Network packets

Fix Me! - Needs additional information on whatever client sends on net packet types 3, 4, 5 and 6.

Network packets are a way of contacting to the host or client via sending or receiving data. The amount of defined network packets has been changing ever since the implementation of Http Dll 2.1[1] in Brick Hill Legacy Beta.

The table will be updated.

Brick Hill Legacy (latest) Network Packets

Network Packet type	**What does the network packet contain?**
1 (Authentication packet, defined by Node Hill as `Authentication`)	What **Client** sends: **Token**[2] (string) **Client version** (string) What **Host** sends[3]: **Network ID**[4] (32-bit integer) **Brick Count** (32-bit integer) **User ID**[5] (32-bit integer) **Username** (string) Are you an **admin**?[6] (8-bit integer) **Membership type** (8-bit integer) **Game ID** (32-bit integer) **Game Name** (string)
2 (Player position Packet, defined by Node Hill as `Position`)[7]	What **Client** sends: **Position by X** (float int) **Position by Y** (float int) **Position by Z** (float int) **Rotation by Z** (float int) **Camera rotation by Z** (float int)
3 (New player packet, client creates the `obj_figure` Object, defined by Node Hill as `SendPlayers`)	What **Host** sends: **Amount of players** (8-bit integer) **Network ID** (32-bit integer) **Username** (string) **User ID**[8] (32-bit integer) Is new player an **admin**?[9] (8-bit integer) **Membership type** (8-bit integer)

Network Packet type	What does the network packet contain?
4 (Client Data Packet, defined in GameMaker Client via `packet_handler` script; defined by Node Hill as `Figure`)	**Client** receives: **ID type** (string) **Network ID** (32-bit integer) **ID types**, defined in GameMaker client: **A** - Position by X (float int) **B** - Position by Y (float int) **C** - Position by Z (float int) **D** - ? (float int, **unused**) **E** - Camera Rotation (float int, **unused**) **F** - Player rotation by Z (float int) **G** - Player scale by X (float int) **H** - Player scale by Y (float int) **I** - Player scale by Z (float int) **K** - Head color (32-bit integer) **L** - Torso color (32-bit integer) **M** - Left Arm color (32-bit integer) **N** - Right arm color (32-bit integer) **O** - Left leg color (32-bit integer) **P** - Right Leg color (32-bit integer) **Q** - Face Item ID (string) **R** - Shirt Item ID[10] (string) **S** - Pants Item ID (string) **T** - T-Shirt Item ID (string) **U** - First Equipped Hat Item ID, receives Mesh UUID (string) and Texture UUID (string) **V** - Second Equipped Hat Item ID, receives Mesh UUID (string) and Texture UUID (string) **W** - Third Equipped Hat Item ID, receives Mesh UUID (string) and Texture UUID (string) **X** - Score (32-bit integer, can be negative) **Y** - Team (32-bit integer) The next ID Types are defined by GameMaker client as **Local Changes**: **1** - Walking Speed (32-bit integer) **2** - Jump Height (32-bit integer) **3** - Field of View (32-bit integer) **4** - Camera distance (32-bit integer, can be negative) **5** - Camera position by X (float int) **6** - Camera position by Y (float int) **7** - Camera position by Z (float int) **8** - Camera rotation by X (float int) **9** - Camera rotation by Y (float int) **a** - Camera rotation by Z (float int) **b** - Camera Type (string) **c** - Camera Object (32-bit integer) **e** - Health (float int) **f** - Speech[11] (string, **limited to 500 characters**[12]) **g** - Equipped tool: **Slot ID** (32-bit integer), **Mesh UUID** (string), **Texture UUID** (string) **h** - Sets `Arm` value to `-1` **i** - Is player alive?[13] (8-bit integer) The next ID Types are defined by GameMaker client as **“new clothing system”**: **j** - `Clothe1` (string) **k** - `Clothe2` (string) **l** - `Clothe3` (string) **m** - `Clothe4` (string) **n** - `Clothe5` (string)

Network Packet type	**What does the network packet contain?**
5 (Remove Player Packet, defined by Node Hill as `RemovePlayer`)	**Client** receives the user's **Network ID** (32-bit integer) and removes its `obj_figure` object from the game.
6 (Chat Message Packet, defined by Node Hill as `Chat`)	**Client** receives a formed by **Host** chat **Message** (string, 500 limit)

7 (Environment Data, GUI; defined by Node Hill as `PlayerModification`)	What **Client** receives:
	Data Type (string):
	`topPrint` - Top Text: **Message** (string), **Time** (32-bit integer)
	`centerPrint` - Middle Text: **Message** (string), **Time** (32-bit integer)
	`bottomPrint` - Bottom Text: **Message** (string), **Time** (32-bit integer)
	`Ambient` - Ambient color (32-bit integer)
	`Sky` - Sky color (32-bit integer)
	`BaseCol` - Baseplate color (32-bit integer)
	`BaseSize` - Baseplate size (32-bit integer)
	`Sun` - Sun intensity (32-bit integer)
	`kick` - Kicks the client[14]
	`prompt` - Prompt message (string)
	`WeatherSnow` - Sets the weather to `snow`
	`WeatherRain` - Sets the weather to `rain`
	`WeatherSun` - Sets the weather to `sun`
	`TweakDisableFigureCulling` - Disables the Figure culling option
	`TweakDisableLighting` - Disables lighting
	`TweakRaySpacing` - ? (float int)
	`MoreClickInfo` - enables more information on click (8-bit integer)

