PC - 0.30c, 1.7, 1.8, 15w40b, 1.9, 1.9.1-pre2, 1.9.2, 1.9.4, 16w20a, 1.10-pre1, 1.10, 1.10.1, 1.10.2, 16w35a, 1.11, 1.11.2, 17w15a, 17w18b, 1.12-pre4, 1.12, 1.12.1, 1.12.2, 17w50a, 1.13, 1.13.1, 1.13.2-pre1, 1.13.2-pre2, 1.13.2, 1.14, 1.14.1, 1.14.3, 1.14.4, 1.15, 1.15.1, 1.15.2, 20w13b, 20w14a, 1.16-rc1, 1.16, 1.16.1, 1.16.2, 1.16.3, 1.16.4, 1.16.5, 21w07a, 1.17, 1.17.1, 1.18, 1.18.1, 1.18.2, 1.19, 1.19.2, 1.19.3, 1.19.4, 1.20, 1.20.1, 1.20.2, 1.20.3, 1.20.4, 1.20.5, 1.20.6, 1.21, 1.21.1, 1.21.3, 1.21.4, 1.21.5, 1.21.6, 1.21.8
Bedrock - 0.14, 0.15, 1.0, 1.16.201, 1.16.210, 1.16.220, 1.17.0, 1.17.10, 1.17.30, 1.17.40, 1.18.0, 1.18.11, 1.18.30, 1.19.1, 1.19.10, 1.19.20, 1.19.21, 1.19.30, 1.19.40, 1.19.50, 1.19.60, 1.19.62, 1.19.63, 1.19.70, 1.19.80, 1.20.0, 1.20.10, 1.20.15, 1.20.30, 1.20.40, 1.20.50, 1.20.61, 1.20.71, 1.20.80, 1.21.0, 1.21.2, 1.21.20, 1.21.30, 1.21.42, 1.21.50, 1.21.60, 1.21.70, 1.21.80, 1.21.90, 1.21.93, 1.21.100, **1.21.111**

Table of Contents

Key	Name
Type	Rotation
Type	BlockCoordinates
Type	PlayerAttributes
Type	TransactionUseItem
Type	TransactionActions
Type	TransactionLegacy
Type	Transaction
Type	ItemStacks
Type	RecipeIngredient
Type	PotionTypeRecipes
Type	PotionContainerChangeRecipes
Type	Recipes
Type	RecipeUnlockingRequirement
Type	SkinImage
Type	Skin
Type	PlayerRecords
Type	Enchant
Type	EnchantOption
Type	Action
Type	StackRequestSlotInfo
Type	ItemStackRequest
Type	ItemStackResponses
Type	CommandOrigin
Type	TrackedObject
Type	MapDecoration
Type	StructureBlockSettings
Type	EducationSharedResourceURI
Type	EducationExternalLinkSettings
Type	BlockUpdate
Type	TransitionType
Type	MaterialReducer
Type	PermissionLevel
Type	CommandPermissionLevel
Type	CommandPermissionLevelVarint
Type	WindowID

Key	Name
Type	WindowIDVarint
Type	WindowType
Type	ContainerSlotType
Type	SoundType
Type	LegacyEntityType
Type	DeviceOS
Type	AbilityLayers
Type	CameraPresets
Type	DisconnectFailReason
Type	FullContainerName
Type	MovementEffectType
Type	BiomeDefinition
Type	BiomeChunkGeneration
Type	BiomeClimate
Type	BiomeMountainParameters
Type	BiomeSurfaceMaterial
Type	BiomeMesaSurface
Type	BiomeCappedSurface
Type	BiomeMultiNoiseRules
Type	BiomeWeight
Type	BiomeTemperatureWeight
Type	BiomeConsolidatedFeature
Type	BiomeScatterParameter
Type	BiomeCoordinate
Type	BiomeElementData
Type	BiomeOverworldRules
Type	BiomeConditionalTransformation
Type	EaseType
0x1	packet_login
Type	LoginTokens
0x2	packet_play_status
0x3	packet_server_to_client_handshake
0x4	packet_client_to_server_handshake
0x5	packet_disconnect
0x6	packet_resource_packs_info

Key	Name
0x7	packet_resource_pack_stack
0x8	packet_resource_pack_client_response
0x9	packet_text
0xa	packet_set_time
0xb	packet_start_game
0xc	packet_add_player
0xd	packet_add_entity
0xe	packet_remove_entity
0xf	packet_add_item_entity
0x11	packet_take_item_entity
0x12	packet_move_entity
0x13	packet_move_player
0x14	packet_rider_jump
0x15	packet_update_block
0x16	packet_add_painting
0x17	packet_tick_sync
0x18	packet_level_sound_event_old
0x19	packet_level_event
0x1a	packet_block_event
0x1b	packet_entity_event
0x1c	packet_mob_effect
0x1d	packet_update_attributes
0x1e	packet_inventory_transaction
0x1f	packet_mob_equipment
0x20	packet_mob_armor_equipment
0x21	packet_interact
0x22	packet_block_pick_request
0x23	packet_entity_pick_request
0x24	packet_player_action
0x26	packet_hurt_armor
0x27	packet_set_entity_data
0x28	packet_set_entity_motion
0x29	packet_set_entity_link
0x2a	packet_set_health
0x2b	packet_set_spawn_position

