Firebase Protocol

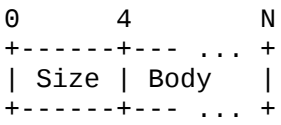
Protocol Version: 1.7
Firebase Version: 1.7

Table of Contents

General

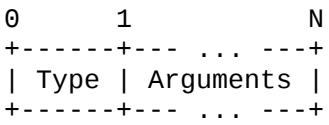
Packet format:

The packets first four bytes will contain the total size of the entire packet (32 bit signed integer).



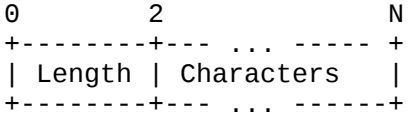
Body:

The body of the packet will firstly contain a byte that specifies the type of the packet.



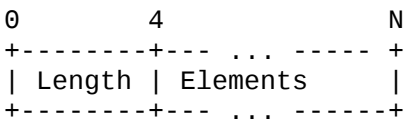
Strings:

Maximum length is 2^{15} characters.



Arrays:

All arrays are prefixed by the length as an integer



Integers:

Unsigned or Signed, big-endian, 32 bit unless specified otherwise.
Signed or Unsigned is specified in the protocol xml-definition.

Structs:

Structs that are contained within a struct does not include the header, i.e. size and type.

Amounts:

Sent as integers, need to be divided by 100.

Identifiers:

The server allows a single player to participate in multiple games at the same time. There is no limit on the number of games that a player can participate in.

Game and player identifiers:

GID = Game ID
TID = Table ID
PID = Player ID.
MTTID = Tournament ID (Multi Table Tournament)
PID and GID are integers.

PID 0 belongs to the server.

Communication

Request / Response

Packets that have the name xxxRequestPacket are all client --> server and is for a single player only. Packets that matches xxxResponsePacket are all single responses for a request done by the receiving client.

A Response packet for a client will never be sent to another client.

Notifications

All packets that starts with Notify are notifications from the server. Notifications are related to actions performed by other players. Information regarding a request made by client A can thus be propagated to many players in the form of a Notify packet.

Game List

Below is the mapping of game type identifiers that are internal to Cubeia. These are primarily used for testing purposes so do not use these.

ID	Packet
1	High Card Wins
2	Tank
3-10	Internal games prototyping
99	Test Game

Type List

Below is the mapping of type identifiers to packet definition.

ID	Packet
0	Version
1	Game Version
2	Good
3	Bad
4	System Message
5	Parameter
6	Parameter Filter
7	Ping
10	Login Request
11	Login Response
12	Logout
13	Player Info
14	Forced Logout
15	Seat Info
16	Player Query Request
17	Player Query Response
18	System Info Request
19	System Info Response
30	Join Request
31	Join Response
32	Watch Request
33	Watch Response
34	Unwatch Request
35	Unwatch Response
36	Leave Request
37	Leave Response
38	Table Info Request
39	Table Info Response
60	Notify Join
61	Notify Leave
62	Notify Joined
63	Notify Watching
64	Kick Player
80	Table Chat
100	Game Transport
101	Service Transport
103	Local Service Transport
104	MTT Transport

105	Encrypted Transport
120	Join Chat Channel Request
121	Join Chat Channel Response
122	Leave Chat Channel
123	Notify Channel Chat
124	Channel Chat
140	Parameter
141	Parameter Filter
142	Lobby Query
143	Table Snapshot
144	Table Update
145	Lobby Subscribe
146	Lobby Unsubscribe
147	Table Removed
148	Tournament Snapshot
149	Tournament Update
150	Tournament Removed
151	Lobby Object Subscribe
152	Lobby Object Unsubscribe
153	Table Snapshot List
154	Table Update List
155	Tournament Snapshot List
156	Tournament Update List
170	Filtered Join Table Request
171	Filtered Join Table Response
172	Filtered Join Cancel Request
173	Filtered Join Cancel Response
174	Filtered Join Table Available
200	Probe Stamp
201	Probe
205	MTT Register Request
206	MTT Register Response
207	MTT Unregister Request
208	MTT Unregister Response
209	MTT Seated
210	MTT Picked Up

General Packets

Version

Used by client to verify Firebase protocol version.

Type id: 0

Field	Size	Remark
Game	4	Game id
Operator ID	4	
Protocol	4	Protocol version

Game Version

Used by client to verify a Game version. The game must implement the version handling and since we do not know of the preferred format, we send the game version as String.

