Protocol

Heads up!

This article is about the protocol for the latest **stable** release of Minecraft **Java Edition** (1.18.2, protocol 758). For the Java Edition pre-releases, see Pre-release protocol. For the incomplete Bedrock Edition docs, see Bedrock Protocol. For the old Pocket Edition, see Pocket Edition Protocol Documentation.

This page presents a dissection of the current **Minecraft (https://minecraft.net/) protocol**.

If you're having trouble, check out the FAQ or ask for help in the IRC channel #mcdevs on irc.libera.chat (ircs://irc.libera.chat:6697) (More Information (https://wiki.vg/MCDevs)).

The changes between versions may be viewed at Protocol History.

Contents

- World Border Lerp Size
- World Border Size
- World Border Warning Delay
- World Border Warning Reach
- Camera
- Held Item Change (clientbound)
- Update View Position
- Update View Distance
- Spawn Position
- Display Scoreboard
- Entity Metadata
- Attach Entity
- Entity Velocity
- Entity Equipment
- Set Experience
- Update Health
- Scoreboard Objective
- Set Passengers
- Teams
- Update Score
- Update Simulation Distance
- Set Title SubTitle
- Time Update
- Set Title Text
- Set Title Times
- Entity Sound Effect
- Sound Effect
- Stop Sound
- Player List Header And Footer
- NBT Query Response
- Collect Item
- Entity Teleport
- Advancements
- Entity Properties
- Entity Effect
- Declare Recipes
- Tags

Serverbound

- Teleport Confirm
- Query Block NBT
- Set Difficulty
- Chat Message (serverbound)
- Client Status
- Client Settings
- Tab-Complete (serverbound)
- Click Window Button
- Click Window
- Close Window (serverbound)
- Plugin Message (serverbound)
- Edit Book
- Query Entity NBT
- Interact Entity
- Generate Structure
- Keep Alive (serverbound)
- Lock Difficulty
- Player Position
- Player Position And Rotation (serverbound)
- Player Rotation
- Player Movement
- Vehicle Move (serverbound)
- Steer Boat
- Pick Item
- Craft Recipe Request
- Player Abilities (serverbound)
- Player Digging

Definitions

The Minecraft server accepts connections from TCP clients and communicates with them using *packets*. A packet is a sequence of bytes sent over the TCP connection. The meaning of a packet depends both on its packet ID and the current state of the connection. The initial state of each connection is Handshaking, and state is switched using the packets Handshake and Login Success.

Data types

All data sent over the network (except for VarInt and VarLong) is big-endian, that is the bytes are sent from most significant byte to least significant byte. The majority of everyday computers are little-endian, therefore it may be necessary to change the endianness before sending data over the network.

Name	Size (bytes)	Encodes	Notes
Boolean	1	Either false or true	True is encoded as `0x01`, false as `0x00`.
Byte	1	An integer between -128 and 127	Signed 8-bit integer, two's complement
Unsigned Byte	1	An integer between 0 and 255	Unsigned 8-bit integer
Short	2	An integer between -32768 and 32767	Signed 16-bit integer, two's complement
Unsigned Short	2	An integer between 0 and 65535	Unsigned 16-bit integer
Int	4	An integer between -2147483648 and 2147483647	Signed 32-bit integer, two's complement
Long	8	An integer between -9223372036854775808 and 9223372036854775807	Signed 64-bit integer, two's complement
Float	4	A single-precision 32-bit IEEE 754 floating point number	
Double	8	A double-precision 64-bit IEEE 754 floating point number	
String (n)	≥ 1 ≤ (n×4) + 3	A sequence of Unicode scalar values (http://unicode.org/glossary/#unicode_scalar_value)	UTF-8 string prefixed with its size in bytes as a VarInt. Maximum length of n characters, which varies by context; up to n × 4 bytes can be used to encode n characters and both of those limits are checked. Maximum n value is 32767. The + 3 is due to the max size of a valid length VarInt.
Chat	≥ 1 ≤ (262144×4) + 3	See Chat	Encoded as a String with max length of 262144.
Identifier	≥ 1 ≤ (32767×4) + 3	See Identifier below	Encoded as a String with max length of 32767.
VarInt	≥ 1 ≤ 5	An integer between -2147483648 and 2147483647	Variable-length data encoding a two's complement signed 32-bit integer; more info in their section
VarLong	≥ 1 ≤ 10	An integer between -9223372036854775808 and 9223372036854775807	Variable-length data encoding a two's complement signed 64-bit integer; more info in their section
Entity Metadata	Varies	Miscellaneous information about an entity	See Entity_metadata#Entity Metadata Format
Slot	Varies	An item stack in an inventory or container	See Slot Data
NBT Tag	Varies	Depends on context	See NBT
Position	8	An integer/block position: x (-33554432 to 33554431), y (-2048 to 2047), z (-33554432 to 33554431)	x as a 26-bit integer, followed by z as a 26-bit integer, followed by y as a 12-bit integer (all signed, two's complement). See also the section below.
Angle	1	A rotation angle in steps of 1/256 of a full turn	Whether or not this is signed does not matter, since the resulting angles are the same.
UUID	16	A UUID	Encoded as an unsigned 128-bit integer (or two unsigned 64-bit integers: the most significant 64 bits and then the least significant 64 bits)
Optional X	0 or size of X	A field of type X, or nothing	Whether or not the field is present must be known from the context.
Array of X	count times size of X	Zero or more fields of type X	The count must be known from the context.
X Enum	size of X	A specific value from a given list	The list of possible values and how each is encoded as an X must be known from the context. An invalid value sent by either side will usually result in the client being disconnected with an error or even crashing.
Byte	Varies	Depends on context	This is just a sequence of zero or more bytes, its meaning should be explained

Array			somewhere else, e.g. in the packet description. The length must also be known from the context.

Identifier

Identifiers are a namespaced location, in the form of `minecraft:thing`. If the namespace is not provided, it defaults to `minecraft` (i.e. `thing` is `minecraft:thing`. Custom content should always be in its own namespace, not the default one. The namespace should only use the characters `0123456789abcdefghijklmnopqrstuvwxyz-_`; actual names may contain more symbols. The naming convention is `lower_case_with_underscores`. More information (https://minecraft.net/en-us/article/minecraft-snapshot-17w43a).

VarInt and VarLong

Variable-length format such that smaller numbers use fewer bytes. These are very similar to Protocol Buffer Varints (http://developers.google.com/protocol-buffers/docs/encoding#varints): the 7 least significant bits are used to encode the value and the most significant bit indicates whether there's another byte after it for the next part of the number. The least significant group is written first, followed by each of the more significant groups; thus, VarInts are effectively little endian (however, groups are 7 bits, not 8).

VarInts are never longer than 5 bytes, and VarLongs are never longer than 10 bytes.

Pseudocode to read and write VarInts and VarLongs:

```
private static final int SEGMENT_BITS = 0x7F;
private static final int CONTINUE_BIT = 0x80;
```

```
public int readVarInt() {
    int value = 0;
    int position = 0;
    byte currentByte;

    while (true) {
        currentByte = readByte();
        value |= (currentByte & SEGMENT_BITS) << position;

        if ((currentByte & CONTINUE_BIT) == 0) break;

        position += 7;

        if (position >= 32) throw new RuntimeException("VarInt is too big");
    }

    return value;
}
```

```
public long readVarLong() {
    long value = 0;
    int position = 0;
    byte currentByte;

    while (true) {
        currentByte = readByte();
        value |= (currentByte & SEGMENT_BITS) << position;

        if ((currentByte & CONTINUE_BIT) == 0) break;

        position += 7;

        if (position >= 64) throw new RuntimeException("VarLong is too big");
    }

    return value;
}
```

```
public void writeVarInt(int value) {
    while (true) {
        if ((value & ~SEGMENT_BITS) == 0) {
          writeByte(value);
          return;
        }

        writeByte((value & SEGMENT_BITS) | CONTINUE_BIT);
        // Note: >>> means that the sign bit is shifted with the rest of the number rather than being left alone
        value >>>= 7;
    }
}
```

```
public void writeVarLong(long value) {
    while (true) {
```

```
        if ((value & ~((long) SEGMENT_BITS)) == 0) {
          writeByte(value);
          return;
        }

        writeByte((value & SEGMENT_BITS) | CONTINUE_BIT);
        // Note: >>> means that the sign bit is shifted with the rest of the number rather than being left alone
        value >>>= 7;
    }
}
```

⚠ Note Minecraft's VarInts are identical to LEB128 (https://en.wikipedia.org/wiki/LEB128) with the slight change of throwing a exception if it goes over a set amount of bytes.

⚠ Note that Minecraft's VarInts are not encoded using Protocol Buffers; it's just similar. If you try to use Protocol Buffers Varints with Minecraft's VarInts, you'll get incorrect results in some cases. The major differences:

- Minecraft's VarInts are all signed, but do not use the ZigZag encoding. Protocol buffers have 3 types of Varints: `uint32` (normal encoding, unsigned), `sint32` (ZigZag encoding, signed), and `int32` (normal encoding, signed). Minecraft's are the `int32` variety. Because Minecraft uses the normal encoding instead of ZigZag encoding, negative values always use the maximum number of bytes.
- Minecraft's VarInts are never longer than 5 bytes and its VarLongs will never be longer than 10 bytes, while Protocol Buffer Varints will always use 10 bytes when encoding negative numbers, even if it's an `int32`.

Sample VarInts:

Value	Hex bytes	Decimal bytes
0	0x00	0
1	0x01	1
2	0x02	2
127	0x7f	127
128	0x80 0x01	128 1
255	0xff 0x01	255 1
25565	0xdd 0xc7 0x01	221 199 1
2097151	0xff 0xff 0x7f	255 255 127
2147483647	0xff 0xff 0xff 0xff 0x07	255 255 255 255 7
-1	0xff 0xff 0xff 0xff 0x0f	255 255 255 255 15
-2147483648	0x80 0x80 0x80 0x80 0x08	128 128 128 128 8

Sample VarLongs:

Value	Hex bytes	Decimal bytes
0	0x00	0
1	0x01	1
2	0x02	2
127	0x7f	127
128	0x80 0x01	128 1
255	0xff 0x01	255 1
2147483647	0xff 0xff 0xff 0xff 0x07	255 255 255 255 7
9223372036854775807	0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0x7f	255 255 255 255 255 255 255 255 127
-1	0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0xff 0x01	255 255 255 255 255 255 255 255 255 1
-2147483648	0x80 0x80 0x80 0x80 0xf8 0xff 0xff 0xff 0xff 0x01	128 128 128 128 248 255 255 255 255 1
-9223372036854775808	0x80 0x80 0x80 0x80 0x80 0x80 0x80 0x80 0x80 0x01	128 128 128 128 128 128 128 128 128 1

Position

Note: What you are seeing here is the latest version of the Data types article, but the position type was different before 1.14 (https://wiki.vg/index.php?title=Data_types&oldid=14345#Position).

64-bit value split in to three parts

- x: 26 MSBs

- z: 26 middle bits
- y: 12 LSBs

Encoded as followed:

```
((x & 0x3FFFFFF) << 38) | ((z & 0x3FFFFFF) << 12) | (y & 0xFFF)
```

And decoded as:

```
val = read_unsigned_long();
x = val >> 38;
y = val & 0xFFF;
z = (val >> 12) & 0x3FFFFFF;
```

Note: The details of bit shifting are rather language dependent; the above may work in Java but probably won't in other languages without some tweaking. In particular, you will usually receive positive numbers even if the actual coordinates are negative. This can be fixed by adding something like the following:

```
if x >= 2^25 { x -= 2^26 }
if y >= 2^11 { y -= 2^12 }
if z >= 2^25 { z -= 2^26 }
```

Fixed-point numbers

Some fields may be stored as fixed-point numbers (https://en.wikipedia.org/wiki/Fixed-point_arithmetic), where a certain number of bits represents the signed integer part (number to the left of the decimal point) and the rest represents the fractional part (to the right). Floating points (float and double), in contrast, keep the number itself (mantissa) in one chunk, while the location of the decimal point (exponent) is stored beside it.

Essentially, while fixed-point numbers have lower range than floating points, their fractional precision is greater for higher values. This makes them ideal for representing global coordinates of an entity in Minecraft, as it's more important to store the integer part accurately than position them more precisely within a single block (or meter).

Coordinates are often represented as a 32-bit integer, where 5 of the least-significant bits are dedicated to the fractional part, and the rest store the integer part.

Java lacks support for fractional integers directly, but you can represent them as integers. To convert from a double to this integer representation, use the following formulas:

```
abs_int = (int) (double * 32.0D);
```

And back again:

```
double = (double) (abs_int / 32.0D);
```

Particle

Field Name	Field Type	Meaning
ID	VarInt	The ID of the particle type, see below.
Data	Varies	Varies based on the particle type, see below.

<table>
<tr><th>Particle Name</th><th>Particle ID</th><th>Data</th></tr>
<tr><td>minecraft:ambient_entity_effect</td><td>0</td><td>None</td></tr>
<tr><td>minecraft:angry_villager</td><td>1</td><td>None</td></tr>
<tr><td>minecraft:block</td><td>2</td><td><table><tr><th>Field Name</th><th>Field Type</th><th>Meaning</th></tr><tr><td>BlockState</td><td>VarInt</td><td>The ID of the block state.</td></tr></table></td></tr>
<tr><td>minecraft:block_marker</td><td>3</td><td><table><tr><th>Field Name</th><th>Field Type</th><th>Meaning</th></tr><tr><td>BlockState</td><td>VarInt</td><td>The ID of the block state.</td></tr></table></td></tr>
<tr><td>minecraft:bubble</td><td>4</td><td>None</td></tr>
<tr><td>minecraft:cloud</td><td>5</td><td>None</td></tr>
<tr><td>minecraft:crit</td><td>6</td><td>None</td></tr>
<tr><td>minecraft:damage_indicator</td><td>7</td><td>None</td></tr>
<tr><td>minecraft:dragon_breath</td><td>8</td><td>None</td></tr>
<tr><td>minecraft:dripping_lava</td><td>9</td><td>None</td></tr>
<tr><td>minecraft:falling_lava</td><td>10</td><td>None</td></tr>
<tr><td>minecraft:landing_lava</td><td>11</td><td>None</td></tr>
<tr><td>minecraft:dripping_water</td><td>12</td><td>None</td></tr>
<tr><td>minecraft:falling_water</td><td>13</td><td>None</td></tr>
<tr><td>minecraft:dust</td><td>14</td><td><table><tr><th>Field Name</th><th>Field Type</th><th>Meaning</th></tr><tr><td>Red</td><td>Float</td><td>Red value, 0-1</td></tr><tr><td>Green</td><td>Float</td><td>Green value, 0-1</td></tr><tr><td>Blue</td><td>Float</td><td>Blue value, 0-1</td></tr><tr><td>Scale</td><td>Float</td><td>The scale, will be clamped between 0.01 and 4.</td></tr></table></td></tr>
<tr><td>minecraft:dust_color_transition</td><td>15</td><td><table><tr><th>Field Name</th><th>Field Type</th><th>Meaning</th></tr><tr><td>FromRed</td><td>Float</td><td>Red value, 0-1</td></tr><tr><td>FromGreen</td><td>Float</td><td>Green value, 0-1</td></tr><tr><td>FromBlue</td><td>Float</td><td>Blue value, 0-1</td></tr><tr><td>Scale</td><td>Float</td><td>The scale, will be clamped between 0.01 and 4.</td></tr><tr><td>ToRed</td><td>Float</td><td>Red value, 0-1</td></tr><tr><td>ToGreen</td><td>Float</td><td>Green value, 0-1</td></tr><tr><td>ToBlue</td><td>Float</td><td>Blue value, 0-1</td></tr></table></td></tr>
<tr><td>minecraft:effect</td><td>16</td><td>None</td></tr>
<tr><td>minecraft:elder_guardian</td><td>17</td><td>None</td></tr>
<tr><td>minecraft:enchanted_hit</td><td>18</td><td>None</td></tr>
<tr><td>minecraft:enchant</td><td>19</td><td>None</td></tr>
<tr><td>minecraft:end_rod</td><td>20</td><td>None</td></tr>
<tr><td>minecraft:entity_effect</td><td>21</td><td>None</td></tr>
<tr><td>minecraft:explosion_emitter</td><td>22</td><td>None</td></tr>
<tr><td>minecraft:explosion</td><td>23</td><td>None</td></tr>
<tr><td>minecraft:falling_dust</td><td>24</td><td><table><tr><th>Field Name</th><th>Field Type</th><th>Meaning</th></tr><tr><td>BlockState</td><td>VarInt</td><td>The ID of the block state.</td></tr></table></td></tr>
</table>

<table>
<tr><td>minecraft:firework</td><td>25</td><td>None</td></tr>
<tr><td>minecraft:fishing</td><td>26</td><td>None</td></tr>
<tr><td>minecraft:flame</td><td>27</td><td>None</td></tr>
<tr><td>minecraft:soul_fire_flame</td><td>28</td><td>None</td></tr>
<tr><td>minecraft:soul</td><td>29</td><td>None</td></tr>
<tr><td>minecraft:flash</td><td>30</td><td>None</td></tr>
<tr><td>minecraft:happy_villager</td><td>31</td><td>None</td></tr>
<tr><td>minecraft:composter</td><td>32</td><td>None</td></tr>
<tr><td>minecraft:heart</td><td>33</td><td>None</td></tr>
<tr><td>minecraft:instant_effect</td><td>34</td><td>None</td></tr>
<tr><td>minecraft:item</td><td>35</td><td><table><tr><th>Field Name</th><th>Field Type</th><th>Meaning</th></tr><tr><td>Item</td><td>Slot</td><td>The item that will be used.</td></tr></table></td></tr>
<tr><td>minecraft:vibration</td><td>36</td><td><table><tr><th>Field Name</th><th>Field Type</th><th>Meaning</th></tr><tr><td>Origin</td><td>Position</td><td>Starting position</td></tr><tr><td>PositionType</td><td>String</td><td>Type of destination</td></tr><tr><td>BlockPosition</td><td>Position</td><td>Present if PositionType is "minecraft:block"</td></tr><tr><td>EntityId</td><td>Varint</td><td>Present if PositionType is "minecraft:entity"</td></tr><tr><td>Ticks</td><td>Varint</td><td></td></tr></table></td></tr>
<tr><td>minecraft:item_slime</td><td>37</td><td>None</td></tr>
<tr><td>minecraft:item_snowball</td><td>38</td><td>None</td></tr>
<tr><td>minecraft:large_smoke</td><td>39</td><td>None</td></tr>
<tr><td>minecraft:lava</td><td>40</td><td>None</td></tr>
<tr><td>minecraft:mycelium</td><td>41</td><td>None</td></tr>
<tr><td>minecraft:note</td><td>42</td><td>None</td></tr>
<tr><td>minecraft:poof</td><td>43</td><td>None</td></tr>
<tr><td>minecraft:portal</td><td>44</td><td>None</td></tr>
<tr><td>minecraft:rain</td><td>45</td><td>None</td></tr>
<tr><td>minecraft:smoke</td><td>46</td><td>None</td></tr>
<tr><td>minecraft:sneeze</td><td>47</td><td>None</td></tr>
<tr><td>minecraft:spit</td><td>48</td><td>None</td></tr>
<tr><td>minecraft:squid_ink</td><td>49</td><td>None</td></tr>
<tr><td>minecraft:sweep_attack</td><td>50</td><td>None</td></tr>
<tr><td>minecraft:totem_of_undying</td><td>51</td><td>None</td></tr>
<tr><td>minecraft:underwater</td><td>52</td><td>None</td></tr>
<tr><td>minecraft:splash</td><td>53</td><td>None</td></tr>
<tr><td>minecraft:witch</td><td>54</td><td>None</td></tr>
<tr><td>minecraft:bubble_pop</td><td>55</td><td>None</td></tr>
<tr><td>minecraft:current_down</td><td>56</td><td>None</td></tr>
<tr><td>minecraft:bubble_column_up</td><td>57</td><td>None</td></tr>
<tr><td>minecraft:nautilus</td><td>58</td><td>None</td></tr>
<tr><td>minecraft:dolphin</td><td>59</td><td>None</td></tr>
<tr><td>minecraft:campfire_cosy_smoke</td><td>60</td><td>None</td></tr>
<tr><td>minecraft:campfire_signal_smoke</td><td>61</td><td>None</td></tr>
<tr><td>minecraft:dripping_honey</td><td>62</td><td>None</td></tr>
<tr><td>minecraft:falling_honey</td><td>63</td><td>None</td></tr>
</table>

minecraft:landing_honey	64	None
minecraft:falling_nectar	65	None
minecraft:falling_spore_blossom	66	None
minecraft:ash	67	None
minecraft:crimson_spore	68	None
minecraft:warped_spore	69	None
minecraft:spore_blossom_air	70	None
minecraft:dripping_obsidian_tear	71	None
minecraft:falling_obsidian_tear	72	None
minecraft:landing_obsidian_tear	73	None
minecraft:reverse_portal	74	None
minecraft:white_ash	75	None
minecraft:small_flame	76	None
minecraft:snowflake	77	None
minecraft:dripping_dripstone_lava	78	None
minecraft:falling_dripstone_lava	79	None
minecraft:dripping_dripstone_water	80	None
minecraft:falling_dripstone_water	81	None
minecraft:glow_squid_ink	82	None
minecraft:glow	83	None
minecraft:wax_on	84	None
minecraft:wax_off	85	None
minecraft:electric_spark	86	None
minecraft:scrape	87	None

BitSet

Represents Java's `BitSet` (https://docs.oracle.com/javase/8/docs/api/java/util/BitSet.html), a list of bits.