Key	Name
0x2c	packet_animate
0x2d	packet_respawn
0x2e	packet_container_open
0x2f	packet_container_close
0x30	packet_player_hotbar
0x31	packet_inventory_content
0x32	packet_inventory_slot
0x33	packet_container_set_data
0x34	packet_crafting_data
0x35	packet_crafting_event
0x36	packet_gui_data_pick_item
0x37	packet_adventure_settings
0x38	packet_block_entity_data
0x39	packet_player_input
0x3a	packet_level_chunk
0x3b	packet_set_commands_enabled
0x3c	packet_set_difficulty
0x3d	packet_change_dimension
0x3e	packet_set_player_game_type
0x3f	packet_player_list
0x40	packet_simple_event
0x41	packet_event
0x42	packet_spawn_experience_orb
0x43	packet_clientbound_map_item_data
0x44	packet_map_info_request
0x45	packet_request_chunk_radius
0x46	packet_chunk_radius_update
0x48	packet_game_rules_changed
0x49	packet_camera
0x4a	packet_boss_event
0x4b	packet_show_credits
0x4c	packet_available_commands
0x4d	packet_command_request
0x4e	packet_command_block_update
0x4f	packet_command_output

Key	Name
0x50	packet_update_trade
0x51	packet_update_equipment
0x52	packet_resource_pack_data_info
0x53	packet_resource_pack_chunk_data
0x54	packet_resource_pack_chunk_request
0x55	packet_transfer
0x56	packet_play_sound
0x57	packet_stop_sound
0x58	packet_set_title
0x59	packet_add_behavior_tree
0x5a	packet_structure_block_update
0x5b	packet_show_store_offer
0x5c	packet_purchase_receipt
0x5d	packet_player_skin
0x5e	packet_sub_client_login
0x5f	packet_initiate_web_socket_connection
0x60	packet_set_last_hurt_by
0x61	packet_book_edit
0x62	packet_npc_request
0x63	packet_photo_transfer
0x64	packet_modal_form_request
0x65	packet_modal_form_response
0x66	packet_server_settings_request
0x67	packet_server_settings_response
0x68	packet_show_profile
0x69	packet_set_default_game_type
0x6a	packet_remove_objective
0x6b	packet_set_display_objective
0x6c	packet_set_score
0x6d	packet_lab_table
0x6e	packet_update_block_synced
0x6f	packet_move_entity_delta
0x70	packet_set_scoreboard_identity
0x71	packet_set_local_player_as_initialized
0x72	packet_update_soft_enum

Key	Name
0x73	packet_network_stack_latency
0x75	packet_script_custom_event
0x76	packet_spawn_particle_effect
0x77	packet_available_entity_identifiers
0x78	packet_level_sound_event_v2
0x79	packet_network_chunk_publisher_update
0x7a	packet_biome_definition_list
0x7b	packet_level_sound_event
0x7c	packet_level_event_generic
0x7d	packet_lectern_update
0x7e	packet_video_stream_connect
0x81	packet_client_cache_status
0x82	packet_on_screen_texture_animation
0x83	packet_map_create_locked_copy
0x84	packet_structure_template_data_export_request
0x85	packet_structure_template_data_export_response
0x86	packet_update_block_properties
0x87	packet_client_cache_blob_status
0x88	packet_client_cache_miss_response
0x89	packet_education_settings
0x8a	packet_emote
0x8b	packet_multiplayer_settings
0x8c	packet_settings_command
0x8d	packet_anvil_damage
0x8e	packet_completed_using_item
0x8f	packet_network_settings
0x90	packet_player_auth_input
0x91	packet_creative_content
0x92	packet_player_enchant_options
0x93	packet_item_stack_request
0x94	packet_item_stack_response
0x95	packet_player_armor_damage
Type	ArmorDamageEntry
0x96	packet_code_builder
0x97	packet_update_player_game_type

Key	Name
0x98	packet_emote_list
0x9a	packet_position_tracking_db_request
0x99	packet_position_tracking_db_broadcast
0x9b	packet_debug_info
0x9c	packet_packet_violation_warning
0x9d	packet_motion_prediction_hints
0x9e	packet_animate_entity
0x9f	packet_camera_shake
0xa0	packet_player_fog
0xa1	packet_correct_player_move_prediction
0xa2	packet_item_registry
0xa3	packet_filter_text_packet
0xa4	packet_debug_renderer
0xa5	packet_sync_entity_property
0xa6	packet_add_volume_entity
0xa7	packet_remove_volume_entity
0xa8	packet_simulation_type
0xa9	packet_npc_dialogue
0xaa	packet_edu_uri_resource_packet
0xab	packet_create_photo
0xac	packet_update_subchunk_blocks
0xad	packet_photo_info_request
Type	HeightMapDataType
Type	SubChunkEntryWithoutCaching
Type	SubChunkEntryWithCaching
0xae	packet_subchunk
0xaf	packet_subchunk_request
0xb0	packet_client_start_item_cooldown
0xb1	packet_script_message
0xb2	packet_code_builder_source
0xb3	packet_ticking_areas_load_status
0xb4	packet_dimension_data
0xb5	packet_agent_action
0xb6	packet_change_mob_property
0xb7	packet_lesson_progress