Type id: 1

Field	Size	Remark
Game	4	Game id
Operator ID	4	
Protocol	String	Game specific version data

Good

A command / request was accepted.

Type id: 2

Field	Size	Remark
Cmd	4	Command reference
Extra	4	Used as needed

Bad:

Command was not accepted, error code describes the problem.

CMD: Command code such as LOGIN, etc.
ERROR: See bottom of this document for the list of error codes.

Type id: 3

Field	Size	Remark
Cmd	4	Command reference
Error	4	Error code

System Message:

Global system message. Can be sent by Admin or by implementing game. Type and Level are not defined within Firebase, i.e. They are free to use as suited.

Type id: 4

Field	Size	Remark
Type	4	
Level	4	
Message	String	

Parameter

A data structure which is contained within other packets. The Parameter models a key-value parameter of the specified type. Type is defined in the Parameter enumeration.

Type id: 5

Field	Size	Remark
Key	String	
Type	1	
Value	...	Byte array with data representing the value for the type specified in Type (packet.type)

Parameter Filter

A data structure which is contained within other packets. The Parameter Filter applies a constraint on top of a Parameter. This is typically used for creating lobby filters (e.g. Filtered Join).

Operator is defined in the Parameter Filter enumeration.

Ping

A simple packet sent by the server to detenct client failures.

Type id: 7

Field	Size	Remark
Id	4	Integer ping id. Should not be modified.

Type id: 6

Field	Size	Remark
Param	Parameter	Contains a Parameter packet
Op	1	Operator for the Parameter.

Example:

- *Key:* *'_SEATED'*
- *Type:* *INT*
- *Value:* *0*
- *Op:* *GREATER THAN*

The Parameter Filter above will match every entity that has an attribute (key) called SEATED with a number greater then 0.

Player / Client Information Packets

Login Request

Request to login. Will receive a Login Response as response from the server.

Type id: 10

Field	Size	Remark
User	String	
Password	String	
Operator ID	4	Operator id is used to locate login handlers.
Credentials		Byte array with arbitrary credential data.

Login Response

Response to a login request.

Status is defined in the Response Status enumeration.

Type id: 11

Field	Size	Remark
Screenname	String	Your screenname
Pid	4	Your unique player id (only applicable if status is OK)
Status	1	Se definition above
Code	4	Error code for denied login
Message	String	Denied message, avoid using this if you can. Using a predefined error code is much more efficient.
Credentials		Byte array with arbitrary credential data.

Logout

Logout from the server. No response will be sent since we will never deny a logout.

If the leave tables flag is set then Firebase will send a leave request to all tables the player is currently seated at. If the flag is false then players at table will be handled similar to a client disconnect, i.e. they will be flagged WAIT_REJOIN and finally DISCONNECTED.

If you want to be able to logout the client and then get 'notify seated at' messages when login in again (if before reaper timeout), then you should set the flag to 'false'.

Type id: 12

Field	Size	Remark
Leave Tables	1	Boolean flag

Player Info

A minimal information struct regarding a player.

Type id: 13

Field	Size	Remark
Pid	4	Player id
Nick	String	
Details	...	List of Parameter's, same format as TableUpdate

Forced Logout

The client was forcibly logged out by the server. The use of a code-response is strongly suggested. Using string-format messages uses high bandwidth and does not allow for different client side locales.

The client will be removed from all tables he is seated at the same way a regular logout would.

Code:
1 = Client logged in remotely
2 = Kicked by server

Type id: 14

Field	Size	Remark
Code	4	Message code (recommended)
Message	String	Message in string format (not recommended)

Seat Info

Information regarding a seat at a table. This will be part of a table info response.

Status is defined in the Player Status enumeration and is the status of the player.

Type id: 15

Field	Size	Remark
Tableid	4	Table id
Seat	1	Seat id
Status	1	Status of the seat
Player		A Player Info packet, null if no one is seated

Player Query Request

Request extended information about a player. This request will be routed to an implementation specific handler.

Type id: 16

Field	Size	Remark
Pid	4	Player id

Player Query Response

Extended information about a player. If you provide your own player lookup implementation then use the Status flag to mark invalid entities.

Type id: 17

Field	Size	Remark
Pid	4	Player id
Nick	String	
Status	1	OK: Found, Denied: Not found, Failed: Failed
Data		Byte array of arbitrary data

System Info Request

Request system information from the game server. The packet contains no contextual fields.

