

# **GGZ Gaming Zone Client/Game Client Protocol Specification**

**The GGZ Gaming Zone developers**

**`ggz-dev@mail.ggzgamingzone.org`**

## **GGZ Gaming Zone Client/Game Client Protocol Specification**

by The GGZ Gaming Zone developers

Copyright © 2005 The GGZ Gaming Zone developers

Module protocol specification for GGZ Gaming Zone game clients. This document covers the communication between the GGZ core clients and the game client modules.

### Revision History

Revision \$Revision: 7521 \$ \$Date: 2005-09-16 23:43:09 +0200 (Fr, 16 Sep 2005) \$

# Table of Contents

Objectives .....	iv
<b>1. The Protocol .....</b>	<b>1</b>
1.1. Startup .....	1
1.2. Connected phase .....	1
1.3. Pregame phase.....	1
1.4. Playing phase .....	1
1.5. Done phase .....	2
<b>A. Protocol Reference .....</b>	<b>3</b>
A.1. Messages from the core client to game client .....	3
GAME_LAUNCH .....	3
GAME_SERVER .....	3
GAME_SERVER_FD .....	4
GAME_PLAYER .....	5
GAME_SEAT .....	6
GAME_SPECTATOR_SEAT .....	6
GAME_CHAT .....	7
GAME_STATS .....	8
GAME_INFO .....	9
A.2. Messages from game client to core client .....	10
GAME_STATE.....	10
STAND .....	10
SIT .....	11
BOOT .....	12
BOT .....	12
OPEN.....	13
CHAT.....	14
INFO.....	14
A.3. Symbolic identifiers and their values .....	15
ControlToTable .....	15
TableToControl .....	16
GGZSeatType .....	17
GGZModState .....	17

# Objectives

A game client, when launched by a GGZ Gaming Zone core client such as kggz or ggz-gtk, will hold a connection to its launcher which is used to get the initial table layout, send out table chat, request seat changes and read a player's game records. This protocol is called the Client/Game Client Protocol, and is available in a reference implementation named libggzmod, written in the C programming language, and its wrappers for C++ and Python.

# Chapter 1. The Protocol

Communication between client and game client happens by means of binary tokens (opcodes), which are of type integer, followed by zero or more opcode-specific variables which can be of type integer, character, or string.

At each point in time, a game client happens to be in a specific state. Messages received from the core client may lead to state changes, as may some explicit transitions being executed by the game client itself. A list of all states can be found in the appendix of states.

Several actions refer to seats on the table the game is being played on. Each seat can be either empty or have an assignment. A full list can be found in the appendix of seat assignments.

Interactions are presented here categorically. For a complete reference of game client/core client interactions, please see the appendix of messages.

## 1.1. Startup

Each game client starts up in `CREATED` mode. In this step there is no information about seats or players yet. The only useful action is to wait for a `GAME_LAUNCH` message so the transition to the `CONNECTED` state can happen. Games can query whether they run on GGZ or not via the environment variable `GGZMODE`.

In order to carry out the transition to `CONNECTED`, the game client must connect to the core client. It does so by reading out the value of the environment variable `GGZSOCKET`, and then either uses this value as file descriptor, or (if no sockets can be passed directly) as local port to connect to. In the former case, `GGZSOCKET` defaults to the value 103.

## 1.2. Connected phase

When in `CONNECTED` state, a game client still isn't operable. There is another mandatory transition it has to do, namely to the `WAITING` state. This happens once the core client tells it where to connect to with a `GAME_SERVER` message, and that connections could be established without problems. Alternatively, depending on the operating system, the core client will establish the connection to the game server and pass it to the game client with a `GAME_SERVER_FD` message, so it can be used from the start on.

## 1.3. Pregame phase

Now that the game client is in `WAITING` state, it is fully operational and connected to the game server. The game server will receive player join events, until the game can start. This transition leads to the `PLAYING` state.

## **1.4. Playing phase**

The PLAYING state is fully under the control of the game client. Messages to request seat changes might be sent from it to the core client, as might chat messages which are then sent to all other players on the table. Likewise, more information about the other players can be requested with an INFO request, upon a GAME\_INFO message informs about those details. Not tied to a specific request but rather sent implicitly is the GAME\_STATS message which contains statistical information about all registered players.

In case of a leaving player, the game client can change the state back to WAITING, and then forward to PLAYING again at any time. Once the game is finished, the (last) transition is done and leads to the DONE state.

## **1.5. Done phase**

Once a game has reached the DONE state, there's no way back anymore. It will be destroyed and the corresponding table removed.