Key	Name
0xb8	packet_request_ability
0xb9	packet_request_permissions
0xba	packet_toast_request
0xbb	packet_update_abilities
0xbc	packet_update_adventure_settings
0xbd	packet_death_info
0xbe	packet_editor_network
0xbf	packet_feature_registry
0xc0	packet_server_stats
0xc1	packet_request_network_settings
0xc2	packet_game_test_request
0xc3	packet_game_test_results
0xc4	packet_update_client_input_locks
0xc5	packet_client_cheat_ability
0xc6	packet_camera_presets
0xc7	packet_unlocked_recipes
0x12c	packet_camera_instruction
0x12d	packet_compressed_biome_definitions
0x12e	packet_trim_data
0x12f	packet_open_sign
0x130	packet_agent_animation
0x131	packet_refresh_entitlements
0x132	packet_toggle_crafter_slot_request
0x133	packet_set_player_inventory_options
0x134	packet_set_hud
Type	Element
0x135	packet_award_achievement
0x10	packet_server_post_move
0x136	packet_clientbound_close_form
0x138	packet_serverbound_loading_screen
0x139	packet_jigsaw_structure_data
0x13a	packet_current_structure_feature
0x13b	packet_serverbound_diagnostics
0x13c	packet_camera_aim_assist
0x13d	packet_container_registry_cleanup

Key	Name
0x13e	packet_movement_effect
0x13f	packet_set_movement_authority
0x140	packet_camera_aim_assist_presets
0x141	packet_client_camera_aim_assist
0x142	packet_client_movement_prediction_sync
0x143	packet_update_client_options
0x144	packet_player_video_capture
0x145	packet_player_update_entity_overrides
0x146	packet_player_location
0x147	packet_clientbound_controls_scheme
0x148	packet_server_script_debug_drawer
0x149	packet_serverbound_pack_setting_change

Type **TexturePackInfos**

Datatype

Field Name	Field Type		Notes
TexturePackInfos length	li16		
TexturePackInfos array	**Uuid**	uuid	
	Version	string	
	Size	lu64	
	Content Key	string	
	Sub Pack Name	string	
	Content Identity	string	
	Has Scripts	bool	
	Addon Pack	bool	
	Rtx Enabled	bool	
	Cdn Url	string	cdn_url is a URL that the client can use to download the pack instead of

Field Name	Field Type		Notes
			the server sending it in chunks, which it will continue to do if this field is left empty.

Type **ResourcePackIdVersions** Datatype

Field Name	Field Type		Notes
ResourcePackIdVersions length	varint		
ResourcePackIdVersions array	**Uuid**	string	The ID of the resource pack.
	Version	string	The version of the resource pack.
	Name	string	The subpack name of the resource pack.

Type **Experiment** Datatype

Field Name	Field Type	Notes
Name	string	
Enabled	bool	

Type **GameMode** Datatype

Field Name	Field Type	
GameMode	zigzag32 enum	
	0	Survival
	1	Creative
	2	Adventure

Field Name	Field Type	Notes
	3 Survival Spectator	
	4 Creative Spectator	
	5 Fallback	
	6 Spectator	

Type **GameRuleI32**

Datatype

Field Name	Field Type	Notes
Name	string	
Editable	bool	
Type	varint enum	
	1 Bool	
	2 Int	
	3 Float	
Value *if Type*	*Is Bool*	bool
	Is Int	li32
	Is Float	lf32

Type **GameRuleVarint**

Datatype

Field Name	Field Type	Notes
Name	string	
Editable	bool	

Field Name	Field Type	Notes
Type	varint enum	
	1	Bool
	2	Int
	3	Float
Value *if Type*	*Is Bool*	bool
	Is Int	varint
	Is Float	lf32

Type **Blob**

Datatype

CacheBlob represents a blob as used in the client side blob cache protocol. It holds a hash of its data and the full data of it.

Field Name	Field Type	Notes
Hash	lu64	Hash is the hash of the blob. The hash is computed using xxHash, and must be deterministic for the same chunk data.
Payload	ByteArray	Payload is the data of the blob. When sent, the client will associate the Hash of the blob with the Payload in it.

Type **BlockProperties**

Datatype

Field Name	Field Type	Notes
BlockProperties length	varint	
BlockProperties array	**Name**	string
	State	nbt

Type **Itemstates**

Datatype

Field Name	Field Type			Notes
Itemstates length	varint			
Itemstates array	**Name**	string		
	Runtime Id	li16		
	Component Based	bool		
	Version	zigzag32 enum		
		0	Legacy	Version is the version of the item entry which is used by the client to determine how to handle the item entry. It is one of the constants above.
		1	Data Driven	
		2	None	
	Nbt	nbt		Components on the item

Type **ItemExtraDataWithBlockingTick** Datatype

Field Name	Field Type			Notes
Has Nbt	lu16 enum			
	65535	True		
	0	False		
Nbt *if Has Nbt*	*Is True*	**Version**	u8	
		Nbt	lnbt	
	Default	void		
Can Place On length	li32			
Can Place On array	ShortString			

Field Name	Field Type	Notes
Can Destroy length	li32	
Can Destroy array	ShortString	
Blocking Tick	li64	

Type **ItemExtraDataWithoutBlockingTick** Datatype

Field Name	Field Type		Notes
Has Nbt	lu16 enum		
	65535	True	
	0	False	
Nbt *if Has Nbt*	*Is True*	**Version** u8	
		Nbt lnbt	
	Default	void	
Can Place On length	li32		
Can Place On array	ShortString		
Can Destroy length	li32		
Can Destroy array	ShortString		

Type **ItemLegacy** Datatype

Same as below but without a "networkStackID" boolean

Field Name	Field Type		Notes
Network Id	zigzag32		
🔁	*Is 0*	void	

Field Name	Field Type				Notes
if Network Id	*Default*	**Count**	lu16		
		Metadata	varint		
		Block Runtime Id	zigzag32		
		Extra *if Network Id*	*Is /ShieldItemID*	["encapsulated", { "lengthType": "varint", "type": "ItemExtraDataWithBlockingTick" }]	The Shield Item ID is sent in the StartGame packet. It is usually 355 in vanilla.
			Default	["encapsulated", { "lengthType": "varint", "type": "ItemExtraDataWithoutBlockingTick" }]	

Type **Item** Datatype

An "ItemStack" here represents an Item instance. You can think about it like a pointer to an item class. The data for the class gets updated with the data in the item field As of 1.16.220, now functionally the same as Item just without an extra boolean when server auth inventories is disabled.