Type id: 18

Field	Size	Remark

System Info Response

Contains current number of logged in players together with customizable parameters. The parameters are defined and inserted by 3rd party service implementation (if present).

Type id: 19

Field	Size	Remark
Players	4	Player id
Params	...	Parameters that will be included in the response as handled by the service implementation.

Table Requests and Responses

Join Request

Join a table. A Join Response will be sent back. If you are seated at the table and you have not watching the table then you will also get a SeatInfoPacket for every player seated at the table including yourself.

You can include any game specific needed attributes in the params field.

If you set seat id = -1 then the server will place at any free seat if available.

Type id: 30

Field	Size	Remark
Table id	4	Table id
Seat	1	Seat index (0 – x)
Params	...	Parameters that will be included in the join request as handled by the game implementation.

Join Response

Response to a join request.

Status is defined in the Join Response Status enumeration.

Type id: 31

Field	Size	Remark
Table id	4	Table id
Seat	1	Seat index (0 – x)
Status	1	

Watch Request

Watch a table. A Watch Response will be sent back. If successful, a set of SeatInfoPacket will be sent, one for each seated player at the table. After that you will receive GameTransportPackets as defined by the implementing game.

Type id: 32

Field	Size	Remark
Table id	4	Table id

Watch Response

Response to a watch request.

Status is defined by the Watch Response Status enumeration.

Type id: 33

Field	Size	Remark
Table id	4	Table id
Status	1	

Unwatch Request

Stop watching a table. You will stop receiving updates from the game when the request has been handled.

Type id: 34

Field	Size	Remark
Table id	4	Table id

Unwatch Response

Response to unwatch request.

Status is defined in the Response Status enumeration.

Type id: 35

Field	Size	Remark
Table id	4	Table id
Status	1	

Leave Request

Request to Leave Table. If you send this to a table where you are not seated then you will get a Leave Response with status = FAILED.

Type id: 36

Field	Size	Remark
Table id	4	Table id

Leave Response

Response to Leave Table request.

Status is defined in the Response Status enumeration.

Type id: 37

Field	Size	Remark
Table id	4	Table id
Status	1	

Table Query Request

Request extended table information.

Type id: 38

Field	Size	Remark
Table id	4	Table id

Table Query Response

Extended information about a table. The Status flag is used to mark invalid entities.

Type id: 39

Field	Size	Remark
Table id	4	Table id
Status	1	OK: Found, Denied: Not found, Failed: Failed
Seats	…	List of SeatInfoPackets

Create Table Request

Send this to request a table to be created. You will receive a Create Table Response with a response status and response code set by the game implementation (if applicable). The response will contain the sequence id as matched by the request.

The request can contain a list of player ids that will be invited to the table upon creation. All invited players will receive a Notify Invitation packet for the table. The activator can flag if the seats should be reserved as well, if so then the invitation notification will contain a seat value > -1 as well.

The game activator contains a possibility to alter the invitation player ids. This makes it possible to send in an internal group id instead, but then you need to match this behavior with the game implementation.

Type id: 40

Field	Size	Remark
Seq	4	Request sequence id
Gameid	4	List of seat info packets
Seats	1	Number of seats to create
Params	...	List of Parameters that will be included in the callback to the game activator. Use this for game/table specific data (e.g. Game type, password etc).
Invitees	...	List of player ids (integers) that will be invited to the table upon creation.

Create Table Response

Response for Create Table Request. If status is ok, then you would normally respond with a Join Table Request for the given table and seat.

Status is defined in the Response Status enumeration.

Type id: 41

Field	Size	Remark
Seq	4	Originating request sequence id
Table id	4	Id of the created table (-1 if denied)
Seat	1	Seat reserved for you. If you do no claim the seat it will be cleaned up after a while.
Status	1	
Code	4	Response code returned by the game implementation

Invite Players Request

Sent by client to invite other players to a specific table. This request will generate a Notify Invited Packet to all logged in recipients. There is no response packet for this request.

There is also no check for table or player ids, i.e. we do not check if the supplied table id does exist in the game space. All invitations are routed as-is.

Type id: 42

Field	Size	Remark
Tableid	4	Table id
Invitees	...	List of player ids (integers) that will be invited to the table.

Notify Invited

You were invited to a table by another table. If the seat is > -1, then the given seat has been reserved for you. If you do not claim the seat it will be cleaned up after a while and you are no longer guaranteed the spot.

