- ggzcore_server_list_rooms, 50
- ggzcore_server_log_session, 51
- ggzcore_server_login, 51
- ggzcore_server_new, 52
- ggzcore_server_read_data, 52
- ggzcore_server_remove_event_hook, 52

- ggzcore_server_remove_event_hook_id,
52
- ggzcore_server_reset, 53
- ggzcore_server_set_hostinfo, 53
- ggzcore_server_set_logininfo, 53
- ggzcore_table_get_seat_count, 54
- ggzcore_table_init, 54
- ggzcore_table_new, 54
- ggzcore_table_set_seat, 54
- GGZDestroyFunc, 25
- GGZGameEvent, 26
- GGZGameType, 26
- GGZHookFunc, 26
- GGZHookReturn, 27
- GGZLoginType, 27
- GGZModuleEnvironment, 27
- GGZOptionFlags, 28
- GGZOptions, 26
- GGZRoomEvent, 28
- GGZServerEvent, 30
- GGZStateID, 34
- ggzcore_conf_commit
ggzcore.h, 35
- ggzcore_conf_initialize
ggzcore.h, 35
- ggzcore_conf_read_int
ggzcore.h, 35
- ggzcore_conf_read_list
ggzcore.h, 36
- ggzcore_conf_read_string
ggzcore.h, 36
- ggzcore_conf_remove_key
ggzcore.h, 36
- ggzcore_conf_remove_section
ggzcore.h, 37
- ggzcore_conf_write_int
ggzcore.h, 37
- ggzcore_conf_write_list
ggzcore.h, 37
- ggzcore_conf_write_string
ggzcore.h, 37
- ggzcore_destroy
ggzcore.h, 38
- ggzcore_game_add_event_hook
ggzcore.h, 38
- ggzcore_game_add_event_hook_full
ggzcore.h, 38
- ggzcore_game_read_data
ggzcore.h, 38
- ggzcore_game_remove_event_hook
ggzcore.h, 38
- ggzcore_game_remove_event_hook_id
ggzcore.h, 39
- ggzcore_gametype_get_id
ggzcore.h, 39
- ggzcore_gametype_get_max_bots
ggzcore.h, 39
- ggzcore_gametype_get_max_players
ggzcore.h, 39
- ggzcore_init
ggzcore.h, 39
- ggzcore_module_add
ggzcore.h, 40
- ggzcore_module_get_icon_path
ggzcore.h, 40
- ggzcore_player_get_highscore
ggzcore.h, 40
- ggzcore_player_get_ranking
ggzcore.h, 40
- ggzcore_player_get_rating
ggzcore.h, 40
- ggzcore_player_get_record
ggzcore.h, 40
- ggzcore_player_set_perm
ggzcore.h, 41
- ggzcore_reload
ggzcore.h, 41
- ggzcore_room_add_event_hook
ggzcore.h, 41
- ggzcore_room_add_event_hook_full
ggzcore.h, 41
- ggzcore_room_admin
ggzcore.h, 42
- ggzcore_room_chat
ggzcore.h, 42
- ggzcore_room_join_table
ggzcore.h, 43
- ggzcore_room_launch_table
ggzcore.h, 43
- ggzcore_room_leave_table
ggzcore.h, 43
- ggzcore_room_list_players
ggzcore.h, 43
- ggzcore_room_list_tables
ggzcore.h, 44
- ggzcore_room_remove_event_hook
ggzcore.h, 44
- ggzcore_room_remove_event_hook_id
ggzcore.h, 44
- ggzcore_server_add_event_hook
ggzcore.h, 45
- ggzcore_server_add_event_hook_full
ggzcore.h, 45
- ggzcore_server_connect
ggzcore.h, 45
- ggzcore_server_create_channel
ggzcore.h, 46
- ggzcore_server_get_channel

- ggzcore.h, 46
- ggzcore_server_get_fd
 - ggzcore.h, 46
- ggzcore_server_get_handle
 - ggzcore.h, 47
- ggzcore_server_get_host
 - ggzcore.h, 47
- ggzcore_server_get_num_gametypes
 - ggzcore.h, 47
- ggzcore_server_get_num_players
 - ggzcore.h, 48
- ggzcore_server_get_num_rooms
 - ggzcore.h, 48
- ggzcore_server_get_password
 - ggzcore.h, 48
- ggzcore_server_get_player
 - ggzcore.h, 48
- ggzcore_server_get_port
 - ggzcore.h, 49
- ggzcore_server_get_room_num
 - ggzcore.h, 49
- ggzcore_server_get_state
 - ggzcore.h, 49
- ggzcore_server_get_tls
 - ggzcore.h, 49
- ggzcore_server_get_type
 - ggzcore.h, 50
- ggzcore_server_join_room
 - ggzcore.h, 50
- ggzcore_server_list_gametypes
 - ggzcore.h, 50
- ggzcore_server_list_rooms
 - ggzcore.h, 50
- ggzcore_server_log_session
 - ggzcore.h, 51
- ggzcore_server_login
 - ggzcore.h, 51
- ggzcore_server_new
 - ggzcore.h, 52
- ggzcore_server_read_data
 - ggzcore.h, 52
- ggzcore_server_remove_event_hook
 - ggzcore.h, 52
- ggzcore_server_remove_event_hook_id
 - ggzcore.h, 52
- ggzcore_server_reset
 - ggzcore.h, 53
- ggzcore_server_set_hostinfo
 - ggzcore.h, 53
- ggzcore_server_set_logininfo
 - ggzcore.h, 53
- ggzcore_table_get_seat_count
 - ggzcore.h, 54
- ggzcore_table_init
 - ggzcore.h, 54
- ggzcore_table_new
 - ggzcore.h, 54
- ggzcore_table_set_seat
 - ggzcore.h, 54
- GGZDestroyFunc
 - ggzcore.h, 25
- GGZErrorEventData, 9
- GGZErrorEventData
 - status, 9
- GGZGameEvent
 - ggzcore.h, 26
- GGZGameType
 - ggzcore.h, 26
- GGZHookFunc
 - ggzcore.h, 26
- GGZHookReturn
 - ggzcore.h, 27
- GGZLoginType
 - ggzcore.h, 27
- GGZModuleEnvironment
 - ggzcore.h, 27
- GGZMotdEventData, 10
- GGZMotdEventData
 - motd, 10
 - url, 10
- GGZOptionFlags
 - ggzcore.h, 28
- GGZOptions
 - ggzcore.h, 26
- GGZRoomChangeEventData, 11
- GGZRoomChangeEventData
 - from_room, 11
 - to_room, 11
- GGZRoomEvent
 - ggzcore.h, 28
- GGZServerEvent
 - ggzcore.h, 30
- GGZStateID
 - ggzcore.h, 34
- GGZTableLeaveEventData, 12
- GGZTableLeaveEventData
 - player, 12
- message
 - GGZChatEventData, 8
- motd
 - GGZMotdEventData, 10
- player
 - GGZTableLeaveEventData, 12
- sender
 - GGZChatEventData, 8

status
 GGZErrorEventData, 9

to_room
 GGZRoomChangeEventData, 11

type
 GGZChatEventData, 8

url
 GGZMotdEventData, 10