Field Name	Field Type				Notes
Network Id	zigzag32				
🔁 *if Network Id*	*Is 0*	void			
	Default	**Count**	lu16		
		Metadata	varint		
		Has Stack Id	u8		When server authoritative inventory is enabled, all allocated items have a unique ID used to identify a specifc item instance.
		Stack Id *if Has Stack Id*	*Is 0*	void	StackNetworkID is the network ID of this item *instance*. If the stack is empty, 0 is always written for this field. If not, the field should be set to 1 if the server authoritative inventories are disabled in the StartGame

Field Name	Field Type				Notes
					packet, or to a unique stack ID if it is enabled.
			Default	zigzag32	
		Block Runtime Id	zigzag32		
		Extra *if Network Id*	*Is /ShieldItemID*	["encapsulated", { "lengthType": "varint", "type": "ItemExtraDataWithBlockingTick" }]	The Shield Item ID is sent in the StartGame packet. It is usually 355 in vanilla.
			Default	["encapsulated", { "lengthType": "varint", "type": "ItemExtraDataWithoutBlockingTick" }]	

Type **Vec3i**

Datatype

Field Name	Field Type	Notes
X	zigzag32	
Y	zigzag32	
Z	zigzag32	

Type **Vec3li**

Datatype

Field Name	Field Type	Notes
X	li32	
Y	li32	
Z	li32	

Type **Vec3u**

Datatype

Field Name	Field Type	Notes
X	varint	
Y	varint	
Z	varint	

Type **Vec3f**

Datatype

Field Name	Field Type	Notes
X	lf32	
Y	lf32	
Z	lf32	

Type **Vec2f**

Datatype

Field Name	Field Type	Notes
X	lf32	
Z	lf32	

Type **Vec3fopts**

Datatype

Field Name	Field Type	Notes
X **Optional**	lf32	
Y **Optional**	lf32	
Z	lf32	

Field Name	Field Type	Notes
Optional		

Type **Vec2fopts**

Datatype

Field Name	Field Type	Notes
X **Optional**	lf32	
Y **Optional**	lf32	

Type **MetadataDictionary**

Datatype

Field Name	Field Type			Notes
MetadataDictionary length	varint			
MetadataDictionary array	**Key**	varint enum		
	0	Flags	https://github.com/pmmp/PocketMine-MP/blob/stable/src/pocketmine/entity/Entity.php#L101	
	1	Health		
	2	Variant		
	3	Color		
	4	Nametag		
	5	Owner Eid		
	6	Target Eid		
	7	Air		
	8	Potion Color		
	9	Potion Ambient		

Field Name	Field Type		Notes
	10	Jump Duration	
	11	Hurt Time	
	12	Hurt Direction	
	13	Paddle Time Left	
	14	Paddle Time Right	
	15	Experience Value	
	16	Minecart Display Block	
	17	Minecart Display Offset	
	18	Minecart Has Display	
	19	Horse Type	
	20	Creeper Swell	
	21	Creeper Swell Direction	
	22	Charge Amount	
	23	Enderman Held Runtime Id	
	24	Entity Age	
	26	Player Flags	
	27	Player Index	
	28	Player Bed Position	
	29	Fireball Power X	

Field Name	Field Type		Notes
	30	Fireball Power Y	
	31	Fireball Power Z	
	32	Aux Power	
	33	Fish X	
	34	Fish Z	
	35	Fish Angle	
	36	Potion Aux Value	
	37	Lead Holder Eid	
	38	Scale	
	39	Interactive Tag	
	40	Npc Skin Id	
	41	Url Tag	
	42	Max Airdata Max Air	
	43	Mark Variant	
	44	Container Type	
	45	Container Base Size	
	46	Container Extra Slots Per Strength	
	47	Block Target	
	48	Wither Invulnerable Ticks	
	49	Wither Target 1	
	50	Wither Target 2	

Field Name	Field Type		Notes
	51	Wither Target 3	
	52	Wither Aerial Attack	
	53	Boundingbox Width	
	54	Boundingbox Height	
	55	Fuse Length	
	56	Rider Seat Position	
	57	Rider Rotation Locked	
	58	Rider Max Rotation	
	59	Rider Min Rotation	
	60	Rider Seat Rotation Offset	
	61	Area Effect Cloud Radius	
	62	Area Effect Cloud Waiting	
	63	Area Effect Cloud Particle Id	
	64	Shulker Peek Id	
	65	Shulker Attach Face	
	66	Shulker Attached	
	67	Shulker Attach Pos	

Field Name	Field Type		Notes
	68	Trading Player Eid	
	69	Trading Career	
	70	Has Command Block	
	71	Command Block Command	
	72	Command Block Last Output	
	73	Command Block Track Output	
	74	Controlling Rider Seat Number	
	75	Strength	
	76	Max Strength	
	77	Evoker Spell Casting Color	
	78	Limited Life	
	79	Armor Stand Pose Index	
	80	Ender Crystal Time Offset	
	81	Always Show Nametag	
	82	Color 2	
	83	Name Author	
	84	Score Tag	