Type id: 43

Field	Size	Remark
Inviter	4	Player id of the inviter
Screenname	String	Screenname of the inviter
Tableid	4	Table you are invited to
Seat	1	Seat. If -1 no seat was specified or reserved.

Server To Client Notifications

Notify Join

A player has join a table you are registered to as a watcher (or seated player).

Type id: 60

Field	Size	Remark
Table id	4	Table id
Pid	4	Player id
Nick	String	
Seat	1	Seat index (0 - x)

Notify Leave

A player has left a table you are registered to as a watcher (or seated player).

Type id: 61

Field	Size	Remark
Table id	4	
Pid	4	Player id

Notify Joined

If you have been disconnected (without a proper logout) and reconnect within the reconnect grace period (configurable, but default is 60 seconds), then you will receive one notify joined for each table you were seated at. To rejoin that table you should send a regular join request for the given table and seat.

Type id: 62

Field	Size	Remark
Table id	4	
Seat	1	

Notify Watching

If you have been disconnected (without a proper logout) and reconnect within the reconnect grace period (configurable, but default is 60 seconds), then you will receive one notify watching for each table you were watching. To watch that table again you should send a regular watch request for the given table.

Type id: 63

Field	Size	Remark
Table id	4	

Kick Player

Sent when a player is kicked from a table. The reason codes are game specific and should be defined by the game. This packet will only be sent to the player that was kicked, the other players at the table will receive a leave notification as normal.

NB: Currently not used, only forced logout is available

Type id: 64

Field	Size	Remark
Table id	4	
Reason code	2	The meaning of the reason codes are game specific and should be defined by the game.

Dual Way Packets (Server <--> Client)

Table Chat

A player is chatting at the table. The information contained is the same regardless of direction.

Type id: 80

Field	Size	Remark
Table id	4	Table id
Pid	4	Player id
Message	String	

Game / 3rd Party Specific Packets

Game Transport

This packet wraps the game specific protocol data.

Player id can be omitted on incoming packets since it will be overridden in the client session.

Type id: 100

Field	Size	Remark
Table id	4	Table id
Pid	4	Player id
Gamedata	...	A byte-array of game specific data.

Service Transport

This packet wraps the service specific protocol data. A service can be identified in two ways, by namespace or by contract. Both identifiers are specified in the service's deployment descriptor (service.xml).

Player id can be omitted on incoming packets since it will be overridden in the client session.

Id Types are defined in the Service Identifier enumeration.

Type id: 101

Field	Size	Remark
Pid	4	Player id
Service	String	Identifier of the targeted service
IdType	1	Type of identifier
Servicedata	...	A byte-array of service specific data.

Local Service Transport

This packet wraps service specific data that will be directed to the deployed local service (if any). This service can be addressed before a client has login (as opposed to Service Transport which requires a logged in client).

Type id: 103

Field	Size	Remark
Seq	4	Sequence identifier. Set this in the client, the response can be tagged with the corresponding seq id.
Servicedata	...	A byte-array of service specific data.

MTT Transport

This packet wraps the tournament specific protocol data.

Type id: 104

Field	Size	Remark
Mtt id	4	Tournament id
pid	4	Player id
Mttdata	...	A byte-array of tournament specific data.

Encrypted Transport

This packet wraps encrypted protocol data. The funtion byte can be one of the following (as defined in CryptoConstants):

- -1 : ILLEGAL_PACKET : Sent by the server if it does not support encryption
- 0 : ENCRYPTED_DATA : Actual transport payload
- 1 : SESSION_KEY_REQUEST : Sent by client to request key
- 2 : SESSION_KEY_RESPONSE : Send by server after key request

Type id: 105

Field	Size	Remark
Func	1	Packet function, see above
Payload	...	A byte-array of encrypted data.

Channel Chat Packets

Join Chat Channel Request

Request to join a chat channel. A Join Chat Channel Response will be sent back. If successful you will start to receive Chat Notifications and can send Channel Chat packets.

Type id: 120

Field	Size	Remark
Channel id	4	

Join Chat Channel Response

Response to join chat channel.

Status is defined in the Response Status enumeration.

Type id: 121

Field	Size	Remark
Channel id	4	
Status	1	

Leave Chat Channel Request

Request to join a chat channel. No response is sent back since we will never deny a leave chat channel request.

Type id: 122

Field	Size	Remark
Channel id	4	

Notify Channel Chat

Someone has said something in the chat.