8 (Kill Packet, defined by Node Hill as `Kill`)	**Client** receives the user's **Network ID** (32-bit integer) and sets the `alive` variable to `0` to its corresponding `obj_figure` object.
9 (Brick Modification Packet, defined by Node Hill as `Brick`)	**Client** receives the `obj_brick`'s **Network ID** (32-bit integer) and the type of **Modification** (string). Brick Modifications: `pos`[15] - **Position by X** (float int) **Position by Y** (float int) **Position by Z** (float int) `rot`[16] - **Rotation by Z** (float int) `scale`[17] - **Scale by X** (float int) **Scale by Y** (float int) **Scale by Z** (float int) `kill`[18] - Time until destruction (32-bit integer) `destroy` - Removes the brick `col`[19] - Color (32-bit integer) `alpha`[20] - Alpha (float int) `lightcol`[21] - Color (32-bit integer) `lightrange`[22] - Range (32-bit integer) `model`[23] - Mesh UUID (string) Texture UUID (string) `collide`[24] - Collision state (8-bit integer) `clickable`[25] - Clickable state (8-bit integer) Clickable distance (32-bit integer) `global` - ? `ignoreLight`[26] - makes `ignoreLight` variable true
10 (New team packet, defined by Node Hill as `Team`)	**Client** receives: **Team ID** (32-bit integer) **Name** (string) **Color** (32-bit integer)

Network Packet type	What does the network packet contain?
11 (New tool packet, defined by Node Hill as `Tool`)	**Client** receives: **Active state** (8-bit integer) **Slot ID** (32-bit integer) **Name** (string)
12 (Bot Data packet, defined in GameMaker Client via `packet_handler_figure` script; defined by Node Hill as `Bot`)	**Client** receives: **String ID** (string) **Figure ID** (32-bit integer) Regarding the `packet_handler_figure` Script: It generally contains the same information as `packet_handler` Script, except it does not need Local Changes.
13 (New Projectile Packet, defined by Node Hill as `Projectile`)	**Client** receives: **Active state** (8-bit integer) **Projectile ID** (32-bit integer) **Diameter** (32-bit integer) **Color** (32-bit integer) **Position by X** (float int) **Position by Y** (float int) **Position by Z** (float int) **Direction** (32-bit integer) **Direction by Z** (32-bit integer) **Velocity** (32-bit integer)
14 (Clear Map Packet, defined by Node Hill as `ClearMap`)	**Client** removes all the bricks after it has received this packet
15 (Remove Bot packet, defined by Node Hill as `DestroyBot`)	**Client** receives the **Figure ID** (32-bit integer) and removes the bot by its **Network ID**
16 (Remove Brick packet, defined by Node Hill as `DeleteBrick`)	**Client** receives the **Brick ID** (32-bit integer) and removes the brick by its ID.
17 (`SendBrick` Packet, Node Hill definition)	What **client** receives: **Network ID** (32-bit integer) **Position by X** (float int) **Position by Y** (float int) **Position by Z** (float int) **Scale by X** (float int) **Scale by Y** (float int) **Scale by Z** (float int) **Color** (32-bit integer) **Transparency** (float int)
18 (`Heartbeat` packet, sent in Brick Hill Legacy Client via `update_delta` script)	What **Client** sends: **Type ID** (18) **Message** (sent via `socket_write_message`[27])

[1] GameMaker 8.1 extension with better networking features.

[2] user's auth key

[3] should be received by client

[4] object id of a `obj_client`, assigned by Node Hill

[5] retrieved via Brick Hill site API, hence the auth key **must be verified**

[6], [9], [13] 1 - true, 0 - false

[7] Earlier: unused method of loading bricks. Exists yet commented out in the game code of latest Brick Hill Legacy Client

[8] retrieved via Brick Hill site API on host side

[10] unused, old. Same with Pants, T-Shirt

[11] bubble chat message

[12] `If this packet function didn't check the length of the chat message then hosts using Game.on('chat') could cause clients to crash by relaying very large chat messages.` Self explanatory.

[14] this will become a vulnerability if the player socket will not be destroyed by **Host** itself

[15] Changes brick position

[16] Changes brick rotation by Z

[17] Changes the scale of brick

[18] Unanchors the brick

[19] Changes the brick color

[20] Changes the brick's transparency

[21] Changes the light color

[22] Changes the light range

[23] Sets the custom model to brick

[24] Changes the brick collision

[25] Changes the click state and distance for a brick

[26] Logically, this should ignore the light, Client source code got its option commented out (was supposed to support 8-bit integer)

[27] Http Dll 2.1 citation: `Sends a message. A message is sent by first sending the size of the message (as uintv), followed by the actual message data. This is handled automatically by the DLL, you don't need to do anything special if the receiver also uses Http Dll 2.`