Field Name	Field Type	Meaning
Length	VarInt	Number of longs in the following array. May be 0 (if no bits are set).
Data	Array of Long	A packed representation of the BitSet as created by `BitSet.toLongArray` (https://docs.oracle.com/javase/8/docs/api/java/util/BitSet.html#toLongArray--).

The *n*th bit in a BitSet is set when `(longs[n/64] & (1L << (n % 64))) != 0`, where *n* starts at 0.

Other definitions

Term	Definition
Player	When used in the singular, Player always refers to the client connected to the server.
Entity	Entity refers to any item, player, mob, minecart or boat etc. See the Minecraft Wiki article (https://minecraft.fandom.com/wiki/Entity) for a full list.
EID	An EID — or Entity ID — is a 4-byte sequence used to identify a specific entity. An entity's EID is unique on the entire server.
XYZ	In this document, the axis names are the same as those shown in the debug screen (F3). Y points upwards, X points east, and Z points south.
Meter	The meter is Minecraft's base unit of length, equal to the length of a vertex of a solid block. The term “block” may be used to mean “meter” or “cubic meter”.
Global palette	A table/dictionary/palette mapping nonnegative integers to block states. Block state IDs are created in a linear fashion based off of order of assignment. One block state ID allocated for each unique block state for a block; if a block has multiple properties then the number of allocated states is the product of the number of values for each property. A current list of properties and state ID ranges is found on burger (https://pokechu22.github.io/Burger/1.16.5.html). Alternatively, the vanilla server now includes an option to export the current block state ID mapping, by running `java -cp minecraft_server.jar net.minecraft.data.Main --reports`. See Data Generators for more information.
Notchian	The official implementation of vanilla Minecraft as developed and released by Mojang.

Packet format

Packets cannot be larger than 2097151 bytes (the maximum that can be sent in a 3-byte VarInt). For compressed packets, this applies to both the compressed length and uncompressed lengths.

Without compression

Field Name	Field Type	Notes
Length	VarInt	Length of Packet ID + Data
Packet ID	VarInt	
Data	Byte Array	Depends on the connection state and packet ID, see the sections below

With compression

Once a Set Compression packet (with a non-negative threshold) is sent, zlib compression is enabled for all following packets. The format of a packet changes slightly to include the size of the uncompressed packet.

Compressed?	Field Name	Field Type	Notes
No	Packet Length	VarInt	Length of Data Length + compressed length of (Packet ID + Data)
No	Data Length	VarInt	Length of uncompressed (Packet ID + Data) or 0
Yes	Packet ID	VarInt	zlib compressed packet ID (see the sections below)
	Data	Byte Array	zlib compressed packet data (see the sections below)

If the size of the buffer containing the packet data and ID (as a VarInt) is smaller than the threshold specified in the packet Set Compression. It will be sent as uncompressed. This is done by setting the data length as 0. (Comparable to sending a non-compressed format with an extra 0 between the length, and packet data).

If it's larger than the threshold, then it follows the regular compressed protocol format.

Compression can be disabled by sending the packet Set Compression with a negative Threshold, or not sending the Set Compression packet at all.

Handshaking

Clientbound

There are no clientbound packets in the Handshaking state, since the protocol immediately switches to a different state after the client sends the first packet.

Serverbound

Handshake

This causes the server to switch into the target state.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x00	Handshaking	Server	Protocol Version	VarInt	See protocol version numbers (currently 758 in Minecraft 1.18.2).
			Server Address	String (255)	Hostname or IP, e.g. localhost or 127.0.0.1, that was used to connect. The Notchian server does not use this information. Note that SRV records are a simple redirect, e.g. if _minecraft._tcp.example.com points to mc.example.org, users connecting to example.com will provide example.org as server address in addition to connecting to it.
			Server Port	Unsigned Short	Default is 25565. The Notchian server does not use this information.
			Next State	VarInt Enum	1 for status, 2 for login.

Legacy Server List Ping

⚠ This packet uses a nonstandard format. It is never length-prefixed, and the packet ID is an Unsigned Byte instead of a VarInt.

While not technically part of the current protocol, legacy clients may send this packet to initiate Server List Ping, and modern servers should handle it correctly.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0xFE	Handshaking	Server	Payload	Unsigned Byte	always 1 (`0x01`).

See Server List Ping#1.6 for the details of the protocol that follows this packet.

Status

Main article: Server List Ping

Clientbound

Response

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x00	Status	Client	JSON Response	String (32767)	See Server List Ping#Response; as with all strings this is prefixed by its length as a VarInt.

Pong

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x01	Status	Client	Payload	Long	Should be the same as sent by the client.

Serverbound

Request

The status can only be requested once immediately after the handshake, before any ping. The server won't respond otherwise.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x00	Status	Server	*no fields*		

Ping

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x01	Status	Server	Payload	Long	May be any number. Notchian clients use a system-dependent time value which is counted in milliseconds.

Login

The login process is as follows:

1. C→S: Handshake with Next State set to 2 (login)
2. C→S: Login Start
3. S→C: Encryption Request
4. Client auth
5. C→S: Encryption Response
6. Server auth, both enable encryption
7. S→C: Set Compression (optional)
8. S→C: Login Success

Set Compression, if present, must be sent before Login Success. Note that anything sent after Set Compression must use the Post Compression packet format.

For unauthenticated ("cracked"/offline-mode) and integrated servers (either of the two conditions is enough for an unencrypted connection) there is no encryption. In that case Login Start is directly followed by Login Success. The Notchian server uses UUID v3 for offline player UUIDs, with the namespace “OfflinePlayer” and the value as the player’s username. For example, Notch’s offline UUID would be derived from the string “OfflinePlayer:Notch”. This is not a requirement however, the UUID may be anything.

See Protocol Encryption for details.

Clientbound

Disconnect (login)

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x00	Login	Client	Reason	Chat	

Encryption Request

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x01	Login	Client	Server ID	String (20)	Appears to be empty.
			Public Key Length	VarInt	Length of Public Key
			Public Key	Byte Array	The server's public key in bytes
			Verify Token Length	VarInt	Length of Verify Token. Always 4 for Notchian servers.
			Verify Token	Byte Array	A sequence of random bytes generated by the server.

See Protocol Encryption for details.

Login Success

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x02	Login	Client	UUID	UUID	
			Username	String (16)	

This packet switches the connection state to play.

⚠ The (notchian) server might take a bit to fully transition to the Play state, so it's recommended to wait before sending Play packets, either by setting a timeout, or waiting for Play packets from the server (usually Player Info).

The notchian client doesn't send any (non-keep alive) packets until the next tick/time update packet.

Set Compression

Enables compression. If compression is enabled, all following packets are encoded in the compressed packet format. Negative or zero values will disable compression, meaning the packet format should remain in the uncompressed packet format. However, this packet is entirely optional, and if not sent, compression will also not be enabled (the notchian server does not send the packet when compression is disabled).

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x03	Login	Client	Threshold	VarInt	Maximum size of a packet before it is compressed.

Login Plugin Request

Used to implement a custom handshaking flow together with Login Plugin Response.

Unlike plugin messages in "play" mode, these messages follow a lock-step request/response scheme, where the client is expected to respond to a request indicating whether it understood. The notchian client always responds that it hasn't understood, and sends an empty payload.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x04	Login	Client	Message ID	VarInt	Generated by the server - should be unique to the connection.
			Channel	Identifier	Name of the plugin channel used to send the data.
			Data	Byte Array	Any data, depending on the channel. The length of this array must be inferred from the packet length.

Serverbound

Login Start

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x00	Login	Server	Name	String (16)	Player's Username.

Encryption Response

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x01	Login	Server	Shared Secret Length	VarInt	Length of Shared Secret.
			Shared Secret	Byte Array	Shared Secret value, encrypted with the server's public key.
			Verify Token Length	VarInt	Length of Verify Token.
			Verify Token	Byte Array	Verify Token value, encrypted with the same public key as the shared secret.

See Protocol Encryption for details.

Login Plugin Response

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x02	Login	Server	Message ID	VarInt	Should match ID from server.
			Successful	Boolean	`true` if the client understands the request, `false` otherwise. When `false`, no payload follows.
			Data	Optional Byte Array	Any data, depending on the channel. The length of this array must be inferred from the packet length.

Play

Clientbound

Spawn Entity

Sent by the server when a vehicle or other non-living entity is created.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x00	Play	Client	Entity ID	VarInt	Entity ID.
			Object UUID	UUID	
			Type	VarInt	The type of entity (same as in Spawn Living Entity).
			X	Double	
			Y	Double	
			Z	Double	
			Pitch	Angle	To get the real pitch, you must divide this by (256.0F / 360.0F)
			Yaw	Angle	To get the real yaw, you must divide this by (256.0F / 360.0F)
			Data	Int	Meaning dependent on the value of the Type field, see Object Data for details.
			Velocity X	Short	Same units as Entity Velocity. Always sent, but only used when Data is greater than 0 (except for some entities which always ignore it; see Object Data for details).
			Velocity Y	Short	
			Velocity Z	Short	

Spawn Experience Orb

Spawns one or more experience orbs.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x01	Play	Client	Entity ID	VarInt	
			X	Double	
			Y	Double	
			Z	Double	
			Count	Short	The amount of experience this orb will reward once collected.

Spawn Living Entity

Sent by the server when a living entity is spawned.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x02	Play	Client	Entity ID	VarInt	
			Entity UUID	UUID	
			Type	VarInt	The type of mob. See Entity_metadata#Mobs.
			X	Double	
			Y	Double	
			Z	Double	
			Yaw	Angle	
			Pitch	Angle	
			Head Yaw	Angle	
			Velocity X	Short	Same units as Entity Velocity.
			Velocity Y	Short	Same units as Entity Velocity.
			Velocity Z	Short	Same units as Entity Velocity.

Spawn Painting

This packet shows location, name, and type of painting.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x03	Play	Client	Entity ID	VarInt	
			Entity UUID	UUID	
			Motive	VarInt	Painting's ID, see below.
			Location	Position	Center coordinates (see below).
			Direction	Byte Enum	Direction the painting faces (North = 2, South = 0, West = 1, East = 3).

Calculating the center of an image: given a (width × height) grid of cells, with `(0, 0)` being the top left corner, the center is `(max(0, width / 2 - 1), height / 2)`. E.g. `(1, 0)` for a 2×1 painting, or `(1, 2)` for a 4×4 painting.

List of paintings by coordinates in `paintings_kristoffer_zetterstrand.png` (where x and y are in pixels from the top left and width and height are in pixels or 16ths of a block):

Name	ID	x	y	width	height
`minecraft:kebab`	0	0	0	16	16
`minecraft:aztec`	1	16	0	16	16
`minecraft:alban`	2	32	0	16	16
`minecraft:aztec2`	3	48	0	16	16
`minecraft:bomb`	4	64	0	16	16
`minecraft:plant`	5	80	0	16	16
`minecraft:wasteland`	6	96	0	16	16
`minecraft:pool`	7	0	32	32	16
`minecraft:courbet`	8	32	32	32	16
`minecraft:sea`	9	64	32	32	16
`minecraft:sunset`	10	96	32	32	16
`minecraft:creebet`	11	128	32	32	16
`minecraft:wanderer`	12	0	64	16	32
`minecraft:graham`	13	16	64	16	32
`minecraft:match`	14	0	128	32	32
`minecraft:bust`	15	32	128	32	32
`minecraft:stage`	16	64	128	32	32
`minecraft:void`	17	96	128	32	32
`skull_and_roses`	18	128	128	32	32
`minecraft:wither`	19	160	128	32	32
`minecraft:fighters`	20	0	96	64	32
`minecraft:pointer`	21	0	192	64	64
`minecraft:pigscene`	22	64	192	64	64
`minecraft:burning_skull`	23	128	192	64	64
`minecraft:skeleton`	24	192	64	64	48
`minecraft:donkey_kong`	25	192	112	64	48

The Minecraft Wiki article on paintings (https://minecraft.fandom.com/wiki/Painting%23Canvases) also provides a list of painting names to the actual images.

Spawn Player

This packet is sent by the server when a player comes into visible range, *not* when a player joins.

This packet must be sent after the Player Info packet that adds the player data for the client to use when spawning a player. If the Player Info for the player spawned by this packet is not present when this packet arrives, Notchian clients will not spawn the player entity. The Player Info packet includes skin/cape data.

Servers can, however, safely spawn player entities for players not in visible range. The client appears to handle it correctly.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x04	Play	Client	Entity ID	VarInt	Player's EID.
			Player UUID	UUID	See below for notes on offline mode (https://minecraft.fandom.com/wiki/Server.properties%23online-mode) and NPCs.
			X	Double	
			Y	Double	
			Z	Double	
			Yaw	Angle	
			Pitch	Angle	

When in online mode (https://minecraft.fandom.com/wiki/Server.properties%23online-mode), the UUIDs must be valid and have valid skin blobs.

In offline mode, UUID v3 is used with the String `OfflinePlayer:<player name>`, encoded in UTF-8 (and case-sensitive). The Notchain server uses `UUID.nameUUIDFromBytes`, implemented by OpenJDK here (https://github.com/AdoptOpenJDK/openjdk-jdk8u/blob/9a91972c76ddda5c1ce28b50ca38cbd8a30b7a72/jdk/src/share/classes/java/util/UUID.java#L153-L175).

For NPCs UUID v2 should be used. Note:

```
<+Grum> i will never confirm this as a feature you know that :)
```

In an example UUID, `xxxxxxxx-xxxx-Yxxx-xxxx-xxxxxxxxxxxx`, the UUID version is specified by `Y`. So, for UUID v3, `Y` will always be `3`, and for UUID v2, `Y` will always be `2`.

Sculk Vibration Signal

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x05	Play	Client	Source Position	Position	Source position for the vibration.
			Destination Identifier	Identifier	Identifier of the destination codec type.
			Destination	Varies	Vanilla default destinations are a block position encoded as a Position (https://wiki.vg/Protocol#Data_types) for "block" or an entity id encoded as a VarInt for "entity".
			Arrival Ticks	VarInt	Ticks for the signal to arrive at the destination.

This packet shows a permanent particle.

Entity Animation (clientbound)

Sent whenever an entity should change animation.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x06	Play	Client	Entity ID	VarInt	Player ID.
			Animation	Unsigned Byte	Animation ID (see below).

Animation can be one of the following values:

ID	Animation
0	Swing main arm
1	Take damage
2	Leave bed
3	Swing offhand
4	Critical effect
5	Magic critical effect

Statistics

Sent as a response to Client Status 0x04 (id 1). Will only send the changed values if previously requested.

Packet ID	State	Bound To	Field Name		Field Type		Notes
0x07	Play	Client	Count		VarInt		Number of elements in the following array.
			Statistic	Category ID	Array	VarInt	See below.
				Statistic ID		VarInt	See below.
				Value		VarInt	The amount to set it to.

Categories (these are namespaced, but with `:` replaced with `.`):

Name	ID	Registry
`minecraft.mined`	0	Blocks
`minecraft.crafted`	1	Items
`minecraft.used`	2	Items
`minecraft.broken`	3	Items
`minecraft.picked_up`	4	Items
`minecraft.dropped`	5	Items
`minecraft.killed`	6	Entities
`minecraft.killed_by`	7	Entities
`minecraft.custom`	8	Custom

Blocks, Items, and Entities use block (not block state), item, and entity ids.