Type id: 123

Field	Size	Remark
Pid	4	Player id
Channel id	4	
Target id	4	Destinated player id. If 0 it is to all in the channel, if it is the client's pid it is a private message.
Nick	String	
Message	String	

Channel Chat

Chat in a Chat Channel.

Type id: 124

Field	Size	Remark
Channel id	4	
Target id	4	Destinated player id for private message or 0 for all in the channel.
Message	String	

Lobby Handling Packets

Lobby Query

Query the lobby for table information. This is very resource demanding so if you are planning on successive updates, please use the subscription model instead.

You must specify which lobby by setting the type accordingly. The possible lobby types are defined in the LobbyType enum. The value used should be the ordinal of the enum.

The query uses a relative path based on the lobby tree.

Gameid is used as a universal id for the deployment identifier, e.g. For games it is the id in the game.xml, for tournaments it is the id in the tournament.xml.

Type id: 142

Field	Size	Remark
Gameid	4	Game/MTT (etc) id
Address	String	The lobby path to query
Type	1	Lobby type (See LobbyType enum)

Example:
Given a lobby tree like (top level node is always game id):

/99/a/b/c
/99 /a/b/d

You can use address like this:

- ***/99/a/b/c** – Gives all tables under the end node of /99/a/b/c*
- ***/99/a/b** – Gives all tables under the end nodes /99/a/b/c and /99/a/b/d*
- ***/99** – Gives all tables for game with id = 99*

Table Snapshot

Full information regarding a Table. If you are subscribing to a lobby path, then all successive packets for this table will be Table Update Packets.

A table snapshot will contain all default attributes, but since the parameters are set runtime Firebase cannot guarantee that all game-specific attributes are set before a table snapshot is sent out. If your lobby specifies a specific parameter, then the client needs to handle the case of a missing parameter.

Type id: 143

Field	Size	Remark
Table id	4	
Address	String	Address in the lobby tree
Name	String	
Capacity	2	How many seats are there in total
Seated	2	How many players are currently seated at this table
Params	...	List of Parameters which are attributes for the table. See Parameter for format.

Table Update

Delta changes for the given table. This packet assumes that you hold a reference for the table id which contains the non-changed attributes. The Parameter map contains all attributes that are changed.

Type id: 144

Field	Size	Remark
Table id	4	
Seated	2	How many players are currently seated at this table
Params	...	List of Parameters which are attributes for the table.
Removed Params	...	List of parameters (string names) that have been removed

Lobby Subscribe

Start a subscription to the lobby. The subscription will be active on the given lobby path as well as all sub-nodes.

E.g.
If the lobby tree is:
a/b/c
a/b/d
a/e/f

And you send in a subscription request for: a/b/, then you will receive updates for nodes:
a/b
a/b/c
a/b/d

You must specify which lobby by setting the type accordingly. The possible lobby types are defined in the LobbyType enum. The value used should be the ordinal of the enum.

Gameid is used as a universal id for the deployment identifier, e.g. For games it is the id in the game.xml, for tournaments it is the id in the tournament.xml.

When subscribing you will first receive a full snapshot for all tables contained in all nodes affected (i.e. the sub nodes). After the full snapshot you will start to receive delta updates. Delta updates can (and will) include full snapshots, delta snapshots and table removed packets. All according to changes in the lobby.

It is possible to subscribe to lobby before logging in.
Type id: 145

Field	Size	Remark
Type	1	Lobby type (See LobbyType enum)
Game id	4	Game/MTT (etc) id
Address	String	Address in the lobby tree

Lobby Unsubscribe

Stop your subscription. This will unsubscribe you to the given address and all sub nodes. This means that if you can unsubscribe to a sub set of your total subscriptions.

You must specify which lobby by setting the type accordingly. The possible lobby types are defined in the LobbyType enum. The value used should be the ordinal of the enum.

Gameid is used as a universal id for the deployment identifier, e.g. For games it is the id in the game.xml, for tournaments it is the id in the tournament.xml.

E.g.
If the lobby tree is:
a/b/c
a/b/d
a/e/f

And you have subscribed to: 'a' (which means you will receive updates for the whole tree in this example).

If you now unsubscribe to 'a/b' you will continue to receive update for 'a/e/f', since that node will not be included in the sub-tree of 'a/b'.