# Appendix A. Protocol Reference

## A.1. Messages from the core client to game client

### GAME\_LAUNCH

#### Name

GAME\_LAUNCH — Initializes the game client

#### Synopsis

GAME_LAUNCH ...		
Data	Type	Example
Opcode	ControlToTable	GAME_LAUNCH

#### Description

This message is always the first one sent to the game client in order to initialize it. It will cause the transition from CREATED to CONNECTED state.

#### Message Data

None

#### Usage

This message only appears during the CREATED state.

### GAME\_SERVER

#### Name

GAME\_SERVER — Tells the game where to connect to

## Synopsis

GAME_SERVER ...		
Data	Type	Example
Opcode	ControlToTable	GAME_SERVER
Hostname	string	live.ggzgamingzone.org
Port number	integer	5688
Player handle	string	player42

## Description

Tells the game where to connect to. This happens when the core client does not establish the connection (channel) to the game server first, as it would send a GAME\_SERVER\_FD in this case. The message causes a transition from CONNECTED to WAITING state.

## Message Data

None

## Usage

This message only appears during the CONNECTED state.

# GAME\_SERVER\_FD

## Name

GAME\_SERVER\_FD — Informs the game client about its connection to the game server

## Synopsis

GAME_SERVER_FD ...		
Data	Type	Example
Opcode	ControlToTable	GAME_SERVER_FD
File descriptor	integer	3



## Description

Depending on the operating system support, core clients can establish the connection to the game servers and pass this connection to the game clients once they're started and in the `CONNECTED` phase. Otherwise, `GAME_SERVER` will be sent to let the game client establish the connection. The message causes a transition from `CONNECTED` to `WAITING` state.

## Message Data

None

## Usage

This message only appears during the `CONNECTED` state.

# GAME\_PLAYER

## Name

`GAME_PLAYER` — Assigns the player's seat on the table

## Synopsis

GAME_PLAYER ...		
Data	Type	Example
Opcode	ControlToTable	GAME_PLAYER
Player name	string	player42
Is player spectator?	integer/boolean	0
Seat number	integer	1

## Description

Assigns the player's seat on the table. While usually `GAME_SEAT` and `GAME_SPECTATOR_SEAT` messages inform about seat assignments, this special message informs about the player's own seat so that the game can already arrange its settings early on.

**Message Data**

None

**Usage**

This message only appears during the WAITING state.

## GAME\_SEAT

**Name**

GAME\_SEAT — Assigns one single seat on the table

**Synopsis**

GAME_SEAT ...		
Data	Type	Example
Opcode	ControlToTable	GAME_SEAT
Seat number	integer	2
Seat type	GGZSeatType	0
Player name	string	player99

**Description**

A single seat assignment or a change thereof is communicated by this message.

**Message Data**

None

**Usage**

This message appears during the WAITING or PLAYING state.

# GAME\_SPECTATOR\_SEAT

## Name

GAME\_SPECTATOR\_SEAT — Assigns one single spectator seat on the table

## Synopsis

GAME_SPECTATOR_SEAT ...		
Data	Type	Example
Opcode	ControlToTable	GAME_SPECTATOR_SEAT
Spectator seat number	integer	3
Spectator name	string	spectator32

## Description

A single spectator seat assignment or a change thereof is communicated by this message. The only difference to GAME\_SEAT is that no seat type is sent.

## Message Data

None

## Usage

This message appears during the WAITING or PLAYING state.

# GAME\_CHAT

## Name

GAME\_CHAT — Message from a player on the table

## Synopsis

GAME_CHAT ...		
Data	Type	Example

Opcode	ControlToTable	GAME_CHAT
Player name	string	player99
Message	string	hello how are you?

## Description

Message from a player on the table

## Message Data

None

## Usage

This message appears during the WAITING or PLAYING state.

# GAME\_STATS

## Name

GAME\_STATS — Statistics about a player

## Synopsis

GAME_STATS ...		
Data	Type	Example
Opcode	ControlToTable	GAME_STATS
ENTRYTBL not supported.		

## Description

Statistics about a player

## Message Data

None

## Usage

This message appears during the WAITING or PLAYING state.

# GAME\_INFO

## Name

GAME\_INFO — Information about a seat

## Synopsis

GAME_INFO ...		
Data	Type	Example
Opcode	ControlToTable	GAME_INFO
Number of seats	integer	1
ENTRYTBL not supported.		