Custom has the following (unit only matters for clients):

Name	ID	Unit
minecraft.leave_game	0	None
minecraft.play_one_minute	1	Time
minecraft.time_since_death	2	Time
minecraft.time_since_rest	3	Time
minecraft.sneak_time	4	Time
minecraft.walk_one_cm	5	Distance
minecraft.crouch_one_cm	6	Distance
minecraft.sprint_one_cm	7	Distance
minecraft.walk_on_water_one_cm	8	Distance
minecraft.fall_one_cm	9	Distance
minecraft.climb_one_cm	10	Distance
minecraft.fly_one_cm	11	Distance
minecraft.walk_under_water_one_cm	12	Distance
minecraft.minecart_one_cm	13	Distance
minecraft.boat_one_cm	14	Distance
minecraft.pig_one_cm	15	Distance
minecraft.horse_one_cm	16	Distance
minecraft.aviate_one_cm	17	Distance
minecraft.swim_one_cm	18	Distance
minecraft.strider_one_cm	19	Distance
minecraft.jump	20	None
minecraft.drop	21	None
minecraft.damage_dealt	22	Damage
minecraft.damage_dealt_absorbed	23	Damage
minecraft.damage_dealt_resisted	24	Damage
minecraft.damage_taken	25	Damage
minecraft.damage_blocked_by_shield	26	Damage
minecraft.damage_absorbed	27	Damage
minecraft.damage_resisted	28	Damage
minecraft.deaths	29	None
minecraft.mob_kills	30	None
minecraft.animals_bred	31	None
minecraft.player_kills	32	None
minecraft.fish_caught	33	None
minecraft.talked_to_villager	34	None
minecraft.traded_with_villager	35	None
minecraft.eat_cake_slice	36	None
minecraft.fill_cauldron	37	None
minecraft.use_cauldron	38	None
minecraft.clean_armor	39	None
minecraft.clean_banner	40	None
minecraft.clean_shulker_box	41	None
minecraft.interact_with_brewingstand	42	None
minecraft.interact_with_beacon	43	None
minecraft.inspect_dropper	44	None
minecraft.inspect_hopper	45	None
minecraft.inspect_dispenser	46	None

minecraft.play_noteblock	47	None
minecraft.tune_noteblock	48	None
minecraft.pot_flower	49	None
minecraft.trigger_trapped_chest	50	None
minecraft.open_enderchest	51	None
minecraft.enchant_item	52	None
minecraft.play_record	53	None
minecraft.interact_with_furnace	54	None
minecraft.interact_with_crafting_table	55	None
minecraft.open_chest	56	None
minecraft.sleep_in_bed	57	None
minecraft.open_shulker_box	58	None
minecraft.open_barrel	59	None
minecraft.interact_with_blast_furnace	60	None
minecraft.interact_with_smoker	61	None
minecraft.interact_with_lectern	62	None
minecraft.interact_with_campfire	63	None
minecraft.interact_with_cartography_table	64	None
minecraft.interact_with_loom	65	None
minecraft.interact_with_stonecutter	66	None
minecraft.bell_ring	67	None
minecraft.raid_trigger	68	None
minecraft.raid_win	69	None
minecraft.interact_with_anvil	70	None
minecraft.interact_with_grindstone	71	None
minecraft.target_hit	72	None
minecraft.interact_with_smithing_table	73	None

Units:

- None: just a normal number (formatted with 0 decimal places)
- Damage: value is 10 times the normal amount
- Distance: a distance in centimeters (hundredths of blocks)
- Time: a time span in ticks

Acknowledge Player Digging

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x08	Play	Client	Location	Position	Position where the digging was happening.
			Block	VarInt	Block state ID of the block that should be at that position now.
			Status	VarInt enum	Same as Player Digging. Only Started digging (0), Cancelled digging (1), and Finished digging (2) are used.
			Successful	Boolean	True if the digging succeeded; false if the client should undo any changes it made locally.

Block Break Animation

0–9 are the displayable destroy stages and each other number means that there is no animation on this coordinate.

Block break animations can still be applied on air; the animation will remain visible although there is no block being broken. However, if this is applied to a transparent block, odd graphical effects may happen, including water losing its transparency. (An effect similar to this can be seen in normal gameplay when breaking ice blocks)

If you need to display several break animations at the same time you have to give each of them a unique Entity ID. The entity ID does not need to correspond to an actual entity on the client. It is valid to use a randomly generated number.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x09	Play	Client	Entity ID	VarInt	Entity ID of the entity breaking the block.
			Location	Position	Block Position.
			Destroy Stage	Byte	0–9 to set it, any other value to remove it.

Block Entity Data

Sets the block entity associated with the block at the given location.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x0A	Play	Client	Location	Position	
			Type	VarInt	The type of the block entity
			NBT Data	NBT Tag	Data to set. May be a TAG_END (0), in which case the block entity at the given location is removed (though this is not required since the client will remove the block entity automatically on chunk unload or block removal).

Block Action

This packet is used for a number of actions and animations performed by blocks, usually non-persistent. The client ignores the provided block type and instead uses the block state in their world.

See Block Actions for a list of values.

⚠ This packet uses a block ID, not a block state.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x0B	Play	Client	Location	Position	Block coordinates.
			Action ID (Byte 1)	Unsigned Byte	Varies depending on block — see Block Actions.
			Action Param (Byte 2)	Unsigned Byte	Varies depending on block — see Block Actions.
			Block Type	VarInt	The block type ID for the block. This must match the block at the given coordinates.

Block Change

Fired whenever a block is changed within the render distance.

⚠ Changing a block in a chunk that is not loaded is not a stable action. The Notchian client currently uses a *shared* empty chunk which is modified for all block changes in unloaded chunks; while in 1.9 this chunk never renders in older versions the changed block will appear in all copies of the empty chunk. Servers should avoid sending block changes in unloaded chunks and clients should ignore such packets.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x0C	Play	Client	Location	Position	Block Coordinates.
			Block ID	VarInt	The new block state ID for the block as given in the global palette (https://minecraft.fandom.com/wiki/Data_values%23Block_IDs). See that section for more information.

Boss Bar

Packet ID	State	Bound To	Field Name		Field Type	Notes
0x0D	Play	Client	UUID		UUID	Unique ID for this bar.
			Action		VarInt Enum	Determines the layout of the remaining packet.
			Action	**Field Name**		
			0: add	Title	Chat	
				Health	Float	From 0 to 1. Values greater than 1 do not crash a Notchian client, and start rendering part of a second health bar (https://i.johni0702.de/nA.png) at around 1.5.
				Color	VarInt Enum	Color ID (see below).
				Division	VarInt Enum	Type of division (see below).
				Flags	Unsigned Byte	Bit mask. 0x1: should darken sky, 0x2: is dragon bar (used to play end music), 0x04: create fog (previously was also controlled by 0x02).
			1: remove	*no fields*	*no fields*	Removes this boss bar.
			2: update health	Health	Float	*as above*
			3: update title	Title	Chat	
			4: update style	Color	VarInt Enum	Color ID (see below).
				Dividers	VarInt Enum	*as above*
			5: update flags	Flags	Unsigned Byte	*as above*

ID	Color
0	Pink
1	Blue
2	Red
3	Green
4	Yellow
5	Purple
6	White

ID	Type of division
0	No division
1	6 notches
2	10 notches
3	12 notches
4	20 notches

Server Difficulty

Changes the difficulty setting in the client's option menu

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x0E	Play	Client	Difficulty	Unsigned Byte	0: peaceful, 1: easy, 2: normal, 3: hard.
			Difficulty locked?	Boolean	

Chat Message (clientbound)

Identifying the difference between Chat/System Message is important as it helps respect the user's chat visibility options. See processing chat for more info about these positions.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x0F	Play	Client	JSON Data	Chat	Limited to 262144 bytes.
			Position	Byte	0: chat (chat box), 1: system message (chat box), 2: game info (above hotbar).
			Sender	UUID	Used by the Notchian client for the disableChat launch option. Setting both longs to 0 will always display the message regardless of the setting.

Clear Titles

Clear the client's current title information, with the option to also reset it.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x10	Play	Client	Reset	Boolean	

Tab-Complete (clientbound)

The server responds with a list of auto-completions of the last word sent to it. In the case of regular chat, this is a player username. Command names and parameters are also supported. The client sorts these alphabetically before listing them.

Packet ID	State	Bound To	Field Name		Field Type		Notes
0x11	Play	Client	ID		VarInt		Transaction ID.
			Start		VarInt		Start of the text to replace.
			Length		VarInt		Length of the text to replace.
			Count		VarInt		Number of elements in the following array.
			Matches	Match	Array	String (32767)	One eligible value to insert, note that each command is sent separately instead of in a single string, hence the need for Count. Note that for instance this doesn't include a leading / on commands.
				Has tooltip		Boolean	True if the following is present.
				Tooltip		Optional Chat	Tooltip to display; only present if previous boolean is true.

Declare Commands

Lists all of the commands on the server, and how they are parsed.

This is a directed graph, with one root node. Each redirect or child node must refer only to nodes that have already been declared.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x12	Play	Client	Count	VarInt	Number of elements in the following array.
			Nodes	Array of Node	An array of nodes.
			Root index	VarInt	Index of the `root` node in the previous array.

For more information on this packet, see the Command Data article.

Close Window (clientbound)

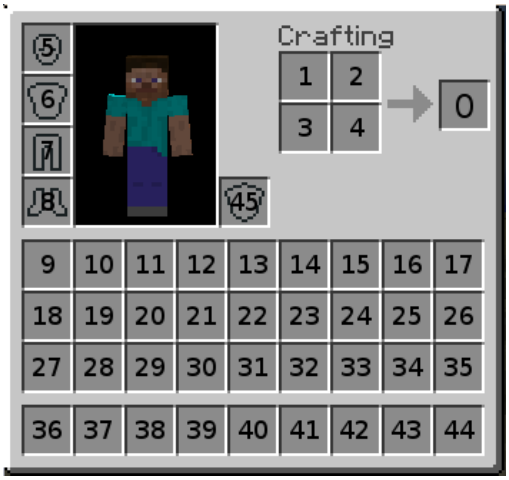
This packet is sent from the server to the client when a window is forcibly closed, such as when a chest is destroyed while it's open. The notchian client disregards the provided window ID and closes any active window.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x13	Play	Client	Window ID	Unsigned Byte	This is the ID of the window that was closed. 0 for inventory.

Window Items

Sent by the server when items in multiple slots (in a window) are added/removed. This includes the main inventory, equipped armour and crafting slots. This packet with Window ID set to "0" is sent during the player joining sequence to initialise the player's inventory.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x14	Play	Client	Window ID	Unsigned Byte	The ID of window which items are being sent for. 0 for player inventory.
			State ID	VarInt	See State ID
			Count	VarInt	Number of elements in the following array.
			Slot Data	Array of Slot	
			Carried Item	Slot	Item held by player.



The inventory slots

See inventory windows for further information about how slots are indexed.

Window Property

This packet is used to inform the client that part of a GUI window should be updated.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x15	Play	Client	Window ID	Unsigned Byte	
			Property	Short	The property to be updated, see below.
			Value	Short	The new value for the property, see below.

The meaning of the Property field depends on the type of the window. The following table shows the known combinations of window type and property, and how the value is to be interpreted.

Window type	Property	Value
Furnace	0: Fire icon (fuel left)	counting from fuel burn time down to 0 (in-game ticks)
	1: Maximum fuel burn time	fuel burn time or 0 (in-game ticks)
	2: Progress arrow	counting from 0 to maximum progress (in-game ticks)
	3: Maximum progress	always 200 on the notchian server
Enchantment Table	0: Level requirement for top enchantment slot	The enchantment's xp level requirement
	1: Level requirement for middle enchantment slot	
	2: Level requirement for bottom enchantment slot	
	3: The enchantment seed	Used for drawing the enchantment names (in SGA) clientside. The same seed *is* used to calculate enchantments, but some of the data isn't sent to the client to prevent easily guessing the entire list (the seed value here is the regular seed bitwise and `0xFFFFFFF0`).
	4: Enchantment ID shown on mouse hover over top enchantment slot	The enchantment id (set to -1 to hide it), see below for values
	5: Enchantment ID shown on mouse hover over middle enchantment slot	
	6: Enchantment ID shown on mouse hover over bottom enchantment slot	
	7: Enchantment level shown on mouse hover over the top slot	The enchantment level (1 = I, 2 = II, 6 = VI, etc.), or -1 if no enchant
	8: Enchantment level shown on mouse hover over the middle slot	
	9: Enchantment level shown on mouse hover over the bottom slot	
Beacon	0: Power level	0-4, controls what effect buttons are enabled
	1: First potion effect	Potion effect ID (https://minecraft.fandom.com/wiki/Data_values%23Status_effects) for the first effect, or -1 if no effect
	2: Second potion effect	Potion effect ID (https://minecraft.fandom.com/wiki/Data_values%23Status_effects) for the second effect, or -1 if no effect
Anvil	0: Repair cost	The repair's cost in xp levels
Brewing Stand	0: Brew time	0 – 400, with 400 making the arrow empty, and 0 making the arrow full
	1: Fuel time	0 - 20, with 0 making the arrow empty, and 20 making the arrow full
Stonecutter	0: Selected recipe	The index of the selected recipe. -1 means none is selected.
Loom	0: Selected pattern	The index of the selected pattern. 0 means none is selected, 0 is also the internal id of the "base" pattern.
Lectern	0: Page number	The current page number, starting from 0.

For an enchanting table, the following numerical IDs are used:

Numerical ID	Enchantment ID	Enchantment Name
0	minecraft:protection	Protection
1	minecraft:fire_protection	Fire Protection
2	minecraft:feather_falling	Feather Falling
3	minecraft:blast_protection	Blast Protection
4	minecraft:projectile_protection	Projectile Protection
5	minecraft:respiration	Respiration
6	minecraft:aqua_affinity	Aqua Affinity
7	minecraft:thorns	Thorns
8	minecraft:depth_strider	Depth Strider
9	minecraft:frost_walker	Frost Walker
10	minecraft:binding_curse	Curse of Binding
11	minecraft:soul_speed	Soul Speed
12	minecraft:sharpness	Sharpness
13	minecraft:smite	Smite
14	minecraft:bane_of_arthropods	Bane of Arthropods
15	minecraft:knockback	Knockback
16	minecraft:fire_aspect	Fire Aspect
17	minecraft:looting	Looting
18	minecraft:sweeping	Sweeping Edge
19	minecraft:efficiency	Efficiency
20	minecraft:silk_touch	Silk Touch
21	minecraft:unbreaking	Unbreaking
22	minecraft:fortune	Fortune
23	minecraft:power	Power
24	minecraft:punch	Punch
25	minecraft:flame	Flame
26	minecraft:infinity	Infinity
27	minecraft:luck_of_the_sea	Luck of the Sea
28	minecraft:lure	Lure
29	minecraft:loyalty	Loyalty
30	minecraft:impaling	Impaling
31	minecraft:riptide	Riptide
32	minecraft:channeling	Channeling
33	minecraft:multishot	Multishot
34	minecraft:quick_charge	Quick Charge
35	minecraft:piercing	Piercing
36	minecraft:mending	Mending
37	minecraft:vanishing_curse	Curse of Vanishing

Set Slot

Sent by the server when an item in a slot (in a window) is added/removed.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x16	Play	Client	Window ID	Byte	The window which is being updated. 0 for player inventory. Note that all known window types include the player inventory. This packet will only be sent for the currently opened window while the player is performing actions, even if it affects the player inventory. After the window is closed, a number of these packets are sent to update the player's inventory window (0).
			State ID	VarInt	See State ID
			Slot	Short	The slot that should be updated.
			Slot Data	Slot	

To set the cursor (the item currently dragged with the mouse), use -1 as Window ID and as Slot.

This packet can only be used to edit the hotbar of the player's inventory if window ID is set to 0 (slots 36 through 45) if the player is in creative, with their inventory open, and not in their survival inventory tab. Otherwise, when window ID is 0, it can edit any slot in the player's inventory. If the window ID is set to -2, then any slot in the inventory can be used but no add item animation will be played.

Set Cooldown

Applies a cooldown period to all items with the given type. Used by the Notchian server with enderpearls. This packet should be sent when the cooldown starts and also when the cooldown ends (to compensate for lag), although the client will end the cooldown automatically. Can be applied to any item, note that interactions still get sent to the server with the item but the client does not play the animation nor attempt to predict results (i.e block placing).

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x17	Play	Client	Item ID	VarInt	Numeric ID of the item to apply a cooldown to.
			Cooldown Ticks	VarInt	Number of ticks to apply a cooldown for, or 0 to clear the cooldown.

Plugin Message (clientbound)

> *Main article: Plugin channels*

Mods and plugins can use this to send their data. Minecraft itself uses several plugin channels. These internal channels are in the `minecraft` namespace.

More documentation on this: https://dinnerbone.com/blog/2012/01/13/minecraft-plugin-channels-messaging/ (https://dinnerbone.com/blog/2012/01/13/minecraft-plugin-channels-messaging/)

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x18	Play	Client	Channel	Identifier	Name of the plugin channel used to send the data.
			Data	Byte Array	Any data, depending on the channel. `minecraft:` channels are documented here. The length of this array must be inferred from the packet length.

Named Sound Effect

> *See also: #Sound Effect*

Used to play a sound effect on the client. Custom sounds may be added by resource packs.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x19	Play	Client	Sound Name	Identifier	All sound effect names as of 1.16.5 can be seen here (https://pokechu22.github.io/Burger/1.16.5.html#sounds).
			Sound Category	VarInt Enum	The category that this sound will be played from (current categories (https://gist.github.com/konwboj/7c0c380d3923443e9d55)).
			Effect Position X	Int	Effect X multiplied by 8 (fixed-point number with only 3 bits dedicated to the fractional part).
			Effect Position Y	Int	Effect Y multiplied by 8 (fixed-point number with only 3 bits dedicated to the fractional part).
			Effect Position Z	Int	Effect Z multiplied by 8 (fixed-point number with only 3 bits dedicated to the fractional part).
			Volume	Float	1 is 100%, can be more.
			Pitch	Float	Float between 0.5 and 2.0 by Notchian clients.

Disconnect (play)

Sent by the server before it disconnects a client. The client assumes that the server has already closed the connection by the time the packet arrives.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x1A	Play	Client	Reason	Chat	Displayed to the client when the connection terminates.

Entity Status

Entity statuses generally trigger an animation for an entity. The available statuses vary by the entity's type (and are available to subclasses of that type as well).

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x1B	Play	Client	Entity ID	Int	
			Entity Status	Byte Enum	See Entity statuses for a list of which statuses are valid for each type of entity.

Explosion

Sent when an explosion occurs (creepers, TNT, and ghast fireballs).

Each block in Records is set to air. Coordinates for each axis in record is int(X) + record.x

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x1C	Play	Client	X	Float	
			Y	Float	
			Z	Float	
			Strength	Float	A strength greater than or equal to 2.0 spawns a `minecraft:explosion_emitter` particle, while a lesser strength spawns a `minecraft:explosion` particle.
			Record Count	VarInt	Number of elements in the following array.
			Records	Array of (Byte, Byte, Byte)	Each record is 3 signed bytes long; the 3 bytes are the XYZ (respectively) signed offsets of affected blocks.
			Player Motion X	Float	X velocity of the player being pushed by the explosion.
			Player Motion Y	Float	Y velocity of the player being pushed by the explosion.
			Player Motion Z	Float	Z velocity of the player being pushed by the explosion.

Unload Chunk

Tells the client to unload a chunk column.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x1D	Play	Client	Chunk X	Int	Block coordinate divided by 16, rounded down.
			Chunk Z	Int	Block coordinate divided by 16, rounded down.

It is legal to send this packet even if the given chunk is not currently loaded.

Change Game State

Used for a wide variety of game state things, from weather to bed use to gamemode to demo messages.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x1E	Play	Client	Reason	Unsigned Byte	See below.
			Value	Float	Depends on Reason.