Field Name	Field Type		Notes
	85	Balloon Attached Entity	
	86	Pufferfish Size	
	87	Bubble Time	
	88	Agent	
	89	Sitting Amount	
	90	Sitting Amount Previous	
	91	Eating Counter	
	92	Flags Extended	
	93	Laying Amount	
	94	Laying Amount Previous	
	95	Area Effect Cloud Duration	
	96	Area Effect Cloud Spawn Time	
	97	Area Effect Cloud Change Rate	
	98	Area Effect Cloud Change On Pickup	
	99	Area Effect Cloud Pickup Count	
	100	Interact Text	

Field Name	Field Type		Notes
	101	Trade Tier	
	102	Max Trade Tier	
	103	Trade Experience	
	104	Skin Id	
	105	Spawning Frames	
	106	Command Block Tick Delay	
	107	Command Block Execute On First Tick	
	108	Ambient Sound Interval	
	109	Ambient Sound Interval Range	
	110	Ambient Sound Event Name	
	111	Fall Damage Multiplier	
	112	Name Raw Text	
	113	Can Ride Target	
	114	Low Tier Cured Discount	
	115	High Tier Cured Discount	
	116	Nearby Cured Discount	

Field Name	Field Type		Notes
	117	Nearby Cured Discount Timestamp	
	118	Hitbox	
	119	Is Buoyant	
	120	Freezing Effect Strength	
	121	Buoyancy Data	
	122	Goat Horn Count	
	123	Update Properties	
	124	Movement Sound Distance Offset	
	125	Heartbeat Interval Ticks	
	126	Heartbeat Sound Event	
	127	Player Last Death Position	
	128	Player Last Death Dimension	
	129	Player Has Died	
	130	Collision Box	
	131	Visible Mob Effects	
	132	Filtered Name	
	133	Bed Enter Position	
	134	Seat Third Person	

Field Name	Field Type		Notes
		Camera Radius	
	135	Seat Camera Relax Distance Smoothing	

Type

varint enum

0	Byte
1	Short
2	Int
3	Float
4	String
5	Compound
6	Vec3i
7	Long
8	Vec3f

Value

if Key

Is Flags	MetadataFlags1
Is Flags Extended	MetadataFlags2
Is Seat Third Person Camera Radius	lf32
Is Seat Camera Relax Distance Smoothing	lf32

Default

if Type

Is Byte	i8
Is Short	li16
Is Int	zigzag32
Is Float	lf32
Is String	string
Is Compound	nbt
Is Vec3i	vec3i
Is Long	zigzag64
Is Vec3f	vec3f

Type **Link**

Datatype

Field Name	Field Type	Notes
Ridden Entity Id	zigzag64	
Rider Entity Id	zigzag64	
Type	u8	
Immediate	bool	
Rider Initiated	bool	
Angular Velocity	lf32	angular velocity of the vehicle that the rider is riding.

Type **EntityAttributes**

Datatype

Field Name	Field Type		Notes
EntityAttributes length	varint		
EntityAttributes array	**Name**	string	
	Min	lf32	
	Value	lf32	
	Max	lf32	

Type **EntityProperties**

Datatype

Field Name	Field Type		Notes
Ints length	varint		
Ints array	**Index**	varint	
	Value	zigzag32	
Floats length	varint		

Field Name	Field Type	Notes
Floats array	**Index**	varint
	Value	lf32

Type **Rotation**

Datatype

Field Name	Field Type	Notes
Yaw	byterot	
Pitch	byterot	
Head Yaw	byterot	

Type **BlockCoordinates**

Datatype

Field Name	Field Type	Notes
X	zigzag32	
Y	varint	
Z	zigzag32	

Type **PlayerAttributes**

Datatype

Field Name	Field Type	Notes
PlayerAttributes length	varint	
PlayerAttributes array	**Min**	lf32
	Max	lf32
	Current	lf32
	Default Min	lf32
	Default Max	lf32
	Default	lf32

Field Name	Field Type		Notes
	Name	string	
	Modifiers length	varint	
	Modifiers array	**Id**	string
		Name	string
		Amount	lf32
		Operation	li32
		Operand	li32
		Serializable	bool

Type **TransactionUseItem** Datatype

UseItemTransactionData represents an inventory transaction data object sent when the client uses an item on a block. Also used in PlayerAuthoritativeInput packet

Field Name	Field Type		Notes
Action Type	varint enum		
	0	Click Block	ActionType is the type of the UseItem inventory transaction. It is one of the action types found above, and specifies the way the player interacted with the block. Right click item use on a surface like placing a block
	1	Click Air	Start right click and hold style item use or potentially interact with nothing. If it is a usable item like food the server is expected to send a SetActorDataPacket with ActorFlags::USINGITEM along with the transaction response. While using an item, movement speed is slowed which will be reflected in the move vector in Player Auth Input.
	2	Break Block	Block breaking like left click. When using server auth block breaking as specified in StartGamePacket this is never sent. Instead, block actions are supplied in Player Auth Input.
	3	Attack	
Trigger Type	varint enum		