## Description

Detail information about a seat or about all seats is available. For registered players, photo URL and realname might be known to the server and are reported here. The hostname can be reported for all players and spectators, but in most cases will require their agreement to publish this data.

## Message Data

None

## Usage

This message appears during the WAITING or PLAYING state.

## A.2. Messages from game client to core client

### GAME\_STATE

#### Name

GAME\_STATE — Indicate state change

#### Synopsis

GAME_STATE ...		
Data	Type	Example
Opcode	TableToControl	GAME_STATE
State	GGZModState	STATE_WAITING

#### Description

Indicate state change

#### Message Data

None

#### Usage

This message appears during the WAITING or PLAYING state.

### STAND

#### Name

STAND — Request to stand up from table and become spectator

#### Synopsis

STAND ...
-----------

Data	Type	Example
Opcode	TableToControl	STAND

## Description

Request to stand up from table and become spectator

## Message Data

None

## Usage

This message appears during the WAITING or PLAYING state.

# SIT

## Name

SIT — Request to sit down and become a player again

## Synopsis

SIT ...		
Data	Type	Example
Opcode	TableToControl	SIT
Seat number	integer	2

## Description

Request to sit down and become a player again

## Message Data

None

## Usage

This message appears during the WAITING or PLAYING state.

# BOOT

## Name

BOOT — Boot a player from the table

## Synopsis

BOOT ...		
Data	Type	Example
Opcode	TableToControl	BOOT
Player name	string	player56

## Description

Boot a player from the table

## Message Data

None

## Usage

This message appears during the WAITING or PLAYING state.

# BOT

## Name

BOT — Request for a bot joining the table



## Synopsis

BOT ...		
Data	Type	Example
Opcode	TableToControl	BOT
Seat number	integer	2

## Description

Request for a bot joining the table

## Message Data

None

## Usage

This message appears during the WAITING or PLAYING state.

# OPEN

## Name

OPEN — Open up a previously reserved seat

## Synopsis

OPEN ...		
Data	Type	Example
Opcode	TableToControl	OPEN
Seat number	integer	2

## Description

Open up a previously reserved seat

## Message Data

None

## Usage

This message appears during the WAITING or PLAYING state.

# CHAT

## Name

CHAT — Send a chat message

## Synopsis

CHAT ...		
Data	Type	Example
Opcode	TableToControl	CHAT
Message	string	hola todos

## Description

Send a chat message

## Message Data

None

## Usage

This message appears during the WAITING or PLAYING state.

# INFO

## Name

INFO — Request seat or player information

## Synopsis

INFO ...		
Data	Type	Example
Opcode	TableToControl	INFO
Seat number	integer	-1

## Description

Request information about a specific seat, or about all seats. (All seats are returned if the seat number is -1.)

## Message Data

None

## Usage

This message appears during the WAITING or PLAYING state.

# A.3. Symbolic identifiers and their values

## ControlToTable

## Name

ControlToTable — Opcodes from GGZ core client to the game client module

## Synopsis

---

Identifier	Value	Description
GAME_LAUNCH	0	message
GAME_SERVER	1	message
GAME_SERVER_FD	2	message
GAME_PLAYER	3	message
GAME_SEAT	4	message
GAME_SPECTATOR_SEAT	5	message
GAME_CHAT	6	message
GAME_STATS	7	message
GAME_INFO	8	message

## Description

All opcodes are of type integer.

## TableToControl

### Name

`TableToControl` — Opcodes from game client module to GGZ core client

## Synopsis

Identifier	Value	Description
GAME_STATE	0	message
STAND	1	request
SIT	2	request
BOOT	3	request
BOT	4	request
OPEN	5	request
CHAT	6	request
INFO	7	request

## Description

All opcodes are of type integer.

## GGZSeatType

### Name

GGZSeatType — Possible seat assignments for a table

### Synopsis

Identifier	Value	Description
GGZ_SEAT_NONE	0	Not initialized yet (invalid)
GGZ_SEAT_OPEN	1	Initialized to open, will be filled later
GGZ_SEAT_BOT	2	Internal or external AI player
GGZ_SEAT_PLAYER	3	Human player
GGZ_SEAT_RESERVED	4	Reserved for AI or human player of a certain name

### Description

All seat types are of type integer.

## GGZModState

### Name

GGZModState — Possible game states for a game client

### Synopsis

Identifier	Value	Description
STATE_CREATED	0	...
STATE_CONNECTED	1	...
STATE_WAITING	2	...
STATE_PLAYING	3	...
STATE_DONE	4	...

## **Description**

All states are of type integer.