Reason codes:

Reason	Effect	Value
0	No respawn block available	Note: Sends message 'block.minecraft.spawn.not_valid'(You have no home bed or charged respawn anchor, or it was obstructed) to the client.
1	End raining	
2	Begin raining	
3	Change gamemode	0: Survival, 1: Creative, 2: Adventure, 3: Spectator.
4	Win game	0: Just respawn player. 1: Roll the credits and respawn player. Note that 1 is only sent by notchian server when player has not yet achieved advancement "The end?", else 0 is sent.
5	Demo event	0: Show welcome to demo screen 101: Tell movement controls 102: Tell jump control 103: Tell inventory control 104: Tell that the demo is over and print a message about how to take a screenshot.
6	Arrow hit player	Note: Sent when any player is struck by an arrow.
7	Rain level change	Note: Seems to change both skycolor and lightning. Rain level starting from 0 to 1.
8	Thunder level change	Note: Seems to change both skycolor and lightning (same as Rain level change, but doesn't start rain). It also requires rain to render by notchian client. Thunder level starting from 0 to 1.
9	Play pufferfish sting sound.	
10	Play elder guardian mob appearance (effect and sound).	
11	Enable respawn screen	0: Enable respawn screen, 1: Immediately respawn (sent when the doImmediateRespawn gamerule changes).

Open Horse Window

This packet is used exclusively for opening the horse GUI. Open Window is used for all other GUIs. The client will not open the inventory if the Entity ID does not point to an horse-like animal.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x1F	Play	Client	Window ID	Unsigned Byte	
			Slot count	VarInt	
			Entity ID	Integer	

Initialize World Border

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x20	Play	Client	X	Double	
			Z	Double	
			Old Diameter	Double	Current length of a single side of the world border, in meters.
			New Diameter	Double	Target length of a single side of the world border, in meters.
			Speed	VarLong	Number of real-time *milli*seconds until New Diameter is reached. It appears that Notchian server does not sync world border speed to game ticks, so it gets out of sync with server lag. If the world border is not moving, this is set to 0.
			Portal Teleport Boundary	VarInt	Resulting coordinates from a portal teleport are limited to ±value. Usually 29999984.
			Warning Blocks	VarInt	In meters.
			Warning Time	VarInt	In seconds as set by `/worldborder warning time`.

The Notchian client determines how solid to display the warning by comparing to whichever is higher, the warning distance or whichever is lower, the distance from the current diameter to the target diameter or the place the border will be after warningTime seconds. In pseudocode:

```
distance = max(min(resizeSpeed * 1000 * warningTime, abs(targetDiameter - currentDiameter)), warningDistance);
if (playerDistance < distance) {
    warning = 1.0 - playerDistance / distance;
} else {
    warning = 0.0;
}
```

Keep Alive (clientbound)

The server will frequently send out a keep-alive, each containing a random ID. The client must respond with the same payload (see serverbound Keep Alive). If the client does not respond to them for over 30 seconds, the server kicks the client. Vice versa, if the server does not send any keep-alives for 20 seconds, the client will disconnect and yields a "Timed out" exception.

The Notchian server uses a system-dependent time in milliseconds to generate the keep alive ID value.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x21	Play	Client	Keep Alive ID	Long	

Chunk Data And Update Light

Main article: Chunk Format
See also: #Unload Chunk

	The following information needs to be added to this page: **How do biomes work now? The biome change happened at the same time as the seed change, but it's not clear how/if biomes could be computed given that it's not the actual seed... (/r/mojira discussion (https://www.reddit.com/r/Mojira/comments/e5at6i/a_discussion_for_the_changes_to_how_biomes_are/) which notes that it seems to be some kind of interpolation, and 3D biomes are only used in the nether)**

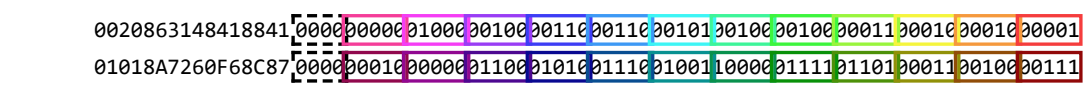
The server only sends skylight information for chunk pillars in the Overworld (https://minecraft.fandom.com/wiki/Overworld), it's up to the client to know in which dimension the player is currently located. You can also infer this information from the primary bitmask and the amount of uncompressed bytes sent. This packet also sends all block entities in the chunk (though sending them is not required; it is still legal to send them with Block Entity Data later).

Packet ID	State	Bound To	Field Name		Field Type		Notes
0x22	Play	Client	Chunk X		Int		Chunk coordinate (block coordinate divided by 16, rounded down)
			Chunk Z		Int		Chunk coordinate (block coordinate divided by 16, rounded down)
			Heightmaps		NBT		Compound containing one long array named `MOTION_BLOCKING`, which is a heightmap for the highest solid block at each position in the chunk (as a compacted long array with 256 entries, with the number of bits per entry varying depending on the world's height, defined by the formula `ceil(log2(height + 1))`). The Notchian server also adds a `WORLD_SURFACE` long array, the purpose of which is unknown, but it's not required for the chunk to be accepted.
			Size		VarInt		Size of Data in bytes
			Data		Byte array		See data structure in Chunk Format
			Number of block entities		VarInt		Number of elements in the following array
			Block Entity	Packed XZ	Array	Byte	The packed section coordinates, calculated from `((blockX & 15) << 4) | (blockZ & 15)`
				Y		Short	The height relative to the world
				Type		VarInt	The type of block entity
				Data		NBT	The block entity's data, without the X, Y, and Z values
			Trust Edges		Boolean		If edges should be trusted for light updates.
			Sky Light Mask		BitSet		BitSet containing bits for each section in the world + 2. Each set bit indicates that the corresponding 16×16×16 chunk section has data in the Sky Light array below. The least significant bit is for blocks 16 blocks to 1 block below the min world height (one section below the world), while the most significant bit covers blocks 1 to 16 blocks above the max world height (one section above the world).
			Block Light Mask		BitSet		BitSet containing bits for each section in the world + 2. Each set bit indicates that the corresponding 16×16×16 chunk section has data in the Block Light array below. The order of bits is the same as in Sky Light Mask.
			Empty Sky Light Mask		BitSet		BitSet containing bits for each section in the world + 2. Each set bit indicates that the corresponding 16×16×16 chunk section has all zeros for its Sky Light data. The order of bits is the same as in Sky Light Mask.
			Empty Block Light Mask		BitSet		BitSet containing bits for each section in the world + 2. Each set bit indicates that the corresponding 16×16×16 chunk section has all zeros for its Block Light data. The order of bits is the same as in Sky Light Mask.
			Sky Light array count		VarInt		Number of entries in the following array; should match the number of bits set in Sky Light Mask
			Sky Light arrays	Length	Array	VarInt	Length of the following array in bytes (always 2048)
				Sky Light array		Array of 2048 bytes	There is 1 array for each bit set to true in the sky light mask, starting with the lowest value. Half a byte per light value. Indexed `((y<<8) | (z<<4) | x) / 2` If there's a remainder, masked 0xF0 else 0x0F.
			Block Light array count		VarInt		Number of entries in the following array; should match the number of bits set in Block Light Mask
			Block Light arrays	Length	Array	VarInt	Length of the following array in bytes (always 2048)
				Block Light array		Array of 2048 bytes	There is 1 array for each bit set to true in the block light mask, starting with the lowest value. Half a byte per light value. Indexed `((y<<8) | (z<<4) | x) / 2` If there's a remainder, masked 0xF0 else 0x0F.

Note that the Notchian client requires an Update View Position packet when it crosses a chunk border, otherwise it'll only display render distance + 2 chunks around the chunk it spawned in.

The compacted array format has been adjusted so that individual entries no longer span across multiple longs, affecting the main data array and heightmaps.

New format, 5 bits per block, containing the following references to blocks in a palette (not shown):
1 2 2 3 4 4 5 6 6 4 8 0 7 4 3 13 15 16 9 14 10 12 0 2



Effect

Sent when a client is to play a sound or particle effect.

By default, the Minecraft client adjusts the volume of sound effects based on distance. The final boolean field is used to disable this, and instead the effect is played from 2 blocks away in the correct direction. Currently this is only used for effect 1023 (wither spawn), effect 1028 (enderdragon death), and effect 1038 (end portal opening); it is ignored on other effects.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x23	Play	Client	Effect ID	Int	The ID of the effect, see below.
			Location	Position	The location of the effect.
			Data	Int	Extra data for certain effects, see below.
			Disable Relative Volume	Boolean	See above.

Effect IDs:

ID	Name	Data
	Sound	
1000	Dispenser dispenses	
1001	Dispenser fails to dispense	
1002	Dispenser shoots	
1003	Ender eye launched	
1004	Firework shot	
1005	Iron door opened	
1006	Wooden door opened	
1007	Wooden trapdoor opened	
1008	Fence gate opened	
1009	Fire extinguished	
1010	Play record	Special case, see below for more info.
1011	Iron door closed	
1012	Wooden door closed	
1013	Wooden trapdoor closed	
1014	Fence gate closed	
1015	Ghast warns	
1016	Ghast shoots	
1017	Enderdragon shoots	
1018	Blaze shoots	
1019	Zombie attacks wood door	
1020	Zombie attacks iron door	
1021	Zombie breaks wood door	
1022	Wither breaks block	
1023	Wither spawned	
1024	Wither shoots	
1025	Bat takes off	
1026	Zombie infects	
1027	Zombie villager converted	
1028	Ender dragon death	
1029	Anvil destroyed	
1030	Anvil used	
1031	Anvil landed	
1032	Portal travel	
1033	Chorus flower grown	
1034	Chorus flower died	
1035	Brewing stand brewed	
1036	Iron trapdoor opened	
1037	Iron trapdoor closed	
1038	End portal created in overworld	
1039	Phantom bites	
1040	Zombie converts to drowned	
1041	Husk converts to zombie by drowning	
1042	Grindstone used	
1043	Book page turned	
	Particle	
1500	Composter composts	

1501	Lava converts block (either water to stone, or removes existing blocks such as torches)	
1502	Redstone torch burns out	
1503	Ender eye placed	
2000	Spawns 10 smoke particles, e.g. from a fire	Direction, see below.
2001	Block break + block break sound	Block state, as an index into the global palette.
2002	Splash potion. Particle effect + glass break sound.	RGB color as an integer (e.g. 8364543 for #7FA1FF).
2003	Eye of Ender entity break animation — particles and sound	
2004	Mob spawn particle effect: smoke + flames	
2005	Bonemeal particles	How many particles to spawn (if set to 0, 15 are spawned).
2006	Dragon breath	
2007	Instant splash potion. Particle effect + glass break sound.	RGB color as an integer (e.g. 8364543 for #7FA1FF).
2008	Ender dragon destroys block	
2009	Wet sponge vaporizes in nether	
3000	End gateway spawn	
3001	Enderdragon growl	
3002	Electric spark	
3003	Copper apply wax	
3004	Copper remove wax	
3005	Copper scrape oxidation	

Smoke directions:

ID	Direction
0	Down
1	Up
2	North
3	South
4	West
5	East

Play record: This is actually a special case within this packet. You can start/stop a record at a specific location. Use a valid Record ID (https://minecraft.fandom.com/wiki/Music_Discs) to start a record (or overwrite a currently playing one), any other value will stop the record. See Data Generators for information on item IDs.

Particle

Displays the named particle

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x24	Play	Client	Particle ID	Int	The particle ID listed in the particle data type.
			Long Distance	Boolean	If true, particle distance increases from 256 to 65536.
			X	Double	X position of the particle.
			Y	Double	Y position of the particle.
			Z	Double	Z position of the particle.
			Offset X	Float	This is added to the X position after being multiplied by `random.nextGaussian()`.
			Offset Y	Float	This is added to the Y position after being multiplied by `random.nextGaussian()`.
			Offset Z	Float	This is added to the Z position after being multiplied by `random.nextGaussian()`.
			Particle Data	Float	The data of each particle.
			Particle Count	Int	The number of particles to create.
			Data	Varies	The variable data listed in the particle data type.

Update Light

Updates light levels for a chunk. See Light (https://minecraft.fandom.com/wiki/Light) for information on how lighting works in Minecraft.

Packet ID	State	Bound To	Field Name		Field Type		Notes
0x25	Play	Client	Chunk X		VarInt		Chunk coordinate (block coordinate divided by 16, rounded down)
			Chunk Z		VarInt		Chunk coordinate (block coordinate divided by 16, rounded down)
			Trust Edges		Boolean		If edges should be trusted for light updates.
			Sky Light Mask		BitSet		BitSet containing bits for each section in the world + 2. Each set bit indicates that the corresponding 16×16×16 chunk section has data in the Sky Light array below. The least significant bit is for blocks 16 blocks to 1 block below the min world height (one section below the world), while the most significant bit covers blocks 1 to 16 blocks above the max world height (one section above the world).
			Block Light Mask		BitSet		BitSet containing bits for each section in the world + 2. Each set bit indicates that the corresponding 16×16×16 chunk section has data in the Block Light array below. The order of bits is the same as in Sky Light Mask.
			Empty Sky Light Mask		BitSet		BitSet containing bits for each section in the world + 2. Each set bit indicates that the corresponding 16×16×16 chunk section has all zeros for its Sky Light data. The order of bits is the same as in Sky Light Mask.
			Empty Block Light Mask		BitSet		BitSet containing bits for each section in the world + 2. Each set bit indicates that the corresponding 16×16×16 chunk section has all zeros for its Block Light data. The order of bits is the same as in Sky Light Mask.
			Sky Light array count		VarInt		Number of entries in the following array; should match the number of bits set in Sky Light Mask
			Sky Light arrays	Length	Array	VarInt	Length of the following array in bytes (always 2048)
				Sky Light array		Array of 2048 bytes	There is 1 array for each bit set to true in the sky light mask, starting with the lowest value. Half a byte per light value.
			Block Light array count		VarInt		Number of entries in the following array; should match the number of bits set in Block Light Mask
			Block Light arrays	Length	Array	VarInt	Length of the following array in bytes (always 2048)
				Block Light array		Array of 2048 bytes	There is 1 array for each bit set to true in the block light mask, starting with the lowest value. Half a byte per light value.

A bit will never be set in both the block light mask and the empty block light mask, though it may be present in neither of them (if the block light does not need to be updated for the corresponding chunk section). The same applies to the sky light mask and the empty sky light mask.

Join Game

See Protocol Encryption for information on logging in.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x26	Play	Client	Entity ID	Int	The player's Entity ID (EID).
			Is hardcore	Boolean	
			Gamemode	Unsigned Byte	0: Survival, 1: Creative, 2: Adventure, 3: Spectator.
			Previous Gamemode	Byte	0: survival, 1: creative, 2: adventure, 3: spectator. The hardcore flag is not included. The previous gamemode. Defaults to -1 if there is no previous gamemode. (More information needed)
			World Count	VarInt	Size of the following array.
			Dimension Names	Array of Identifier	Identifiers for all dimensions on the server.
			Dimension Codec	NBT Tag Compound	The full extent of these is still unknown, but the tag represents a dimension and biome registry. See below for the vanilla default.
			Dimension	NBT Tag Compound	Valid dimensions are defined per dimension registry sent before this. The structure of this tag is a dimension type (see below).
			Dimension Name	Identifier	Name of the dimension being spawned into.
			Hashed seed	Long	First 8 bytes of the SHA-256 hash of the world's seed. Used client side for biome noise
			Max Players	VarInt	Was once used by the client to draw the player list, but now is ignored.
			View Distance	VarInt	Render distance (2-32).
			Simulation Distance	VarInt	The distance that the client will process specific things, such as entities.
			Reduced Debug Info	Boolean	If true, a Notchian client shows reduced information on the debug screen (https://minecraft.fandom.com/wiki/Debug_screen). For servers in development, this should almost always be false.
			Enable respawn screen	Boolean	Set to false when the doImmediateRespawn gamerule is true.
			Is Debug	Boolean	True if the world is a debug mode (https://minecraft.fandom.com/wiki/Debug_mode) world; debug mode worlds cannot be modified and have predefined blocks.
			Is Flat	Boolean	True if the world is a superflat (https://minecraft.fandom.com/wiki/Superflat) world; flat worlds have different void fog and a horizon at y=0 instead of y=63.

The **Dimension Codec** NBT Tag Compound (Default value in SNBT (https://gist.github.com/joserobjr/d877abd88767eda006ee2d1d8779b176)) includes two registries: "minecraft:dimension_type" and "minecraft:worldgen/biome".

Name	Type	Notes
minecraft:dimension_type	TAG_Compound	The dimension type registry (see below).
minecraft:worldgen/biome	TAG_Compound	The biome registry (see below).

Dimension type registry:

Name	Type	Notes
type	TAG_String	The name of the registry. Always "minecraft:dimension_type".
value	TAG_List	List of dimension types registry entries (see below).

Dimension type registry entry:

Name	Type	Notes
name	TAG_String	The name of the dimension type (for example, "minecraft:overworld").
id	TAG_Int	The protocol ID of the dimension (matches the index of the element in the registry list).
element	TAG_Compound	The dimension type (see below).

Dimension type:

Name	Type	Meaning	Values
piglin_safe	TAG_Byte	Whether piglins shake and transform to zombified piglins.	1: true, 0: false.
natural	TAG_Byte	When false, compasses spin randomly. When true, nether portals can spawn zombified piglins.	1: true, 0: false.
ambient_light	TAG_Float	How much light the dimension has.	0.0 to 1.0.
fixed_time	Optional TAG_Long	If set, the time of the day is the specified value.	If set, 0 to 24000.
infiniburn	TAG_String	A resource location defining what block tag to use for infiniburn.	"" or minecraft resource "minecraft:...".
respawn_anchor_works	TAG_Byte	Whether players can charge and use respawn anchors.	1: true, 0: false.
has_skylight	TAG_Byte	Whether the dimension has skylight access or not.	1: true, 0: false.
bed_works	TAG_Byte	Whether players can use a bed to sleep.	1: true, 0: false.
effects	TAG_String	?	"minecraft:overworld", "minecraft:the_nether", "minecraft:the_end" or something else.
has_raids	TAG_Byte	Whether players with the Bad Omen effect can cause a raid.	1: true, 0: false.
min_y	TAG_Int	The minimum Y level.	
height	TAG_Int	The maximum height.	
logical_height	TAG_Int	The maximum height to which chorus fruits and nether portals can bring players within this dimension.	0-256.
coordinate_scale	TAG_Double	The multiplier applied to coordinates when traveling to the dimension.	0.00001 - 30000000.0
ultrawarm	TAG_Byte	Whether the dimensions behaves like the nether (water evaporates and sponges dry) or not. Also causes lava to spread thinner.	1: true, 0: false.
has_ceiling	TAG_Byte	Whether the dimension has a bedrock ceiling or not. When true, causes lava to spread faster.	1: true, 0: false.

Biome registry:

Name	Type	Notes
type	TAG_String	The name of the registry. Always "minecraft:worldgen/biome".
value	TAG_List	List of biome registry entries (see below).

Biome registry entry:

Name	Type	Notes
name	TAG_String	The name of the biome (for example, "minecraft:ocean").
id	TAG_Int	The protocol ID of the biome (matches the index of the element in the registry list).
element	TAG_Compound	The biome properties (see below).