<table>
<tr><th>Field Name</th><th colspan="2">Field Type</th><th>Notes</th></tr>
<tr><td></td><td>0</td><td>Unknown</td><td>TriggerType is the type of the trigger that caused the inventory transaction. It is one of the trigger types found in the constants above. If TriggerType is TriggerTypePlayerInput, the transaction is from the initial input of the player. If it is TriggerTypeSimulationTick, the transaction is from a simulation tick when the player is holding down the input.</td></tr>
<tr><td></td><td>1</td><td>Player Input</td><td></td></tr>
<tr><td></td><td>2</td><td>Simulation Tick</td><td></td></tr>
<tr><td>Block Position</td><td colspan="2">BlockCoordinates</td><td>BlockPosition is the position of the block that was interacted with. This is only really a correct block position if ActionType is not UseItemActionClickAir.</td></tr>
<tr><td>Face</td><td colspan="2">zigzag32</td><td>BlockFace is the face of the block that was interacted with. When clicking the block, it is the face clicked. When breaking the block, it is the face that was last being hit until the block broke.</td></tr>
<tr><td>Hotbar Slot</td><td colspan="2">zigzag32</td><td>HotBarSlot is the hot bar slot that the player was holding while clicking the block. It should be used to ensure that the hot bar slot and held item are correctly synchronised with the server.</td></tr>
<tr><td>Held Item</td><td colspan="2">Item</td><td>HeldItem is the item that was held to interact with the block. The server should check if this item is actually present in the HotBarSlot.</td></tr>
<tr><td>Player Pos</td><td colspan="2">vec3f</td><td>Position is the position of the player at the time of interaction. For clicking a block, this is the position at that time, whereas for breaking the block it is the position at the time of breaking.</td></tr>
<tr><td>Click Pos</td><td colspan="2">vec3f</td><td>ClickedPosition is the position that was clicked relative to the block's base coordinate. It can be used to find out exactly where a player clicked the block.</td></tr>
<tr><td>Block Runtime Id</td><td colspan="2">varint</td><td>BlockRuntimeID is the runtime ID of the block that was clicked. It may be used by the server</td></tr>
</table>

Field Name	Field Type		Notes
			to verify that the player's world client-side is synchronised with the server's.
Client Prediction	varint enum		
	0	Failure	ClientPrediction is the client's prediction on the output of the transaction.
	1	Success	

Type **TransactionActions** Datatype

Actions is a list of actions that took place, that form the inventory transaction together. Each of these actions hold one slot in which one item was changed to another. In general, the combination of all of these actions results in a balanced inventory transaction. This should be checked to ensure that no items are cheated into the inventory.

Field Name	Field Type			
TransactionActions length	varint			
TransactionActions array	**Source Type**	varint enum		
		0	Container	
		1	Global	
		2	World Interaction	
		3	Creative	
		100	Craft Slot	
		99999	Craft	
	🔁 *if Source Type*	*Is Container Or Craft*	**Inventory Id**	WindowIDVarint
		Is World Interaction	**Flags**	varint
		Is Craft Or Craft Slot	**Action**	varint
		Default	void	

Field Name	Field Type	Notes
	Slot	varint
	Old Item	Item
	New Item	Item

Type **TransactionLegacy** Datatype

The Minecraft bedrock inventory system was refactored, but not all inventory actions use the new packet. This data structure holds actions that have not been updated to the new system.

Field Name	Field Type		Notes
Legacy Request Id	zigzag32		LegacyRequestID is an ID that is only non-zero at times when sent by the client. The server should always send 0 for this. When this field is not 0, the LegacySetItemSlots slice below will have values in it. LegacyRequestID ties in with the ItemStackResponse packet. If this field is non-0, the server should respond with an ItemStackResponse packet. Some inventory actions such as dropping an item out of the hotbar are still one using this packet, and the ItemStackResponse packet needs to tie in with it.
Legacy Transactions *if Legacy Request Id*	*Is 0*	void	legacy_transactions are only present if the LegacyRequestID is non-zero. These item slots inform the server of the slots that were changed during the inventory transaction, and the server should send back an ItemStackResponse packet with these slots present in it. (Or false with no slots, if rejected.)
	Default **length**	varint	
	Default array	**Container Id**	u8
		Changed Slots length	varint
		Changed Slots array	**Slot Id** u8

Type **Transaction** Datatype

Field Name	Field Type		Notes
Legacy	TransactionLegacy		Old transaction system data
Transaction Type	varint enum		
	0	Normal	What type of transaction took place Sent for container UI operations depending on if ItemStackNetManager is enabled
	1	Inventory Mismatch	Sent from server to client to reject a transaction
	2	Item Use	Sent for a player performing right click style item use. See the contained ItemUseInventoryTransaction::ActionType for the expected use case.
	3	Item Use On Entity	Sent for a player right clicking on an entity or attacking them. See ItemUseInventoryTransaction::ActionType for which it is.
	4	Item Release	Sent when releasing right click on a chargeable item like a bow or finishing charging like a crossbow. This is different than canceling item use early which would be in Player Auth Input. See ItemReleaseInventoryTransaction::ActionType for which it is.
Actions	TransactionActions		The list of inventory internal actions in this packet, e.g. inventory GUI actions
Transaction Data *if Transaction Type*	*Is Normal Or Inventory Mismatch*	void	Extra data if an intenal inventory transaction did not take place, e.g. use of an item
	Is Item Use	TransactionUseItem	UseItemTransactionData represents an inventory transaction data object sent when the client uses an item on a block.
	Is Item Use On Entity	**Entity Runtime Id** varint64	UseItemOnEntityTransactionData represents an inventory transaction data object sent when the client uses an item on an entity. TargetEntityRuntimeID is the entity runtime ID of the