Type id: 146

Field	Size	Remark
Type	1	Lobby type (See LobbyType enum)
Game id	4	
Address	String	Address in the lobby tree

Table Removed

This table has been removed from the lobby. Table removed packets can be received for tables not included in the initial snapshot.

Type id: 147

Field	Size	Remark
Table id	4	

Tournament Snapshot

Full lobby information regarding a tournament. If you are subscribing to a lobby path, then all successive packets for this table will be Tournament Update Packets.

Since tournaments are more diverse then tables, the common parameters capacity, registered, name and players are not included as fixed fields but rather as parameters. This way they will be included in the delta changes which is a more efficient way of sending out volatile data.

The fields supplied by Firebase are:

Parameter key	Description
TOURNAMENT_ID	ID of the implementing tournament logic (MTTLogic identifier).
NAME	Name of the tournament
CAPACITY	Mtt capacity, ie. maximum number of participating players.
REGISTERED	How many players are currently registered to the tournament.
ACTIVE_PLAYERS	Remaining(active) number of players

Type id: 148

Field	Size	Remark
MTT id	4	Tournament id (MTT - Multi Table Tournament)
Address	String	Address in the lobby tree
Params	...	List of Parameters which are attributes for the table. See Parameter for format.

Tournament Update

Delta changes for the given tournament. This packet assumes that you hold a reference for the tournament id(mttid) which contains the non-changed attributes. The Parameter map contains all attributes that are changed.

Se Tournament Snapshot for a list of fields provided by Firebase.

Type id: 149

Field	Size	Remark
MTT id	4	
Params	...	List of Parameters which are attributes for the tournament.
Removed Params	...	List of parameters (string names) that have been removed

Tournament Removed

This tournament has been removed from the lobby.

Type id: 150

Field	Size	Remark
Tournament id	4	

Lobby Object Subscribe

Start a subscription to a specific lobby object.

To stop subscribe to a lobby object you must do an explicit Lobby Object Unsubscribe. Lobby object subscriptions will not be terminated by a regular Lobby Unsubscribe.

It is possible to subscribe to lobby objects before logging in.

Lobby enum, game id and update handling are the same as for a regular Lobby Subscription.

Type id: 151

Field	Size	Remark
Type	1	Lobby type (See LobbyType enum)
Game id	4	Game/MTT (etc) id
Address	String	Address in the lobby tree
Object Id	4	Id of the specific object (e.g. tableid, mttid)

Lobby Unsubscribe

Stop your subscription to a lobby object.

Lobby enum, game id and update handling are the same as for a regular Lobby Unsubscription.

Type id: 152

Field	Size	Remark
Type	1	Lobby type (See LobbyType enum)
Game id	4	
Address	String	Address in the lobby tree
Object Id	4	Id of the specific object (e.g. tableid, mttid)

Table Snapshot Packet List

A list of table snapshots.

Type id: 153

Field	Size	Remark
Snapshots	...	Array of TableSnapshotPacket's

Table Update Packet List

A list of table updates.

Type id: 154

Field	Size	Remark
udpates	...	Array of TableUpdatePacket's

Tournament Snapshot Packet List

A list of tournament snapshots.

Type id: 155

Field	Size	Remark
Snapshots	...	Array of TournamentSnapshotPacket's

Tournament Update Packet List

A list of tournament updates.

Type id: 156

Field	Size	Remark
Updates	...	Array of TournamentUpdatePacket's

Filtered Join Packets

Filtered Join Table Request

Send a request to join a table with the given attributes. The handler on the server side will first try to match the request towards current tables. If no match is found then the request will be placed in the server side waiting list. In that case you will be prompted with a join packet when you are granted a seat.

The platform internal table attributes are:

_ID	id
_NAME	name
_CAPACITY	How many seats are at the table
_SEATED	How many are seated at the table
_WATCHERS	How many are watching the table
_GAMEID	The associated game's id
_LAST_MODIFIED	Last time of modification

Type id: 170

Field	Size	Remark
Seq	4	Request Sequence ID
Game id	4	Which game
Address	String	Path in the lobby
Params	...	List of Parameter Filters which are attributes for the table.

Example:

- *Game: 99*
- *Address: /*
- *Key: '_SEATED'*
- *Type: INT*
- *Value: 0*
- *Op: GREATER THAN*

The request above will trigger when any tables for game id 99 has a table that has one or more seated players. (Internal fields like seated, game id etc. are all prefixed with an underscore, so seated = _SEATED).

Filtered Join Table Response

Response to the filtered request. This is not a seating response but rather a response that your request has been acknowledged.