Biome properties:

Name		Type		Meaning		Values
precipitation		TAG_String		The type of precipitation in the biome.		"rain", "snow", or "none".
depth		TAG_Float		The depth factor of the biome.		The default values vary between 1.5 and -1.8.
temperature		TAG_Float		The temperature factor of the biome.		The default values vary between 2.0 and -0.5.
scale		TAG_Float		?		The default values vary between 1.225 and 0.0.
downfall		TAG_Float		?		The default values vary between 1.0 and 0.0.
category		TAG_String		The category of the biome.		Known values are "ocean", "plains", "desert", "forest", "extreme_hills", "taiga", "swamp", "river", "nether", "the_end", "icy", "mushroom", "beach", "jungle", "mesa", "savanna", and "none".
temperature_modifier		Optional TAG_String		?		The only known value is "frozen".
effects	sky_color	TAG_Compound	TAG_Int	The color of the sky.		Example: 8364543, which is #7FA1FF in RGB.
	water_fog_color		TAG_Int	Possibly the tint color when swimming.		Example: 8364543, which is #7FA1FF in RGB.
	fog_color		TAG_Int	Possibly the color of the fog effect when looking past the view distance.		Example: 8364543, which is #7FA1FF in RGB.
	water_color		TAG_Int	The tint color of the water blocks.		Example: 8364543, which is #7FA1FF in RGB.
	foliage_color		Optional TAG_Int	The tint color of the grass.		Example: 8364543, which is #7FA1FF in RGB.
	grass_color		Optional TAG_Int	?		Example: 8364543, which is #7FA1FF in RGB.
	grass_color_modifier		Optional TAG_String	Unknown, likely affects foliage color.		If set, known values are "swamp" and "dark_forest".
	music		Optional TAG_Compound	Music properties for the biome.		If present, contains the fields: replace_current_music (TAG_Byte), sound (TAG_String), max_delay (TAG_Int), min_delay (TAG_Int).
	ambient_sound		Optional TAG_String	Ambient soundtrack.		If present, the ID of a soundtrack. Example: "minecraft:ambient.basalt_deltas.loop".
	additions_sound		Optional TAG_Compound	Additional ambient sound that plays randomly.		If present, contains the fields: sound (TAG_String), tick_chance (TAG_Double).
	mood_sound		Optional TAG_Compound	Additional ambient sound that plays at an interval.		If present, contains the fields: sound (TAG_String), tick_delay (TAG_Int), offset (TAG_Double), block_search_extent (TAG_Int).
particle	probability	Optional TAG_Compound	TAG_FLOAT	Particles that appear randomly in the biome.	Possibly the probability of spawning the particle.	?
	options		TAG_COMPOUND		The properties of the particle to spawn.	Contains the field "type" (TAG_String), which identifies the particle type.

Map Data

Updates a rectangular area on a map (https://minecraft.fandom.com/wiki/Map) item.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x27	Play	Client	Map ID	VarInt	Map ID of the map being modified
			Scale	Byte	From 0 for a fully zoomed-in map (1 block per pixel) to 4 for a fully zoomed-out map (16 blocks per pixel)
			Locked	Boolean	True if the map has been locked in a cartography table
			Tracking Position	Boolean	Specifies whether player and item frame icons are shown
			Icon Count	VarInt	Number of elements in the following array. Only present if "Tracking Position" is true.
			Icon: Type	Array: VarInt enum	See below
			Icon: X	Array: Byte	Map coordinates: -128 for furthest left, +127 for furthest right
			Icon: Z	Array: Byte	Map coordinates: -128 for highest, +127 for lowest
			Icon: Direction	Array: Byte	0-15
			Icon: Has Display Name	Array: Boolean	
			Icon: Display Name	Array: Optional Chat	Only present if previous Boolean is true
			Columns	Unsigned Byte	Number of columns updated
			Rows	Optional Unsigned Byte	Only if Columns is more than 0; number of rows updated
			X	Optional Byte	Only if Columns is more than 0; x offset of the westernmost column
			Z	Optional Byte	Only if Columns is more than 0; z offset of the northernmost row
			Length	Optional VarInt	Only if Columns is more than 0; length of the following array
			Data	Optional Array of Unsigned Byte	Only if Columns is more than 0; see Map item format (https://minecraft.fandom.com/wiki/Map_item_format)

For icons, a direction of 0 is a vertical icon and increments by 22.5° (360/16).

Types are based off of rows and columns in `map_icons.png`:

Icon type	Result
0	White arrow (players)
1	Green arrow (item frames)
2	Red arrow
3	Blue arrow
4	White cross
5	Red pointer
6	White circle (off-map players)
7	Small white circle (far-off-map players)
8	Mansion
9	Temple
10	White Banner
11	Orange Banner
12	Magenta Banner
13	Light Blue Banner
14	Yellow Banner
15	Lime Banner
16	Pink Banner
17	Gray Banner
18	Light Gray Banner
19	Cyan Banner
20	Purple Banner
21	Blue Banner
22	Brown Banner
23	Green Banner
24	Red Banner
25	Black Banner
26	Treasure marker

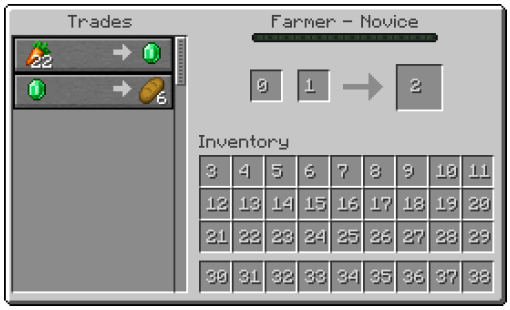
Trade List

The list of trades a villager NPC is offering.

Packet ID	State	Bound To	Field Name		Field Type		Notes
0x28	Play	Client	Window ID		VarInt		The ID of the window that is open; this is an int rather than a byte.
			Size		Byte		The number of trades in the following array.
			Trades	Input item 1	Array	Slot	The first item the player has to supply for this villager trade. The count of the item stack is the default "price" of this trade.
				Output item		Slot	The item the player will receive from this villager trade.
				Has second item		Boolean	Whether there is a second item.
				Input item 2		Optional Slot	The second item the player has to supply for this villager trade; only present if has a second item is true.
				Trade disabled		Boolean	True if the trade is disabled; false if the trade is enabled.
				Number of trade uses		Integer	Number of times the trade has been used so far. If equal to the maximum number of trades, the client will display a red X.
				Maximum number of trade uses		Integer	Number of times this trade can be used before it's exhausted.
				XP		Integer	Amount of XP the villager will earn each time the trade is used.
				Special Price		Integer	Can be zero or negative. The number is added to the price when an item is discounted due to player reputation or other effects.
				Price Multiplier		Float	Can be low (0.05) or high (0.2). Determines how much demand, player reputation, and temporary effects will adjust the price.
				Demand		Integer	If positive, causes the price to increase. Negative values seem to be treated the same as zero.
			Villager level		VarInt		Appears on the trade GUI; meaning comes from the translation key `merchant.level.` + level. 1: Novice, 2: Apprentice, 3: Journeyman, 4: Expert, 5: Master.
			Experience		VarInt		Total experience for this villager (always 0 for the wandering trader).
			Is regular villager		Boolean		True if this is a regular villager; false for the wandering trader. When false, hides the villager level and some other GUI elements.
			Can restock		Boolean		True for regular villagers and false for the wandering trader. If true, the "Villagers restock up to two times per day." message is displayed when hovering over disabled trades.

Modifiers can increase or decrease the number of items for the first input slot. The second input slot and the output slot never change the nubmer of items. The number of items may never be less than 1, and never more than the stack size. If special price and demand are both zero, only the default price is displayed. If either is non-zero, then the adjusted price is displayed next to the crossed-out default price. The adjusted prices is calculated as follows:

Adjusted price = default price + floor(default price x multiplier x demand) + special price



The merchant UI, for reference

Entity Position

This packet is sent by the server when an entity moves less then 8 blocks; if an entity moves more than 8 blocks Entity Teleport should be sent instead.

This packet allows at most 8 blocks movement in any direction, because short range is from -32768 to 32767. And `32768 / (128 * 32)` = 8.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x29	Play	Client	Entity ID	VarInt	
			Delta X	Short	Change in X position as `(currentX * 32 - prevX * 32) * 128`.
			Delta Y	Short	Change in Y position as `(currentY * 32 - prevY * 32) * 128`.
			Delta Z	Short	Change in Z position as `(currentZ * 32 - prevZ * 32) * 128`.
			On Ground	Boolean	

Entity Position and Rotation

This packet is sent by the server when an entity rotates and moves. Since a short range is limited from -32768 to 32767, and movement is offset of fixed-point numbers, this packet allows at most 8 blocks movement in any direction. (`-32768 / (32 * 128) == -8`)

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x2A	Play	Client	Entity ID	VarInt	
			Delta X	Short	Change in X position as `(currentX * 32 - prevX * 32) * 128`.
			Delta Y	Short	Change in Y position as `(currentY * 32 - prevY * 32) * 128`.
			Delta Z	Short	Change in Z position as `(currentZ * 32 - prevZ * 32) * 128`.
			Yaw	Angle	New angle, not a delta.
			Pitch	Angle	New angle, not a delta.
			On Ground	Boolean	

Entity Rotation

This packet is sent by the server when an entity rotates.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x2B	Play	Client	Entity ID	VarInt	
			Yaw	Angle	New angle, not a delta.
			Pitch	Angle	New angle, not a delta.
			On Ground	Boolean	

Vehicle Move (clientbound)

Note that all fields use absolute positioning and do not allow for relative positioning.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x2C	Play	Client	X	Double	Absolute position (X coordinate).
			Y	Double	Absolute position (Y coordinate).
			Z	Double	Absolute position (Z coordinate).
			Yaw	Float	Absolute rotation on the vertical axis, in degrees.
			Pitch	Float	Absolute rotation on the horizontal axis, in degrees.

Open Book

Sent when a player right clicks with a signed book. This tells the client to open the book GUI.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x2D	Play	Client	Hand	VarInt enum	0: Main hand, 1: Off hand .

Open Window

This is sent to the client when it should open an inventory, such as a chest, workbench, or furnace. This message is not sent anywhere for clients opening their own inventory. Resending this packet with already existing window id, will update the window title and window type without closing the window. For horses, use Open Horse Window.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x2E	Play	Client	Window ID	VarInt	A unique id number for the window to be displayed. Notchian server implementation is a counter, starting at 1.
			Window Type	VarInt	The window type to use for display. Contained in the `minecraft:menu` registry; see Inventory for the different values.
			Window Title	Chat	The title of the window.

Open Sign Editor

Sent when the client has placed a sign and is allowed to send Update Sign. There must already be a sign at the given location (which the client does not do automatically) - send a Block Change first.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x2F	Play	Client	Location	Position	

Ping

Packet is not used by the Notchian server. When sent to the client, client responds with a Pong packet with the same id.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x30	Play	Client	ID	Int	

Craft Recipe Response

Response to the serverbound packet (Craft Recipe Request), with the same recipe ID. Appears to be used to notify the UI.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x31	Play	Client	Window ID	Byte	
			Recipe	Identifier	A recipe ID.

Player Abilities (clientbound)

The latter 2 floats are used to indicate the field of view and flying speed respectively, while the first byte is used to determine the value of 4 booleans.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x32	Play	Client	Flags	Byte	Bit field, see below.
			Flying Speed	Float	0.05 by default.
			Field of View Modifier	Float	Modifies the field of view, like a speed potion. A Notchian server will use the same value as the movement speed sent in the Entity Properties packet, which defaults to 0.1 for players.

About the flags:

Field	Bit
Invulnerable	0x01
Flying	0x02
Allow Flying	0x04
Creative Mode (Instant Break)	0x08

End Combat Event

Unused by the Notchain client. This data was once used for twitch.tv metadata circa 1.8.f

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x33	Play	Client	Duration	VarInt	Length of the combat in ticks.
			Entity ID	Int	ID of the primary opponent of the ended combat, or -1 if there is no obvious primary opponent.

Enter Combat Event

Unused by the Notchain client. This data was once used for twitch.tv metadata circa 1.8.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x34	Play	Client	*no fields*		

Death Combat Event

Used to send a respawn screen.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x35	Play	Client	Player ID	VarInt	Entity ID of the player that died (should match the client's entity ID).
			Entity ID	Int	The killer entity's ID, or -1 if there is no obvious killer.
			Message	Chat	The death message.

Player Info

Sent by the server to update the user list (<tab> in the client).

Packet ID	State	Bound To	Field Name			Field Type			Notes
0x36	Play	Client	Action			VarInt			Determines the rest of the Player format after the UUID.
			Number Of Players			VarInt			Number of elements in the following array.
			Player	UUID		Array	UUID		
				Action	**Field Name**				
				0: add player	Name		String (16)		
					Number Of Properties		VarInt		Number of elements in the following array.
					Property	Name	Array	String (32767)	
						Value		String (32767)	
						Is Signed		Boolean	
						Signature		Optional String (32767)	Only if Is Signed is true.
					Gamemode			VarInt	
					Ping			VarInt	Measured in milliseconds.
					Has Display Name			Boolean	
					Display Name			Optional Chat	Only if Has Display Name is true.
				1: update gamemode	Gamemode			VarInt	
				2: update latency	Ping			VarInt	Measured in milliseconds.
				3: update display name	Has Display Name			Boolean	
					Display Name			Optional Chat	Only send if Has Display Name is true.
				4: remove player	*no fields*		*no fields*		

The Property field looks as in the response of Mojang API#UUID -> Profile + Skin/Cape, except of course using the protocol format instead of JSON. That is, each player will usually have one property with Name “textures” and Value being a base64-encoded JSON string as documented at Mojang API#UUID -> Profile + Skin/Cape. An empty properties array is also acceptable, and will cause clients to display the player with one of the two default skins depending on UUID.

Ping values correspond with icons in the following way:

- A ping that negative (i.e. not known to the server yet) will result in the no connection icon.
- A ping under 150 milliseconds will result in 5 bars
- A ping under 300 milliseconds will result in 4 bars
- A ping under 600 milliseconds will result in 3 bars
- A ping under 1000 milliseconds (1 second) will result in 2 bars
- A ping greater than or equal to 1 second will result in 1 bar.

Face Player

Used to rotate the client player to face the given location or entity (for `/teleport [<targets>] <x> <y> <z> facing`).

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x37	Play	Client	Feet/eyes	VarInt enum	Values are feet=0, eyes=1. If set to eyes, aims using the head position; otherwise aims using the feet position.
			Target x	Double	x coordinate of the point to face towards.
			Target y	Double	y coordinate of the point to face towards.
			Target z	Double	z coordinate of the point to face towards.
			Is entity	Boolean	If true, additional information about an entity is provided.
			Entity ID	Optional VarInt	Only if is entity is true — the entity to face towards.
			Entity feet/eyes	Optional VarInt enum	Whether to look at the entity's eyes or feet. Same values and meanings as before, just for the entity's head/feet.

If the entity given by entity ID cannot be found, this packet should be treated as if is entity was false.

Player Position And Look (clientbound)

Updates the player's position on the server. This packet will also close the “Downloading Terrain” screen when joining/respawning.

If the distance between the last known position of the player on the server and the new position set by this packet is greater than 100 meters, the client will be kicked for “You moved too quickly :((Hacking?)”.

Also if the fixed-point number of X or Z is set greater than `3.2E7D` the client will be kicked for “Illegal position”.

Yaw is measured in degrees, and does not follow classical trigonometry rules. The unit circle of yaw on the XZ-plane starts at (0, 1) and turns counterclockwise, with 90 at (-1, 0), 180 at (0, -1) and 270 at (1, 0). Additionally, yaw is not clamped to between 0 and 360 degrees; any number is valid, including negative numbers and numbers greater than 360.

Pitch is measured in degrees, where 0 is looking straight ahead, -90 is looking straight up, and 90 is looking straight down.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x38	Play	Client	X	Double	Absolute or relative position, depending on Flags.
			Y	Double	Absolute or relative position, depending on Flags.
			Z	Double	Absolute or relative position, depending on Flags.
			Yaw	Float	Absolute or relative rotation on the X axis, in degrees.
			Pitch	Float	Absolute or relative rotation on the Y axis, in degrees.
			Flags	Byte	Bit field, see below.
			Teleport ID	VarInt	Client should confirm this packet with Teleport Confirm containing the same Teleport ID.
			Dismount Vehicle	Boolean	True if the player should dismount their vehicle.

About the Flags field:

```
<Dinnerbone> It's a bitfield, X/Y/Z/Y_ROT/X_ROT. If X is set, the x value is relative and not absolute.
```

Field	Bit
X	0x01
Y	0x02
Z	0x04
Y_ROT	0x08
X_ROT	0x10

Unlock Recipes

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x39	Play	Client	Action	VarInt	0: init, 1: add, 2: remove.
			Crafting Recipe Book Open	Boolean	If true, then the crafting recipe book will be open when the player opens its inventory.
			Crafting Recipe Book Filter Active	Boolean	If true, then the filtering option is active when the players opens its inventory.
			Smelting Recipe Book Open	Boolean	If true, then the smelting recipe book will be open when the player opens its inventory.
			Smelting Recipe Book Filter Active	Boolean	If true, then the filtering option is active when the players opens its inventory.
			Blast Furnace Recipe Book Open	Boolean	If true, then the blast furnace recipe book will be open when the player opens its inventory.
			Blast Furnace Recipe Book Filter Active	Boolean	If true, then the filtering option is active when the players opens its inventory.
			Smoker Recipe Book Open	Boolean	If true, then the smoker recipe book will be open when the player opens its inventory.
			Smoker Recipe Book Filter Active	Boolean	If true, then the filtering option is active when the players opens its inventory.
			Array size 1	VarInt	Number of elements in the following array.
			Recipe IDs	Array of Identifier	
			Array size 2	Optional VarInt	Number of elements in the following array, only present if mode is 0 (init).
			Recipe IDs	Optional Array of Identifier	Only present if mode is 0 (init)

Action:

- 0 (init) = All the recipes in list 1 will be tagged as displayed, and all the recipes in list 2 will be added to the recipe book. Recipes that aren't tagged will be shown in the notification.
- 1 (add) = All the recipes in the list are added to the recipe book and their icons will be shown in the notification.
- 2 (remove) = Remove all the recipes in the list. This allows them to be re-displayed when they are re-added.

Destroy Entities

Sent by the server when an entity is to be destroyed on the client.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x3A	Play	Client	Count	VarInt	Number of elements in the following array.
			Entity IDs	Array of VarInt	The list of entities to destroy.

Remove Entity Effect

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x3B	Play	Client	Entity ID	VarInt	
			Effect ID	VarInt	See this table (https://minecraft.fandom.com/wiki/Status_effect%23Effect_IDs).

Resource Pack Send

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x3C	Play	Client	URL	String (32767)	The URL to the resource pack.
			Hash	String (40)	A 40 character hexadecimal and lowercase SHA-1 hash of the resource pack file. (must be lower case in order to work) If it's not a 40 character hexadecimal string, the client will not use it for hash verification and likely waste bandwidth — but it will still treat it as a unique id
			Forced	Boolean	The notchian client will be forced to use the resource pack from the server. If they decline they will be kicked from the server.
			Has Prompt Message	Boolean	`true` If the next field will be sent `false` otherwise. When `false`, this is the end of the packet
			Prompt Message	Optional Chat	This is shown in the prompt making the client accept or decline the resource pack.