Field Name	Field Type			Notes
				target that was clicked. It is the runtime ID that was assigned to it in the AddEntity packet.
		Action Type	varint enum	
			0 Interact	ActionType is the type of the UseItemOnEntity inventory transaction. It is one of the action types found in the constants above, and specifies the way the player interacted with the entity. Right click interact with actor.
			1 Attack	Left click style attack of actor or elytra spin attack. Server is expected to deal damage to the entity with visuals.
		Hotbar Slot	zigzag32	HotBarSlot is the hot bar slot that the player was holding while clicking the entity. It should be used to ensure that the hot bar slot and held item are correctly synchronised with the server.
		Held Item	Item	HeldItem is the item that was held to interact with the entity. The server should check if this item is actually present in the HotBarSlot.
		Player Pos	vec3f	Position is the position of the player at the time of clicking the entity.
		Click Pos	vec3f	ClickedPosition is the position that was clicked relative to the entity's base coordinate. It can be used to find out exactly where a player clicked the entity.
	Is Item Release	**Action Type**	varint enum	
			0 Release	ReleaseItemTransactionData represents an inventory transaction data object sent when the client releases the item it was using, for example when stopping while eating or

Field Name	Field Type		Notes		
					stopping the charging of a bow. ActionType is the type of the ReleaseItem inventory transaction. It is one of the action types found in the constants above, and specifies the way the item was released. As of 1.13, the ActionType is always 0. This field can be ignored, because releasing food (by consuming it) or releasing a bow (to shoot an arrow) is essentially the same. Release right click and hold style item use, like firing a bow
			1	Consume	Finish right click and hold style item use, like charging a crossbow
		Hotbar Slot	zigzag32		HotBarSlot is the hot bar slot that the player was holding while releasing the item. It should be used to ensure that the hot bar slot and held item are correctly synchronised with the server.
		Held Item	Item		HeldItem is the item that was released. The server should check if this item is actually present in the HotBarSlot.
		Head Pos	vec3f		HeadPosition is the position of the player's head at the time of releasing the item. This is used mainly for purposes such as spawning eating particles at that position.

Type **RecipeIngredient** Datatype

Field Name	Field Type	Notes
Type	u8 enum	

Field Name	Field Type		Notes
	0	Invalid	
	1	Int Id Meta	DefaultItemDescriptor represents an item descriptor for regular items.
	2	Molang	MoLangItemDescriptor represents an item descriptor for items that use MoLang (e.g. behaviour packs).
	3	Item Tag	ItemTagItemDescriptor represents an item descriptor that uses item tagging. This should be used to reduce duplicative entries for items that can be grouped under a single tag.
	4	String Id Meta	DeferredItemDescriptor represents an item descriptor that uses a namespace and metadata value to identify the item. There is no clear benefit of using this item descriptor.
	5	Complex Alias	ComplexAliasItemDescriptor represents an item descriptor that uses a single name to identify the item. There is no clear benefit of using this item descriptor and only seem to be used for specific recipes.
🔁 *if Type*	*Is Int Id Meta*	**Network Id**	li16	NetworkID is the numerical network ID of the item. This is sometimes a positive ID, and sometimes a negative ID, depending on what item it concerns.
		Metadata *if Network Id*		
			Is 0	void
			Default	li16
	Is Molang	**Expression**	string	Expression represents the MoLang expression used to identify the item/it's associated tag.
		Version	u8	Version represents the version of MoLang to use.
	Is Item Tag	**Tag**	string	Tag represents the tag that the item is part of.

Field Name	Field Type			Notes
	Is String Id Meta	**Name**	string	Name is the name of the item, which is a name like 'minecraft:stick'.
		Metadata	li16	MetadataValue is the metadata value of the item. For some items, this is the damage value, whereas for other items it is simply an identifier of a variant of the item.
	Is Complex Alias	**Name**	string	Name is the name of the item, which is a name like 'minecraft:stick'.
Count	zigzag32			

Type **PotionTypeRecipes** Datatype

Field Name	Field Type	Notes
PotionTypeRecipes length	varint	
PotionTypeRecipes array	**Input Item Id**	zigzag32
	Input Item Meta	zigzag32
	Ingredient Id	zigzag32
	Ingredient Meta	zigzag32
	Output Item Id	zigzag32
	Output Item Meta	zigzag32

Type **PotionContainerChangeRecipes** Datatype

Field Name	Field Type	Notes
PotionContainerChangeRecipes length	varint	
PotionContainerChangeRecipes array	**Input Item Id**	zigzag32
	Ingredient Id	zigzag32

Field Name	Field Type	Notes
	Output Item Id	zigzag32

Type **Recipes** Datatype

Field Name	Field Type		Notes
Recipes length	varint		
Recipes array	**Type**	zigzag32 enum	
		0 Shapeless	
		1 Shaped	
		2 Furnace	
		3 Furnace With Metadata	furnace_with_metadata is a recipe specifically used for furnace-type crafting stations. It is equal to furnace, except it has an input item with a specific metadata value, instead of any metadata value.
		4 Multi	
		5 Shulker Box	
		6 Shapeless Chemistry	
		7 Shaped Chemistry	
		8 Smithing Transform	SmithingTransformRecipe is a recipe specifically used for smithing tables. It has two input items and adds them together, resulting in a new item.
		9 Smithing Trim	
	Recipe *if Type*	*Is Shapeless Or Shulker Box Or Shapeless Chemistry*	**Recipe Id** LatinString
			Input length varint
			Input array RecipeIngredient
			Output length varint