Status is defined in the Filtered Join Response Status enumeration.

SEATING: A table was found and you will be seated / receive table available soon.
WAIT_LIST: No table was found right away so you are placed on the waiting list.

Type id: 171

Field	Size	Remark
Seq	4	Request Sequence ID
Game id	4	Which game
Address	String	Path in the lobby
Status	1	

Filtered Join Cancel Response

Cancel a request to join a table.

Type id: 172

Field	Size	Remark
Seq	4	Sequence number for the request

Filtered Join Cancel Response

Response to the filtered request. This is not a seating response but rather a response that your request has been acknowledged.

Status is defined in the Response Status enumeration.

FAILED: An error was reported when removing the request (not expected)
DENIED: The request was not found. Perhaps we have found a match before the request was handled.

Type id: 173

Field	Size	Remark
Seq	4	Sequence number for the request
Status	1	

Filtered Join Table Available

A table has been found that matches your filtered join request.

Type id: 174

Field	Size	Remark
Seq	4	Sequence number for the request
Table id	4	
Seat	1	

Probe Packets

Probe Stamp

This is a data structure which is contained in other packets. It contains a class name and a time stamp (UTC)

Type id: 200

Field	Size	Remark
Clazz	String	Class name of the class that placed the time stamp
Time stamp	8	Time stamp in UTC format (ms since 1970)

Probe

The probe is used for benchmarking and testing reasons. As the probe messages travels through the system it is stamped with different classes at strategical checkpoints. This allows for a fine grained latency check of the system.

Type id: 201

Field	Size	Remark
Id	4	Identifier for the probe packet (set by client)
Table id	4	Table to target for a probe
Stamps	...	List of Probe Stamp packets.

MTT Packets

MTT Register Request

Register for an MTT. An MTT Register Response will be returned by server.

Type id: 205

Field	Size	Remark
MTT id	4	

MTT Register Response

Response to a register request.

Status is defined in the Tournament Register Response Status enumeration.

Type id: 206

Field	Size	Remark
MTT id	4	
Status	1	

MTT Unregister Request

Unregister for an MTT. An MTT Unregister Response will be returned by server.

Type id: 207

Field	Size	Remark
MTT id	4	

MTT Unregister Response

Response to an unregister request.

Status is defined in the Response Status enumeration.

Type id: 208

Field	Size	Remark
MTT id	4	
Status	1	

MTT Seated

You have been seated at a table in a tournament.

Type id: 209

Field	Size	Remark
MTT id	4	Tournament id
Table id	4	Table id
Seat	1	Seat index (0 – x)

MTT Picked Up

You have been picked up from a table in a tournament. This could either be because you are out or that you are being move to a different table as a result of a table merge/balancing.

Type id: 210

Field	Size	Remark
MTT id	4	Tournament id
Table id	4	Table id
Keep watching	1	True if the player is still a watcher

Enumerations

Below are enumerations as defined in the protocol.

Definitions

Parameter Type

Key	Remark
STRING	The parameter's value is of type String
INT	The parameter's value is of type Integer

Parameter Filter

Key
EQUALS
GREATER_THAN
SMALLER_THAN
EQUALS_OR_GREATER_THAN
EQUALS_OR_SMALLER_THAN

Lobby Type

Key	Remark
REGULAR	The regular lobby (cash game tables)
MTT	The tournament lobby

Tournament Attributes

The name of the enumerations matches the supplied default attributes in the tournament lobby parameters.

Key
NAME
CAPACITY
REGISTERED
ACTIVE_PLAYERS
STATUS

Service Identifier

Key	Remark
NAMESPACE	Use namespace as identifier
CONTRACT	Use the contract class as identifier

Player Status

Key
CONNECTED
WAITING_REJOIN
DISCONNECTED
LEAVING
TABLE_LOCAL
RESERVATION

Response Statuses

Response Status

A generic response status enumeration.

Key
OK
FAILED
DENIED

Join Response Status

A generic response status enumeration.

Key
OK
FAILED
DENIED

Watch Response Status

Key
OK
FAILED
DENIED
DENIED_ALREADY_SEATED

Filtered Join Response Status

Key
OK
FAILED
DENIED
SEATING
WAIT_LIST

Tournament Register Response Status

Key
OK
FAILED
DENIED
DENIED_LOW_FUNDS
DENIED_MTT_FULL
DENIED_NO_ACCESS