Respawn

To change the player's dimension (overworld/nether/end), send them a respawn packet with the appropriate dimension, followed by prechunks/chunks for the new dimension, and finally a position and look packet. You do not need to unload chunks, the client will do it automatically.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x3D	Play	Client	Dimension	NBT Tag Compound	Valid dimensions are defined per dimension registry sent in Join Game
			Dimension Name	Identifier	Name of the dimension being spawned into.
			Hashed seed	Long	First 8 bytes of the SHA-256 hash of the world's seed. Used client side for biome noise
			Gamemode	Unsigned Byte	0: survival, 1: creative, 2: adventure, 3: spectator. The hardcore flag is not included
			Previous Gamemode	Unsigned Byte	-1: null 0: survival, 1: creative, 2: adventure, 3: spectator. The hardcore flag is not included. The previous gamemode. (More information needed)
			Is Debug	Boolean	True if the world is a debug mode (https://minecraft.fandom.com/wiki/Debug_mode) world; debug mode worlds cannot be modified and have predefined blocks.
			Is Flat	Boolean	True if the world is a superflat (https://minecraft.fandom.com/wiki/Superflat) world; flat worlds have different void fog and a horizon at y=0 instead of y=63.
			Copy metadata	Boolean	If false, metadata is reset on the respawned player entity. Set to true for dimension changes (including the dimension change triggered by sending client status perform respawn to exit the end poem/credits), and false for normal respawns.

⚠ Avoid changing player's dimension to same dimension they were already in unless they are dead. If you change the dimension to one they are already in, weird bugs can occur, such as the player being unable to attack other players in new world (until they die and respawn).

If you must respawn a player in the same dimension without killing them, send two respawn packets, one to a different world and then another to the world you want. You do not need to complete the first respawn; it only matters that you send two packets.

Entity Head Look

Changes the direction an entity's head is facing.

While sending the Entity Look packet changes the vertical rotation of the head, sending this packet appears to be necessary to rotate the head horizontally.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x3E	Play	Client	Entity ID	VarInt	
			Head Yaw	Angle	New angle, not a delta.

Multi Block Change

Fired whenever 2 or more blocks are changed within the same chunk on the same tick.

⚠ Changing blocks in chunks not loaded by the client is unsafe (see note on Block Change).

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x3F	Play	Client	Chunk section position	Long	Chunk section coordinate (encoded chunk x and z with each 22 bits, and section y with 20 bits, from left to right).
				Boolean	Always inverse the preceding Update Light packet's "Trust Edges" bool
			Blocks array size	VarInt	Number of elements in the following array.
			Blocks	Array of VarLong	Each entry is composed of the block state id, shifted left by 12, and the relative block position in the chunk section (4 bits for x, z, and y, from left to right).

Chunk section position is encoded:

```
((sectionX & 0x3FFFFF) << 42) | (sectionY & 0xFFFFF) | ((sectionZ & 0x3FFFFF) << 20);
```

and decoded:

```
sectionX = long >> 42;
sectionY = long << 44 >> 44;
sectionZ = long << 22 >> 42;
```

Blocks are encoded:

```
blockStateId << 12 | (blockLocalX << 8 | blockLocalZ << 4 | blockLocalY)
//Uses the local position of the given block position relative to its respective chunk section
```

Select Advancement Tab

Sent by the server to indicate that the client should switch advancement tab. Sent either when the client switches tab in the GUI or when an advancement in another tab is made.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x40	Play	Client	Has id	Boolean	Indicates if the next field is present.
			Optional Identifier	String (32767)	See below.

The Identifier can be one of the following:

Optional Identifier
minecraft:story/root
minecraft:nether/root
minecraft:end/root
minecraft:adventure/root
minecraft:husbandry/root

If no or an invalid identifier is sent, the client will switch to the first tab in the GUI.

Action Bar

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x41	Play	Client	Action bar text	Chat	Displays a message above the hotbar (the same as position 2 in Chat Message (clientbound).

World Border Center

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x42	Play	Client	X	Double	
			Z	Double	

World Border Lerp Size

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x43	Play	Client	Old Diameter	Double	Current length of a single side of the world border, in meters.
			New Diameter	Double	Target length of a single side of the world border, in meters.
			Speed	VarLong	Number of real-time *milli*seconds until New Diameter is reached. It appears that Notchian server does not sync world border speed to game ticks, so it gets out of sync with server lag. If the world border is not moving, this is set to 0.

World Border Size

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x44	Play	Client	Diameter	Double	Length of a single side of the world border, in meters.

World Border Warning Delay

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x45	Play	Client	Warning Time	VarInt	In seconds as set by `/worldborder warning time`.

World Border Warning Reach

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x46	Play	Client	Warning Blocks	VarInt	In meters.

Camera

Sets the entity that the player renders from. This is normally used when the player left-clicks an entity while in spectator mode.

The player's camera will move with the entity and look where it is looking. The entity is often another player, but can be any type of entity. The player is unable to move this entity (move packets will act as if they are coming from the other entity).

If the given entity is not loaded by the player, this packet is ignored. To return control to the player, send this packet with their entity ID.

The Notchian server resets this (sends it back to the default entity) whenever the spectated entity is killed or the player sneaks, but only if they were spectating an entity. It also sends this packet whenever the player switches out of spectator mode (even if they weren't spectating an entity).

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x47	Play	Client	Camera ID	VarInt	ID of the entity to set the client's camera to.

The notchian client also loads certain shaders for given entities:

- Creeper → `shaders/post/creeper.json`
- Spider (and cave spider) → `shaders/post/spider.json`
- Enderman → `shaders/post/invert.json`
- Anything else → the current shader is unloaded

Held Item Change (clientbound)

Sent to change the player's slot selection.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x48	Play	Client	Slot	Byte	The slot which the player has selected (0–8).

Update View Position

The following information needs to be added to this page:

Why is this even needed? Is there a better name for it? My guess is that it's something to do with logical behavior with latency, but it still seems weird.

Updates the client's location. This is used to determine what chunks should remain loaded and if a chunk load should be ignored; chunks outside of the view distance may be unloaded.

Sent whenever the player moves across a chunk border horizontally, and also (according to testing) for any integer change in the vertical axis, even if it doesn't go across a chunk section border.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x49	Play	Client	Chunk X	VarInt	Chunk X coordinate of the player's position.
			Chunk Z	VarInt	Chunk Z coordinate of the player's position.

Update View Distance

Sent by the integrated singleplayer server when changing render distance. This packet is sent by the server when the client reappears in the overworld after leaving the end.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x4A	Play	Client	View Distance	VarInt	Render distance (2-32).

Spawn Position

Sent by the server after login to specify the coordinates of the spawn point (the point at which players spawn at, and which the compass points to). It can be sent at any time to update the point compasses point at.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x4B	Play	Client	Location	Position	Spawn location.
			Angle	Float	The angle at which to respawn at.

Display Scoreboard

This is sent to the client when it should display a scoreboard.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x4C	Play	Client	Position	Byte	The position of the scoreboard. 0: list, 1: sidebar, 2: below name, 3 - 18: team specific sidebar, indexed as 3 + team color.
			Score Name	String (16)	The unique name for the scoreboard to be displayed.

Entity Metadata

Updates one or more metadata properties for an existing entity. Any properties not included in the Metadata field are left unchanged.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x4D	Play	Client	Entity ID	VarInt	
			Metadata	Entity Metadata	

Attach Entity

This packet is sent when an entity has been leashed (https://minecraft.fandom.com/wiki/Lead) to another entity.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x4E	Play	Client	Attached Entity ID	Int	Attached entity's EID.
			Holding Entity ID	Int	ID of the entity holding the lead. Set to -1 to detach.

Entity Velocity

Velocity is believed to be in units of 1/8000 of a block per server tick (50ms); for example, -1343 would move (-1343 / 8000) = −0.167875 blocks per tick (or −3,3575 blocks per second).

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x4F	Play	Client	Entity ID	VarInt	
			Velocity X	Short	Velocity on the X axis.
			Velocity Y	Short	Velocity on the Y axis.
			Velocity Z	Short	Velocity on the Z axis.

Entity Equipment

Packet ID	State	Bound To	Field Name		Field Type		Notes
0x50	Play	Client	Entity ID		VarInt		Entity's EID.
			Equipment	Slot	Array	Byte Enum	Equipment slot. 0: main hand, 1: off hand, 2–5: armor slot (2: boots, 3: leggings, 4: chestplate, 5: helmet). Also has the top bit set if another entry follows, and otherwise unset if this is the last item in the array.
				Item		Slot	

Set Experience

Sent by the server when the client should change experience levels.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x51	Play	Client	Experience bar	Float	Between 0 and 1.
			Level	VarInt	
			Total Experience	VarInt	See Experience#Leveling up (https://minecraft.fandom.com/wiki/Experience%23Leveling_up) on the Minecraft Wiki for Total Experience to Level conversion.

Update Health

Sent by the server to update/set the health of the player it is sent to.

Food saturation (https://minecraft.fandom.com/wiki/Food%23Hunger_vs._Saturation) acts as a food "overcharge". Food values will not decrease while the saturation is over zero. Players logging in automatically get a saturation of 5.0. Eating food increases the saturation as well as the food bar.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x52	Play	Client	Health	Float	0 or less = dead, 20 = full HP.
			Food	VarInt	0–20.
			Food Saturation	Float	Seems to vary from 0.0 to 5.0 in integer increments.

Scoreboard Objective

This is sent to the client when it should create a new scoreboard (https://minecraft.fandom.com/wiki/Scoreboard) objective or remove one.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x53	Play	Client	Objective Name	String (16)	A unique name for the objective.
			Mode	Byte	0 to create the scoreboard. 1 to remove the scoreboard. 2 to update the display text.
			Objective Value	Optional Chat	Only if mode is 0 or 2. The text to be displayed for the score.
			Type	Optional VarInt enum	Only if mode is 0 or 2. 0 = "integer", 1 = "hearts".

Set Passengers

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x54	Play	Client	Entity ID	VarInt	Vehicle's EID.
			Passenger Count	VarInt	Number of elements in the following array.
			Passengers	Array of VarInt	EIDs of entity's passengers.

Teams

Creates and updates teams.

Packet ID	State	Bound To	Field Name		Field Type	Notes
0x55	Play	Client	Team Name		String (16)	A unique name for the team. (Shared with scoreboard).
			Mode		Byte	Determines the layout of the remaining packet.
			0: create team	Team Display Name	Chat	
				Friendly Flags	Byte	Bit mask. 0x01: Allow friendly fire, 0x02: can see invisible players on same team.
				Name Tag Visibility	String Enum (32)	`always`, `hideForOtherTeams`, `hideForOwnTeam`, `never`.
				Collision Rule	String Enum (32)	`always`, `pushOtherTeams`, `pushOwnTeam`, `never`.
				Team Color	VarInt enum	Used to color the name of players on the team; see below.
				Team Prefix	Chat	Displayed before the names of players that are part of this team.
				Team Suffix	Chat	Displayed after the names of players that are part of this team.
				Entity Count	VarInt	Number of elements in the following array.
				Entities	Array of String (40)	Identifiers for the entities in this team. For players, this is their username; for other entities, it is their UUID.
			1: remove team	*no fields*	*no fields*	
			2: update team info	Team Display Name	Chat	
				Friendly Flags	Byte	Bit mask. 0x01: Allow friendly fire, 0x02: can see invisible entities on same team.
				Name Tag Visibility	String Enum (32)	`always`, `hideForOtherTeams`, `hideForOwnTeam`, `never`
				Collision Rule	String Enum (32)	`always`, `pushOtherTeams`, `pushOwnTeam`, `never`
				Team Color	VarInt enum	Used to color the name of players on the team; see below.
				Team Prefix	Chat	Displayed before the names of players that are part of this team.
				Team Suffix	Chat	Displayed after the names of players that are part of this team.
			3: add entities to team	Entity Count	VarInt	Number of elements in the following array.
				Entities	Array of String (40)	Identifiers for the added entities. For players, this is their username; for other entities, it is their UUID.
			4: remove entities from team	Entity Count	VarInt	Number of elements in the following array.
				Entities	Array of String (40)	Identifiers for the removed entities. For players, this is their username; for other entities, it is their UUID.

Team Color: The color of a team defines how the names of the team members are visualized; any formatting code can be used. The following table lists all the possible values.

ID	Formatting
0-15	Color formatting, same values as Chat colors.
16	Obfuscated
17	Bold
18	Strikethrough
19	Underlined
20	Italic
21	Reset

Update Score

This is sent to the client when it should update a scoreboard item.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x56	Play	Client	Entity Name	String (40)	The entity whose score this is. For players, this is their username; for other entities, it is their UUID.
			Action	Byte	0 to create/update an item. 1 to remove an item.
			Objective Name	String (16)	The name of the objective the score belongs to.
			Value	Optional VarInt	The score to be displayed next to the entry. Only sent when Action does not equal 1.

Update Simulation Distance

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x57	Play	Client	Simulation Distance	VarInt	The distance that the client will process specific things, such as entities.

Set Title SubTitle

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x58	Play	Client	Subtitle Text	Chat	

Time Update

Time is based on ticks, where 20 ticks happen every second. There are 24000 ticks in a day, making Minecraft days exactly 20 minutes long.

The time of day is based on the timestamp modulo 24000. 0 is sunrise, 6000 is noon, 12000 is sunset, and 18000 is midnight.

The default SMP server increments the time by `20` every second.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x59	Play	Client	World Age	Long	In ticks; not changed by server commands.
			Time of day	Long	The world (or region) time, in ticks. If negative the sun will stop moving at the Math.abs of the time.

Set Title Text

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x5A	Play	Client	Title Text	Chat	

Set Title Times

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x5B	Play	Client	Fade In	Int	Ticks to spend fading in.
			Stay	Int	Ticks to keep the title displayed.
			Fade Out	Int	Ticks to spend out, not when to start fading out.

Entity Sound Effect

Plays a sound effect from an entity.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x5C	Play	Client	Sound ID	VarInt	ID of hardcoded sound event (events (https://pokechu22.github.io/Burger/1.16.5.html#sounds) as of 1.16.5).
			Sound Category	VarInt Enum	The category that this sound will be played from (current categories (https://gist.github.com/konwboj/7c0c380d3923443e9d55)).
			Entity ID	VarInt	
			Volume	Float	1.0 is 100%, capped between 0.0 and 1.0 by Notchian clients.
			Pitch	Float	Float between 0.5 and 2.0 by Notchian clients.

Sound Effect

This packet is used to play a number of hardcoded sound events. For custom sounds, use Named Sound Effect.

⚠ Numeric sound effect IDs are liable to change between versions

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x5D	Play	Client	Sound ID	VarInt	ID of hardcoded sound event (events (https://pokechu22.github.io/Burger/1.16.5.html#sounds) as of 1.16.5).
			Sound Category	VarInt Enum	The category that this sound will be played from (current categories (https://gist.github.com/konwboj/7c0c380d3923443e9d55)).
			Effect Position X	Int	Effect X multiplied by 8 (fixed-point number with only 3 bits dedicated to the fractional part).
			Effect Position Y	Int	Effect Y multiplied by 8 (fixed-point number with only 3 bits dedicated to the fractional part).
			Effect Position Z	Int	Effect Z multiplied by 8 (fixed-point number with only 3 bits dedicated to the fractional part).
			Volume	Float	1.0 is 100%, capped between 0.0 and 1.0 by Notchian clients.
			Pitch	Float	Float between 0.5 and 2.0 by Notchian clients.

Stop Sound

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x5E	Play	Client	Flags	Byte	Controls which fields are present.
			Source	Optional VarInt enum	Only if flags is 3 or 1 (bit mask 0x1). See below. If not present, then sounds from all sources are cleared.
			Sound	Optional Identifier	Only if flags is 2 or 3 (bit mask 0x2). A sound effect name, see Named Sound Effect. If not present, then all sounds are cleared.

Categories:

Name	Value
master	0
music	1
record	2
weather	3
block	4
hostile	5
neutral	6
player	7
ambient	8
voice	9

Player List Header And Footer

This packet may be used by custom servers to display additional information above/below the player list. It is never sent by the Notchian server.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x5F	Play	Client	Header	Chat	To remove the header, send a empty text component: `{"text":""}`.
			Footer	Chat	To remove the footer, send a empty text component: `{"text":""}`.

NBT Query Response

Sent in response to Query Block NBT or Query Entity NBT.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x60	Play	Client	Transaction ID	VarInt	Can be compared to the one sent in the original query packet.
			NBT	NBT Tag	The NBT of the block or entity. May be a TAG_END (0) in which case no NBT is present.

Collect Item

Sent by the server when someone picks up an item lying on the ground — its sole purpose appears to be the animation of the item flying towards you. It doesn't destroy the entity in the client memory, and it doesn't add it to your inventory. The server only checks for items to be picked up after each Player Position (and Player Position And Look) packet sent by the client. The collector entity can be any entity; it does not have to be a player. The collected entity also can be any entity, but the Notchian server only uses this for items, experience orbs, and the different varieties of arrows.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x61	Play	Client	Collected Entity ID	VarInt	
			Collector Entity ID	VarInt	
			Pickup Item Count	VarInt	Seems to be 1 for XP orbs, otherwise the number of items in the stack.

Entity Teleport

This packet is sent by the server when an entity moves more than 8 blocks.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x62	Play	Client	Entity ID	VarInt	
			X	Double	
			Y	Double	
			Z	Double	
			Yaw	Angle	(Y Rot)New angle, not a delta.
			Pitch	Angle	(X Rot)New angle, not a delta.
			On Ground	Boolean	

Advancements

Packet ID	State	Bound To	Field Name		Field Type		Notes
0x63	Play	Client	Reset/Clear		Boolean		Whether to reset/clear the current advancements.
			Mapping size		VarInt		Size of the following array.
			Advancement mapping	Key	Array	Identifier	The identifier of the advancement.
				Value		Advancement	See below
			List size		VarInt		Size of the following array.
			Identifiers		Array of Identifier		The identifiers of the advancements that should be removed.
			Progress size		VarInt		Size of the following array.
			Progress mapping	Key	Array	Identifier	The identifier of the advancement.
				Value		Advancement progress	See below.

Advancement structure:

Field Name		Field Type		Notes
Has parent		Boolean		Indicates whether the next field exists.
Parent id		Optional Identifier		The identifier of the parent advancement.
Has display		Boolean		Indicates whether the next field exists.
Display data		Optional advancement display		See below.
Number of criteria		VarInt		Size of the following array.
Criteria	Key	Array	Identifier	The identifier of the criterion.
	Value		**Void**	There is *no* content written here. Perhaps this will be expanded in the future?
Array length		VarInt		Number of arrays in the following array.
Requirements	Array length 2	Array	VarInt	Number of elements in the following array.
	Requirement		Array of String	Array of required criteria.

