

Paintball Protocol

Packet Structure

Byte Offset	Type	Data	Description
0	Byte	Asterisk (*)	Signals start of packet
1	Char	Packet Type	Describes type of packet (also implicitly describes payload length)
3	(Variable)	Payload	Different things depending on packet type

Packet Types

Login Packet (Type F):

Payload Byte Offset	Type	Data	Description
0	String	Username	Client→Server: username of new client signing on. Server→Client: username of new client that has signed on (to update clients on new players)

Purpose: Sent by client to login to the server. Sent by server to all existing clients to update clients on new clients.

Update Packet (Type A):

Payload Byte Offset	Type	Data	Description
0	Char	Object Type	Type of object being updated (whether it's a player, paintball, obstacle, etc.)
2	Int	Object ID	ID number of object that needs to be updated (every object in the game – players, paintballs, obstacles – has a unique ID number). If the client detects a new ID number, the assumption is that a new object has been created.
6	Int	Position X	X value of updated position (-1 if object is dead)
10	Int	Position Y	Y value of updated position (-1 if object is dead)
14	Int	Velocity X	X value of updated velocity vector
18	Int	Velocity Y	Y value of updated velocity vector

22	Double	Facing	Updated facing angle of object (radians, [1 0] is considered 0 rad, [0 1] is considered pi/2 rad)
----	--------	--------	---

Purpose: Sent by server to update clients on the properties of the game state. Each packet will contain information on a single object within the game.

Notes: Server→Client only

Player Update Request Packet (Type C)

Payload Byte Offset	Type	Data	Description
0	Int	Velocity X	X value of new velocity
4	Int	Velocity Y	Y value of new velocity
8	Double	Facing	New facing angle (radians, [1 0] is considered 0 rad, [0 1] is considered pi/2 rad)

Purpose: Sent by client to server to request an update on the player's velocity and facing.

Notes: Client→Server only

Shoot Update Request Packet (Type D)

Payload Byte Offset	Type	Data	Description
0	Byte	Shooting State	Either 0, 1, or 2, each describing a different shooting state: 0: not shooting (used to turn off automatic fire) 1: single shot 2: automatic fire (will continue to fire until a not shooting request is sent)

Purpose: Sent by client to server to request an update on the player's shooting state. (The server reserves the right to limit firing rates, so flooding the server with single shot requests will not necessarily make you shoot any faster than sending a single automatic fire request)

Notes: Client→Server only

Game Dump Request Packet (Type E)

Payload Byte Offset	Type	Data	Description
0	Null	N/A	This packet type has no payload.

Purpose: Sent by client to server to request a complete game dump.

Notes: Client→Server only

Protocol

Login: Clients connect to the server and send a Login Packet. The server registers the clients and sends out Login Packets to all other clients that have logged in to inform them of a new client joining. The server autoassigns new players to teams.

Game Start: The game is always running, players can join whenever.

Mid-Game: Game state is controlled completely by the server. Clients send Player and Shoot Update Request Packets to the server whenever it is necessary (whenever they want to change their velocity or to shoot). The server processes the requests and updates the game state accordingly. Every time the game state changes (a player's velocity changes, a player shoots, a player dies, etc.), the server sends out an Update Packet to all clients containing information on whatever needs to be updated. Additionally, periodically the server will send a complete game dump to all clients to resync the clients. The game dump will simply consist of multiple Update Packets, all of which put together describes the entire updated game state.

Game Dumps: At any point during the game, clients may send a Game Dump Request Packet to the server. If the server receives one, it will send a complete game dump to the appropriate client.

Game End: Game end depends on game type. We're still developing how to handle this.