Advancement display:

Field Name	Field Type	Notes
Title	Chat	
Description	Chat	
Icon	Slot	
Frame type	VarInt enum	0 = `task`, 1 = `challenge`, 2 = `goal`.
Flags	Integer	0x01: has background texture; 0x02: `show_toast`; 0x04: `hidden`.
Background texture	Optional Identifier	Background texture location. Only if flags indicates it.
X coord	Float	
Y coord	Float	

Advancement progress:

Field Name		Field Type		Notes
Size		VarInt		Size of the following array.
Criteria	Criterion identifier	Array	Identifier	The identifier of the criterion.
	Criterion progress		Criterion progress	

Criterion progress:

Field Name	Field Type	Notes
Achieved	Boolean	If true, next field is present.
Date of achieving	Optional Long	As returned by `Date.getTime` (https://docs.oracle.com/javase/6/docs/api/java/util/Date.html#getTime()).

Entity Properties

Sets attributes (https://minecraft.fandom.com/wiki/Attribute) on the given entity.

Packet ID	State	Bound To	Field Name		Field Type		Notes
0x64	Play	Client	Entity ID		VarInt		
			Number Of Properties		VarInt		Number of elements in the following array.
			Property	Key	Array	Identifier	See below.
				Value		Double	See below.
				Number Of Modifiers		VarInt	Number of elements in the following array.
				Modifiers		Array of Modifier Data	See Attribute#Modifiers (https://minecraft.fandom.com/wiki/Attribute%23Modifiers). Modifier Data defined below.

Known Key values (see also Attribute#Modifiers (https://minecraft.fandom.com/wiki/Attribute%23Modifiers)):

Key	Default	Min	Max	Label
generic.max_health	20.0	0.0	1024.0	Max Health.
generic.follow_range	32.0	0.0	2048.0	Follow Range.
generic.knockback_resistance	0.0	0.0	1.0	Knockback Resistance.
generic.movement_speed	0.7	0.0	1024.0	Movement Speed.
generic.attack_damage	2.0	0.0	2048.0	Attack Damage.
generic.attack_speed	4.0	0.0	1024.0	Attack Speed.
generic.flying_speed	0.4	0.0	1024.0	Flying Speed.
generic.armor	0.0	0.0	30.0	Armor.
generic.armor_toughness	0.0	0.0	20.0	Armor Toughness.
generic.attack_knockback	0.0	0.0	5.0	—
generic.luck	0.0	-1024.0	1024.0	Luck.
horse.jump_strength	0.7	0.0	2.0	Jump Strength.
zombie.spawn_reinforcements	0.0	0.0	1.0	Spawn Reinforcements Chance.
generic.reachDistance	5.0	0.0	1024.0	Player Reach Distance (Forge only).
forge.swimSpeed	1.0	0.0	1024.0	Swimming Speed (Forge only).

Unknown attributes will cause a game crash (MC-150405 (https://bugs.mojang.com/browse/MC-150405)) due to the default minimum being larger than the default value.

Modifier Data structure:

Field Name	Field Type	Notes
UUID	UUID	
Amount	Double	May be positive or negative.
Operation	Byte	See below.

The operation controls how the base value of the modifier is changed.

- 0: Add/subtract amount
- 1: Add/subtract amount percent of the current value
- 2: Multiply by amount percent

All of the 0's are applied first, and then the 1's, and then the 2's.

Entity Effect

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x65	Play	Client	Entity ID	VarInt	
			Effect ID	VarInt	See this table (https://minecraft.fandom.com/wiki/Status_effect%23Effect_IDs).
			Amplifier	Byte	Notchian client displays effect level as Amplifier + 1.
			Duration	VarInt	Duration in ticks.
			Flags	Byte	Bit field, see below.

Within flags:

- 0x01: Is ambient - was the effect spawned from a beacon? All beacon-generated effects are ambient. Ambient effects use a different icon in the HUD (blue border rather than gray). If all effects on an entity are ambient, the "Is potion effect ambient" living metadata field should be set to true. Usually should not be enabled.
- 0x02: Show particles - should all particles from this effect be hidden? Effects with particles hidden are not included in the calculation of the effect color, and are not rendered on the HUD (but are still rendered within the inventory). Usually should be enabled.
- 0x04: Show icon - should the icon be displayed on the client? Usually should be enabled.

Declare Recipes

Packet ID	State	Bound To	Field Name		Field Type		Notes
0x66	Play	Client	Num Recipes		VarInt		Number of elements in the following array.
			Recipe	Type	Array	Identifier	The recipe type, see below.
				Recipe ID		Identifier	
				Data		Optional, varies	Additional data for the recipe. For some types, there will be no data.

Recipe types:

<table>
<tr><th>Type</th><th>Description</th><th>Data</th></tr>
<tr><td><code>crafting_shapeless</code></td><td>Shapeless crafting recipe. All items in the ingredient list must be present, but in any order/slot.</td><td>As follows:<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>Group</td><td>String</td><td>Used to group similar recipes together in the recipe book. Tag is present in recipe JSON.</td></tr>
<tr><td>Ingredient count</td><td>VarInt</td><td>Number of elements in the following array.</td></tr>
<tr><td>Ingredients</td><td>Array of Ingredient.</td><td></td></tr>
<tr><td>Result</td><td>Slot</td><td></td></tr>
</table></td></tr>
<tr><td><code>crafting_shaped</code></td><td>Shaped crafting recipe. All items must be present in the same pattern (which may be flipped horizontally or translated).</td><td>As follows:<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>Width</td><td>VarInt</td><td></td></tr>
<tr><td>Height</td><td>VarInt</td><td></td></tr>
<tr><td>Group</td><td>String</td><td>Used to group similar recipes together in the recipe book. Tag is present in recipe JSON.</td></tr>
<tr><td>Ingredients</td><td>Array of Ingredient</td><td>Length is <code>width * height</code>. Indexed by <code>x + (y * width)</code>.</td></tr>
<tr><td>Result</td><td>Slot</td><td></td></tr>
</table></td></tr>
<tr><td><code>crafting_special_armordye</code></td><td>Recipe for dying leather armor</td><td rowspan="15">None</td></tr>
<tr><td><code>crafting_special_bookcloning</code></td><td>Recipe for copying contents of written books</td></tr>
<tr><td><code>crafting_special_mapcloning</code></td><td>Recipe for copying maps</td></tr>
<tr><td><code>crafting_special_mapextending</code></td><td>Recipe for adding paper to maps</td></tr>
<tr><td><code>crafting_special_firework_rocket</code></td><td>Recipe for making firework rockets</td></tr>
<tr><td><code>crafting_special_firework_star</code></td><td>Recipe for making firework stars</td></tr>
<tr><td><code>crafting_special_firework_star_fade</code></td><td>Recipe for making firework stars fade between multiple colors</td></tr>
<tr><td><code>crafting_special_repairitem</code></td><td>Recipe for repairing items via crafting</td></tr>
<tr><td><code>crafting_special_tippedarrow</code></td><td>Recipe for crafting tipped arrows</td></tr>
<tr><td><code>crafting_special_bannerduplicate</code></td><td>Recipe for copying banner patterns</td></tr>
<tr><td><code>crafting_special_banneraddpattern</code></td><td>Recipe for adding patterns to banners</td></tr>
<tr><td><code>crafting_special_shielddecoration</code></td><td>Recipe for applying a banner's pattern to a shield</td></tr>
<tr><td><code>crafting_special_shulkerboxcoloring</code></td><td>Recipe for recoloring a shulker box</td></tr>
<tr><td><code>crafting_special_suspiciousstew</code></td><td></td></tr>
<tr><td><code>smelting</code></td><td>Smelting recipe</td><td rowspan="4">As follows:</td></tr>
<tr><td><code>blasting</code></td><td>Blast furnace recipe</td></tr>
<tr><td><code>smoking</code></td><td>Smoker recipe</td></tr>
<tr><td><code>campfire_cooking</code></td><td>Campfire recipe</td></tr>
</table>

		Name / **Type** / **Description** Group / String / Used to group similar recipes together in the recipe book. Ingredient / Ingredient / Result / Slot / Experience / Float / Cooking time / VarInt /
`stonecutting`	Stonecutter recipe	As follows: **Name** / **Type** / **Description** Group / String / Used to group similar recipes together in the recipe book. Tag is present in recipe JSON. Ingredient / Ingredient / Result / Slot /
`smithing`	Smithing table recipe	As follows: **Name** / **Type** / **Description** Base / Ingredient / First item. Addition / Ingredient / Second item. Result / Slot /

Ingredient is defined as:

Name	Type	Description
Count	VarInt	Number of elements in the following array.
Items	Array of Slot	Any item in this array may be used for the recipe. The count of each item should be 1.

Tags

Packet ID	State	Bound To	Field Name		Field Type		Notes
0x67	Play	Client	Length of the array		VarInt		
			Array of tags	Tag type	Array	Identifier	Tag identifier (Vanilla required tags are `minecraft:block`, `minecraft:item`, `minecraft:fluid`, `minecraft:entity_type`, and `minecraft:game_event`)
				Array of Tag		(See below)	

Tags look like:

Field Name		Field Type		Notes
Length		VarInt		Number of elements in the following array
Tags	Tag name	Array	Identifier	
	Count		VarInt	Number of elements in the following array
	Entries		Array of VarInt	Numeric ID of the given type (block, item, etc.).

More information on tags is available at: https://minecraft.gamepedia.com/Tag

And a list of all tags is here: https://minecraft.gamepedia.com/Tag#List_of_tags

Serverbound

Teleport Confirm

Sent by client as confirmation of Player Position And Look.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x00	Play	Server	Teleport ID	VarInt	The ID given by the Player Position And Look packet.

Query Block NBT

Used when `Shift+F3+I` is pressed while looking at a block.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x01	Play	Server	Transaction ID	VarInt	An incremental ID so that the client can verify that the response matches.
			Location	Position	The location of the block to check.

Set Difficulty

Must have at least op level 2 to use. Appears to only be used on singleplayer; the difficulty buttons are still disabled in multiplayer.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x02	Play	Server	New difficulty	Byte	0: peaceful, 1: easy, 2: normal, 3: hard .

Chat Message (serverbound)

Used to send a chat message to the server. The message may not be longer than 256 characters or else the server will kick the client.

If the message starts with a `/`, the server will attempt to interpret it as a command. Otherwise, the server will broadcast the same chat message to all players on the server (including the player that sent the message), prepended with player's name. Specifically, it will respond with a translate chat component, "`chat.type.text`" with the first parameter set to the display name of the player (including some chat component logic to support clicking the name to send a PM) and the second parameter set to the message. See processing chat for more information.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x03	Play	Server	Message	String (256)	The client sends the raw input, not a Chat component.

Client Status

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x04	Play	Server	Action ID	VarInt Enum	See below

Action ID values:

Action ID	Action	Notes
0	Perform respawn	Sent when the client is ready to complete login and when the client is ready to respawn after death.
1	Request stats	Sent when the client opens the Statistics menu.

Client Settings

Sent when the player connects, or when settings are changed.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x05	Play	Server	Locale	String (16)	e.g. `en_GB`.
			View Distance	Byte	Client-side render distance, in chunks.
			Chat Mode	VarInt Enum	0: enabled, 1: commands only, 2: hidden. See processing chat for more information.
			Chat Colors	Boolean	“Colors” multiplayer setting. Can the chat be colored?
			Displayed Skin Parts	Unsigned Byte	Bit mask, see below.
			Main Hand	VarInt Enum	0: Left, 1: Right.
			Enable text filtering	Boolean	Enables filtering of text on signs and written book titles. Currently always false (i.e. the filtering is disabled)
			Allow server listings	Boolean	Servers usually list online players, this option should let you not show up in that list.

Displayed Skin Parts flags:

- Bit 0 (0x01): Cape enabled
- Bit 1 (0x02): Jacket enabled
- Bit 2 (0x04): Left Sleeve enabled
- Bit 3 (0x08): Right Sleeve enabled
- Bit 4 (0x10): Left Pants Leg enabled
- Bit 5 (0x20): Right Pants Leg enabled
- Bit 6 (0x40): Hat enabled

The most significant bit (bit 7, 0x80) appears to be unused.

Tab-Complete (serverbound)

Sent when the client needs to tab-complete a `minecraft:ask_server` suggestion type.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x06	Play	Server	Transaction Id	VarInt	The id of the transaction that the server will send back to the client in the response of this packet. Client generates this and increments it each time it sends another tab completion that doesn't get a response.
			Text	String (32500)	All text behind the cursor without the / (e.g. to the left of the cursor in left-to-right languages like English).

Click Window Button

Used when clicking on window buttons. Until 1.14, this was only used by enchantment tables.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x07	Play	Server	Window ID	Byte	The ID of the window sent by Open Window.
			Button ID	Byte	Meaning depends on window type; see below.

Window type	ID	Meaning
Enchantment Table	0	Topmost enchantment.
	1	Middle enchantment.
	2	Bottom enchantment.
Lectern	1	Previous page (which does give a redstone output).
	2	Next page.
	3	Take Book.
	100+page	Opened page number - 100 + number.
Stonecutter	Recipe button number - 4*row + col. Depends on the item.	
Loom	Recipe button number - 4*row + col. Depends on the item.	

Click Window

This packet is sent by the client when the player clicks on a slot in a window.

Packet ID	State	Bound To	Field Name		Field Type		Notes
0x08	Play	Server	Window ID		Unsigned Byte		The ID of the window which was clicked. 0 for player inventory.
			State ID		VarInt		The last recieved State ID from either a Set Slot or a Window Items packet
			Slot		Short		The clicked slot number, see below.
			Button		Byte		The button used in the click, see below.
			Mode		VarInt Enum		Inventory operation mode, see below.
			Length of the array		VarInt		
			Array of slots	Slot number	Array	Short	
				Slot data		Slot	New data for this slot
			Clicked item		Slot		The clicked slot. Has to be empty (item ID = -1) for drop mode. Is always empty for mode 2 and mode 5 packets.

See Inventory for further information about how slots are indexed.

When right-clicking on a stack of items, half the stack will be picked up and half left in the slot. If the stack is an odd number, the half left in the slot will be smaller of the amounts.

The distinct type of click performed by the client is determined by the combination of the Mode and Button fields.

Mode	Button	Slot	Trigger
0	0	Normal	Left mouse click
	1	Normal	Right mouse click
	0	-999	Left click outside inventory (drop cursor stack)
	1	-999	Right click outside inventory (drop cursor single item)
1	0	Normal	Shift + left mouse click
	1	Normal	Shift + right mouse click *(identical behavior)*
2	0	Normal	Number key 1
	1	Normal	Number key 2
	2	Normal	Number key 3
	⋮	⋮	⋮
	8	Normal	Number key 9
	40	Normal	Offhand swap key F
3	2	Normal	Middle click, only defined for creative players in non-player inventories.
4	0	Normal*	Drop key (Q) (* Clicked item is always empty)
	1	Normal*	Control + Drop key (Q) (* Clicked item is always empty)
5	0	-999	Starting left mouse drag
	4	-999	Starting right mouse drag
	8	-999	Starting middle mouse drag, only defined for creative players in non-player inventories. (Note: the vanilla client will still incorrectly send this for non-creative players - see MC-46584 (https://bugs.mojang.com/browse/MC-46584))
	1	Normal	Add slot for left-mouse drag
	5	Normal	Add slot for right-mouse drag
	9	Normal	Add slot for middle-mouse drag, only defined for creative players in non-player inventories. (Note: the vanilla client will still incorrectly send this for non-creative players - see MC-46584 (https://bugs.mojang.com/browse/MC-46584))
	2	-999	Ending left mouse drag
	6	-999	Ending right mouse drag
	10	-999	Ending middle mouse drag, only defined for creative players in non-player inventories. (Note: the vanilla client will still incorrectly send this for non-creative players - see MC-46584 (https://bugs.mojang.com/browse/MC-46584))
6	0	Normal	Double click

Starting from version 1.5, “painting mode” is available for use in inventory windows. It is done by picking up stack of something (more than 1 item), then holding mouse button (left, right or middle) and dragging held stack over empty (or same type in case of right button) slots. In that case client sends the following to server after mouse button release (omitting first pickup packet which is sent as usual):

1. packet with mode 5, slot -999, button (0 for left | 4 for right);
2. packet for every slot painted on, mode is still 5, button (1 | 5);
3. packet with mode 5, slot -999, button (2 | 6);

If any of the painting packets other than the “progress” ones are sent out of order (for example, a start, some slots, then another start; or a left-click in the middle) the painting status will be reset.

Close Window (serverbound)

This packet is sent by the client when closing a window.

Notchian clients send a Close Window packet with Window ID 0 to close their inventory even though there is never an Open Window packet for the inventory.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x09	Play	Server	Window ID	Unsigned Byte	This is the ID of the window that was closed. 0 for player inventory.

Plugin Message (serverbound)

Main article: Plugin channels

Mods and plugins can use this to send their data. Minecraft itself uses some plugin channels. These internal channels are in the `minecraft` namespace.

More documentation on this: https://dinnerbone.com/blog/2012/01/13/minecraft-plugin-channels-messaging/ (https://dinnerbone.com/blog/2012/01/13/minecraft-plugin-channels-messaging/)

Note that the length of Data is known only from the packet length, since the packet has no length field of any kind.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x0A	Play	Server	Channel	Identifier	Name of the plugin channel used to send the data.
			Data	Byte Array	Any data, depending on the channel. `minecraft:` channels are documented here. The length of this array must be inferred from the packet length.

Edit Book

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x0B	Play	Server	Hand	VarInt enum	0: Main hand, 1: Off hand.
			Count	VarInt	Number of elements in the following array
			Entries	Array of Strings	Text from each page.
			Has title	Boolean	If true, the next field is present.
			Title	String	Title of book.

When editing a draft, the NBT section of the Slot contains this:

```
TAG_Compound(''): 1 entry
{
  TAG_List('pages'): 2 entries
  {
    TAG_String(0): 'Something on Page 1'
    TAG_String(1): 'Something on Page 2'
  }
}
```

When signing the book, it instead looks like this:

```
TAG_Compound(''): 3 entires
{
  TAG_String('author'): 'Steve'
  TAG_String('title'): 'A Wonderful Book'
  TAG_List('pages'): 2 entries
  {
    TAG_String(0): 'Something on Page 1'
    TAG_String(1): 'Something on Page 2'
  }
}
```

Query Entity NBT

Used when `Shift+F3+I` is pressed while looking at an entity.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x0C	Play	Server	Transaction ID	VarInt	An incremental ID so that the client can verify that the response matches.
			Entity ID	VarInt	The ID of the entity to query.

Interact Entity

This packet is sent from the client to the server when the client attacks or right-clicks another entity (a player, minecart, etc).

A Notchian server only accepts this packet if the entity being attacked/used is visible without obstruction and within a 4-unit radius of the player's position.

The target X, Y, and Z fields represent the difference between the vector location of the cursor at the time of the packet and the entity's position.

Note that middle-click in creative mode is interpreted by the client and sent as a Creative Inventory Action packet instead.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x0D	Play	Server	Entity ID	VarInt	The ID of the entity to interact.
			Type	VarInt Enum	0: interact, 1: attack, 2: interact at.
			Target X	Optional Float	Only if Type is interact at.
			Target Y	Optional Float	Only if Type is interact at.
			Target Z	Optional Float	Only if Type is interact at.
			Hand	Optional VarInt Enum	Only if Type is interact or interact at; 0: main hand, 1: off hand.
			Sneaking	Boolean	If the client is sneaking.

Generate Structure

Sent when Generate is pressed on the Jigsaw Block (https://minecraft.fandom.com/wiki/Jigsaw_Block) interface.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x0E	Play	Server	Location	Position	Block entity location.
			Levels	VarInt	Value of the levels slider/max depth to generate.
			Keep Jigsaws	Boolean	

Keep Alive (serverbound)

The server will frequently send out a keep-alive, each containing a random ID. The client must respond with the same packet.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x0F	Play	Server	Keep Alive ID	Long	

Lock Difficulty

Must have at least op level 2 to use. Appears to only be used on singleplayer; the difficulty buttons are still disabled in multiplayer.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x10	Play	Server	Locked	Boolean	

Player Position

Updates the player's XYZ position on the server.

Checking for moving too fast is achieved like this:

- Each server tick, the player's current position is stored
- When a player moves, the changes in x, y, and z coordinates are compared with the positions from the previous tick (Δx, Δy, Δz)
- Total movement distance squared is computed as $\Delta x^2 + \Delta y^2 + \Delta z^2$
- The expected movement distance squared is computed as $velocityX^2 + velocityY^2 + velocityZ^2$
- If the total movement distance squared value minus the expected movement distance squared value is more than 100 (300 if the player is using an elytra), they are moving too fast.

If the player is moving too fast, it will be logged that "<player> moved too quickly! " followed by the change in x, y, and z, and the player will be teleported back to their current (before this packet) serverside position.

Also, if the absolute value of X or the absolute value of Z is a value greater than 3.2×10^7, or X, Y, or Z are not finite (either positive infinity, negative infinity, or NaN), the client will be kicked for “Invalid move player packet received”.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x11	Play	Server	X	Double	Absolute position.
			Feet Y	Double	Absolute feet position, normally Head Y - 1.62.
			Z	Double	Absolute position.
			On Ground	Boolean	True if the client is on the ground, false otherwise.

Player Position And Rotation (serverbound)

A combination of Player Rotation and Player Position.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x12	Play	Server	X	Double	Absolute position.
			Feet Y	Double	Absolute feet position, normally Head Y - 1.62.
			Z	Double	Absolute position.
			Yaw	Float	Absolute rotation on the X Axis, in degrees.
			Pitch	Float	Absolute rotation on the Y Axis, in degrees.
			On Ground	Boolean	True if the client is on the ground, false otherwise.

Player Rotation

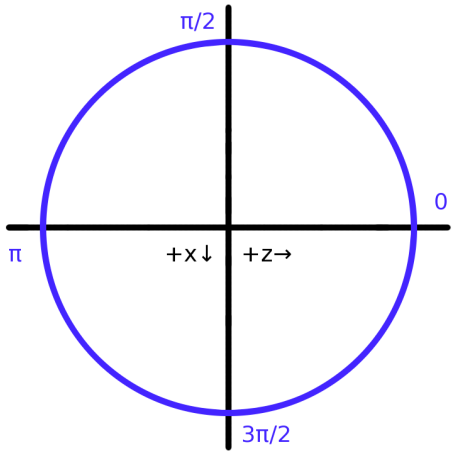
Updates the direction the player is looking in.

Yaw is measured in degrees, and does not follow classical trigonometry rules. The unit circle of yaw on the XZ-plane starts at (0, 1) and turns counterclockwise, with 90 at (-1, 0), 180 at (0,-1) and 270 at (1, 0). Additionally, yaw is not clamped to between 0 and 360 degrees; any number is valid, including negative numbers and numbers greater than 360.

Pitch is measured in degrees, where 0 is looking straight ahead, -90 is looking straight up, and 90 is looking straight down.

The yaw and pitch of player (in degrees), standing at point (x0, y0, z0) and looking towards point (x, y, z) can be calculated with:

```
dx = x-x0
dy = y-y0
dz = z-z0
r = sqrt( dx*dx + dy*dy + dz*dz )
yaw = -atan2(dx,dz)/PI*180
if yaw < 0 then
    yaw = 360 + yaw
pitch = -arcsin(dy/r)/PI*180
```

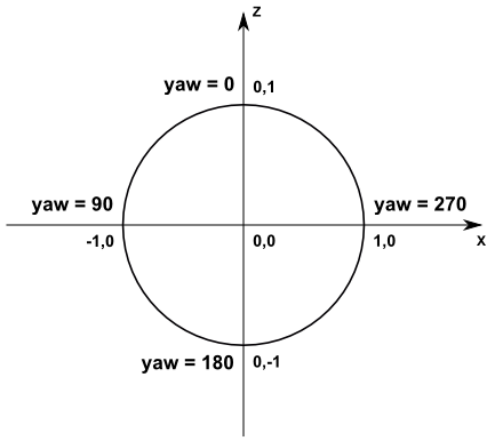


The unit circle for yaw

You can get a unit vector from a given yaw/pitch via:

```
x = -cos(pitch) * sin(yaw)
y = -sin(pitch)
z =  cos(pitch) * cos(yaw)
```

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x13	Play	Server	Yaw	Float	Absolute rotation on the X Axis, in degrees.
			Pitch	Float	Absolute rotation on the Y Axis, in degrees.
			On Ground	Boolean	True if the client is on the ground, false otherwise.



The unit circle of yaw, redrawn

Player Movement

This packet as well as Player Position, Player Look, and Player Position And Look are called the “serverbound movement packets”. Vanilla clients will send Player Position once every 20 ticks even for a stationary player.

This packet is used to indicate whether the player is on ground (walking/swimming), or airborne (jumping/falling).

When dropping from sufficient height, fall damage is applied when this state goes from false to true. The amount of damage applied is based on the point where it last changed from true to false. Note that there are several movement related packets containing this state.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x14	Play	Server	On Ground	Boolean	True if the client is on the ground, false otherwise.

Vehicle Move (serverbound)

Sent when a player moves in a vehicle. Fields are the same as in Player Position And Look. Note that all fields use absolute positioning and do not allow for relative positioning.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x15	Play	Server	X	Double	Absolute position (X coordinate).
			Y	Double	Absolute position (Y coordinate).
			Z	Double	Absolute position (Z coordinate).
			Yaw	Float	Absolute rotation on the vertical axis, in degrees.
			Pitch	Float	Absolute rotation on the horizontal axis, in degrees.

Steer Boat

Used to *visually* update whether boat paddles are turning. The server will update the Boat entity metadata to match the values here.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x16	Play	Server	Left paddle turning	Boolean	
			Right paddle turning	Boolean	

Right paddle turning is set to true when the left button or forward button is held, left paddle turning is set to true when the right button or forward button is held.

Pick Item

Used to swap out an empty space on the hotbar with the item in the given inventory slot. The Notchain client uses this for pick block functionality (middle click) to retrieve items from the inventory.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x17	Play	Server	Slot to use	VarInt	See Inventory.

The server will first search the player's hotbar for an empty slot, starting from the current slot and looping around to the slot before it. If there are no empty slots, it will start a second search from the current slot and find the first slot that does not contain an enchanted item. If there still are no slots that meet that criteria, then the server will use the currently selected slot.

After finding the appropriate slot, the server swaps the items and then send 3 packets:

- Set Slot, with window ID set to -2 and slot set to the newly chosen slot and the item set to the item that is now in that slot (which was previously at the slot the client requested)
- Set Slot, with window ID set to -2 and slot set to the slot the player requested, with the item that is now in that slot and was previously on the hotbar slot
- Held Item Change, with the slot set to the newly chosen slot.

Craft Recipe Request

This packet is sent when a player clicks a recipe in the crafting book that is craftable (white border).

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x18	Play	Server	Window ID	Byte	
			Recipe	Identifier	A recipe ID.
			Make all	Boolean	Affects the amount of items processed; true if shift is down when clicked.

Player Abilities (serverbound)

The vanilla client sends this packet when the player starts/stops flying with the Flags parameter changed accordingly.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x19	Play	Server	Flags	Byte	Bit mask. 0x02: is flying.

Player Digging

Sent when the player mines a block. A Notchian server only accepts digging packets with coordinates within a 6-unit radius between the center of the block and 1.5 units from the player's feet (*not* their eyes).

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x1A	Play	Server	Status	VarInt Enum	The action the player is taking against the block (see below).
			Location	Position	Block position.
			Face	Byte Enum	The face being hit (see below).

Status can be one of seven values:

Value	Meaning	Notes
0	Started digging	
1	Cancelled digging	Sent when the player lets go of the Mine Block key (default: left click).
2	Finished digging	Sent when the client thinks it is finished.
3	Drop item stack	Triggered by using the Drop Item key (default: Q) with the modifier to drop the entire selected stack (default: Control or Command, depending on OS). Location is always set to 0/0/0, Face is always set to -Y.
4	Drop item	Triggered by using the Drop Item key (default: Q). Location is always set to 0/0/0, Face is always set to -Y.
5	Shoot arrow / finish eating	Indicates that the currently held item should have its state updated such as eating food, pulling back bows, using buckets, etc. Location is always set to 0/0/0, Face is always set to -Y.
6	Swap item in hand	Used to swap or assign an item to the second hand. Location is always set to 0/0/0, Face is always set to -Y.

The Face field can be one of the following values, representing the face being hit:

Value	Offset	Face
0	-Y	Bottom
1	+Y	Top
2	-Z	North
3	+Z	South
4	-X	West
5	+X	East

Entity Action

Sent by the client to indicate that it has performed certain actions: sneaking (crouching), sprinting, exiting a bed, jumping with a horse, and opening a horse's inventory while riding it.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x1B	Play	Server	Entity ID	VarInt	Player ID
			Action ID	VarInt Enum	The ID of the action, see below.
			Jump Boost	VarInt	Only used by the “start jump with horse” action, in which case it ranges from 0 to 100. In all other cases it is 0.

Action ID can be one of the following values:

ID	Action
0	Start sneaking
1	Stop sneaking
2	Leave bed
3	Start sprinting
4	Stop sprinting
5	Start jump with horse
6	Stop jump with horse
7	Open horse inventory
8	Start flying with elytra

Leave bed is only sent when the “Leave Bed” button is clicked on the sleep GUI, not when waking up due today time.

Open horse inventory is only sent when pressing the inventory key (default: E) while on a horse — all other methods of opening a horse's inventory (involving right-clicking or shift-right-clicking it) do not use this packet.

Steer Vehicle

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x1C	Play	Server	Sideways	Float	Positive to the left of the player.
			Forward	Float	Positive forward.
			Flags	Unsigned Byte	Bit mask. 0x1: jump, 0x2: unmount.

Also known as 'Input' packet.

Pong

Response to the clientbound packet (Ping) with the same id.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x1D	Play	Server	ID	Int	id is the same as the ping packet

Set Recipe Book State

Replaces Recipe Book Data, type 1.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x1E	Play	Server	Book ID	VarInt enum	0: crafting, 1: furnace, 2: blast furnace, 3: smoker.
			Book Open	Boolean	
			Filter Active	Boolean	

Set Displayed Recipe

Replaces Recipe Book Data, type 0.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x1F	Play	Server	Recipe ID	Identifier	

Name Item

Sent as a player is renaming an item in an anvil (each keypress in the anvil UI sends a new Name Item packet). If the new name is empty, then the item loses its custom name (this is different from setting the custom name to the normal name of the item). The item name may be no longer than 50 characters long, and if it is longer than that, then the rename is silently ignored.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x20	Play	Server	Item name	String (32767)	The new name of the item.

Resource Pack Status

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x21	Play	Server	Result	VarInt Enum	0: successfully loaded, 1: declined, 2: failed download, 3: accepted.

Advancement Tab

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x22	Play	Server	Action	VarInt enum	0: Opened tab, 1: Closed screen.
			Tab ID	Optional identifier	Only present if action is Opened tab.

Select Trade

When a player selects a specific trade offered by a villager NPC.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x23	Play	Server	Selected slot	VarInt	The selected slot in the players current (trading) inventory. (Was a full Integer for the plugin message).

Set Beacon Effect

Changes the effect of the current beacon.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x24	Play	Server	Primary Effect	VarInt	A Potion ID (https://minecraft.gamepedia.com/Data_values#Potions). (Was a full Integer for the plugin message).
			Secondary Effect	VarInt	A Potion ID (https://minecraft.gamepedia.com/Data_values#Potions). (Was a full Integer for the plugin message).

Held Item Change (serverbound)

Sent when the player changes the slot selection

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x25	Play	Server	Slot	Short	The slot which the player has selected (0–8).

Update Command Block

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x26	Play	Server	Location	Position	
			Command	String (32767)	
			Mode	VarInt enum	One of SEQUENCE (0), AUTO (1), or REDSTONE (2).
			Flags	Byte	0x01: Track Output (if false, the output of the previous command will not be stored within the command block); 0x02: Is conditional; 0x04: Automatic.

Update Command Block Minecart

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x27	Play	Server	Entity ID	VarInt	
			Command	String	
			Track Output	Boolean	If false, the output of the previous command will not be stored within the command block.

Creative Inventory Action

While the user is in the standard inventory (i.e., not a crafting bench) in Creative mode, the player will send this packet.

Clicking in the creative inventory menu is quite different from non-creative inventory management. Picking up an item with the mouse actually deletes the item from the server, and placing an item into a slot or dropping it out of the inventory actually tells the server to create the item from scratch. (This can be verified by clicking an item that you don't mind deleting, then severing the connection to the server; the item will be nowhere to be found when you log back in.) As a result of this implementation strategy, the "Destroy Item" slot is just a client-side implementation detail that means "I don't intend to recreate this item.". Additionally, the long listings of items (by category, etc.) are a client-side interface for choosing which item to create. Picking up an item from such listings sends no packets to the server; only when you put it somewhere does it tell the server to create the item in that location.

This action can be described as "set inventory slot". Picking up an item sets the slot to item ID -1. Placing an item into an inventory slot sets the slot to the specified item. Dropping an item (by clicking outside the window) effectively sets slot -1 to the specified item, which causes the server to spawn the item entity, etc.. All other inventory slots are numbered the same as the non-creative inventory (including slots for the 2x2 crafting menu, even though they aren't visible in the vanilla client).

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x28	Play	Server	Slot	Short	Inventory slot.
			Clicked Item	Slot	

Update Jigsaw Block

Sent when Done is pressed on the Jigsaw Block (https://minecraft.fandom.com/wiki/Jigsaw_Block) interface.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x29	Play	Server	Location	Position	Block entity location
			Name	Identifier	
			Target	Identifier	
			Pool	Identifier	
			Final state	String	"Turns into" on the GUI, `final_state` in NBT.
			Joint type	String	`rollable` if the attached piece can be rotated, else `aligned`.

Update Structure Block

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x2A	Play	Server	Location	Position	Block entity location.
			Action	VarInt enum	An additional action to perform beyond simply saving the given data; see below.
			Mode	VarInt enum	One of SAVE (0), LOAD (1), CORNER (2), DATA (3).
			Name	String	
			Offset X	Byte	Between -32 and 32.
			Offset Y	Byte	Between -32 and 32.
			Offset Z	Byte	Between -32 and 32.
			Size X	Byte	Between 0 and 32.
			Size Y	Byte	Between 0 and 32.
			Size Z	Byte	Between 0 and 32.
			Mirror	VarInt enum	One of NONE (0), LEFT_RIGHT (1), FRONT_BACK (2).
			Rotation	VarInt enum	One of NONE (0), CLOCKWISE_90 (1), CLOCKWISE_180 (2), COUNTERCLOCKWISE_90 (3).
			Metadata	String	
			Integrity	Float	Between 0 and 1.
			Seed	VarLong	
			Flags	Byte	0x01: Ignore entities; 0x02: Show air; 0x04: Show bounding box.

Possible actions:

- 0 - Update data
- 1 - Save the structure
- 2 - Load the structure
- 3 - Detect size

The Notchian client uses update data to indicate no special action should be taken (i.e. the done button).

Update Sign

This message is sent from the client to the server when the “Done” button is pushed after placing a sign.

The server only accepts this packet after Open Sign Editor, otherwise this packet is silently ignored.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x2B	Play	Server	Location	Position	Block Coordinates.
			Line 1	String (384)	First line of text in the sign.
			Line 2	String (384)	Second line of text in the sign.
			Line 3	String (384)	Third line of text in the sign.
			Line 4	String (384)	Fourth line of text in the sign.

Animation (serverbound)

Sent when the player's arm swings.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x2C	Play	Server	Hand	VarInt Enum	Hand used for the animation. 0: main hand, 1: off hand.

Spectate

Teleports the player to the given entity. The player must be in spectator mode.

The Notchian client only uses this to teleport to players, but it appears to accept any type of entity. The entity does not need to be in the same dimension as the player; if necessary, the player will be respawned in the right world. If the given entity cannot be found (or isn't loaded), this packet will be ignored. It will also be ignored if the player attempts to teleport to themselves.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x2D	Play	Server	Target Player	UUID	UUID of the player to teleport to (can also be an entity UUID).

Player Block Placement

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x2E	Play	Server	Hand	VarInt Enum	The hand from which the block is placed; 0: main hand, 1: off hand.
			Location	Position	Block position.
			Face	VarInt Enum	The face on which the block is placed (as documented at Player Digging).
			Cursor Position X	Float	The position of the crosshair on the block, from 0 to 1 increasing from west to east.
			Cursor Position Y	Float	The position of the crosshair on the block, from 0 to 1 increasing from bottom to top.
			Cursor Position Z	Float	The position of the crosshair on the block, from 0 to 1 increasing from north to south.
			Inside block	Boolean	True when the player's head is inside of a block.

Upon placing a block, this packet is sent once.

The Cursor Position X/Y/Z fields (also known as in-block coordinates) are calculated using raytracing. The unit corresponds to sixteen pixels in the default resource pack. For example, let's say a slab is being placed against the south face of a full block. The Cursor Position X will be higher if the player was pointing near the right (east) edge of the face, lower if pointing near the left. The Cursor Position Y will be used to determine whether it will appear as a bottom slab (values 0.0–0.5) or as a top slab (values 0.5-1.0). The Cursor Position Z should be 1.0 since the player was looking at the southernmost part of the block.

Inside block is true when a player's head (specifically eyes) are inside of a block's collision. In 1.13 and later versions, collision is rather complicated and individual blocks can have multiple collision boxes. For instance, a ring of vines has a non-colliding hole in the middle. This value is only true when the player is directly in the box. In practice, though, this value is only used by scaffolding to place in front of the player when sneaking inside of it (other blocks will place behind when you intersect with them -- try with glass for instance).

Use Item

Sent when pressing the Use Item key (default: right click) with an item in hand.

Packet ID	State	Bound To	Field Name	Field Type	Notes
0x2F	Play	Server	Hand	VarInt Enum	Hand used for the animation. 0: main hand, 1: off hand.