Field Name	Field Type				Notes
			Output array		ItemLegacy
			Uuid		uuid
			Block		string
			Priority		zigzag32
			Unlocking Requirement		RecipeUnlockingRequirement
			Network Id		varint
		Is Shaped Or Shaped Chemistry	**Recipe Id**		LatinString
			Width		zigzag32
			Height		zigzag32
			*Length for Input below is **Width** from above*		2D input array, size of width*height
			Input array	*Length for __831 below is **Height** from above*	
				__831 array	RecipeIngredient
			Output length		varint
			Output array		ItemLegacy
			Uuid		uuid
			Block		string
			Priority		zigzag32
			Assume Symmetry		bool
			Unlocking Requirement		RecipeUnlockingRequirement
			Network Id		varint
		Is Furnace	**Input Id**		zigzag32
			Output		ItemLegacy

Field Name	Field Type				Notes
			Block		string
		Is Furnace With Metadata	**Input Id**		zigzag32
			Input Meta		zigzag32
			Output		ItemLegacy
			Block		string
		Is Multi	**Uuid**		uuid
			Network Id		varint
		Is Smithing Transform	**Recipe Id**	LatinString	RecipeID is a unique ID of the recipe. This ID must be unique amongst all other types of recipes too, but its functionality is not exactly known.
			Template	RecipeIngredient	
			Base	RecipeIngredient	Base is the item that the Addition is being applied to in the smithing table.
			Addition	RecipeIngredient	Addition is the item that is being added to the Base item to result in a modified item.
			Result	ItemLegacy	Result is the resulting item from the two items being added together.
			Tag	string	Tag is a serialized compound tag in the network little endian format.
			Network Id	varint	RecipeNetworkID is a unique ID used to identify the recipe over network. Each recipe must have a unique network ID. Recommended is to just increment a variable for each unique recipe registered. This field must never be 0.

Field Name	Field Type			Notes
		Is Smithing Trim	**Recipe Id**	LatinString
			Template	RecipeIngredient
			Input	RecipeIngredient
			Addition	RecipeIngredient
			Block	string
			Network Id	varint

Type **RecipeUnlockingRequirement** Datatype

Field Name	Field Type		Notes
Context	u8 enum		
	0	None	
	1	Always Unlocked	
	2	Player In Water	
	3	Player Has Many Items	
Ingredients	*Is None* **length**		varint
if Context	*Is None* array		RecipeIngredient

Type **SkinImage** Datatype

Field Name	Field Type	Notes
Width	li32	
Height	li32	
Data	ByteArray	

Datatype

Type **Skin**

Field Name	Field Type	Notes
Skin Id	string	
Play Fab Id	string	
Skin Resource Pack	string	
Skin Data	SkinImage	
Animations length	li32	
Animations array	**Skin Image**	SkinImage
	Animation Type	li32
	Animation Frames	lf32
	Expression Type	lf32
Cape Data	SkinImage	
Geometry Data	string	
Geometry Data Version	string	
Animation Data	string	
Cape Id	string	
Full Skin Id	string	
Arm Size	string	
Skin Color	string	
Personal Pieces length	li32	
Personal Pieces array	**Piece Id**	string
	Piece Type	string
	Pack Id	string
	Is Default Piece	bool
	Product Id	string
Piece Tint Colors length	li32	

Field Name	Field Type	Notes
Piece Tint Colors array	**Piece Type**	string
	Colors length	li32
	Colors array	string
Premium	bool	
Persona	bool	
Cape On Classic	bool	PersonaCapeOnClassicSkin specifies if the skin had a Persona cape (in-game skin creator cape) equipped on a classic skin.
Primary User	bool	
Overriding Player Appearance	bool	

Type **PlayerRecords** Datatype

Field Name	Field Type		Notes
Type	u8 enum		
	0	Add	
	1	Remove	
Records Count	varint		
*Length for Records below is **Records Count** from above*			
Records array	🔁 *if Type*	*Is Add*	**Uuid** uuid
			Entity Unique Id zigzag64
			Username string
			Xbox User Id string

Field Name	Field Type				Notes
			Platform Chat Id	string	
			Build Platform	li32	
			Skin Data	Skin	
			Is Teacher	bool	
			Is Host	bool	
			Is Subclient	bool	
			Player Color	li32	PlayerColour is the colour of the player that is shown in UI elements, currently only used for the player locator bar.
		Is Remove	**Uuid**	uuid	
Verified	*Length for Is Add below is **Records Count** from above*				
if Type	*Is Add* array			bool	

Type **Enchant** Datatype

Field Name	Field Type	Notes
Id	u8	
Level	u8	

Type **EnchantOption** Datatype

Field Name	Field Type	Notes
Cost	varint	

Field Name	Field Type	Notes
Slot Flags	li32	
Equip Enchants length	varint	
Equip Enchants array	Enchant	
Held Enchants length	varint	
Held Enchants array	Enchant	
Self Enchants length	varint	
Self Enchants array	Enchant	
Name	string	
Option Id	zigzag32	

Type **Action** Datatype

Field Name	Field Type	Notes
Action	zigzag32 enum	

0	Start Break
1	Abort Break
2	Stop Break
3	Get Updated Block
4	Drop Item
5	Start Sleeping
6	Stop Sleeping
7	Respawn
8	Jump
9	Start Sprint
10	Stop Sprint

Field Name	Field Type		Notes
	11	Start Sneak	
	12	Stop Sneak	
	13	Creative Player Destroy Block	
	14	Dimension Change Ack	sent when spawning in a different dimension to tell the server we spawned
	15	Start Glide	
	16	Stop Glide	
	17	Build Denied	
	18	Crack Break	
	19	Change Skin	
	20	Set Enchatnment Seed	no longer used
	21	Swimming	
	22	Stop Swimming	
	23	Start Spin Attack	
	24	Stop Spin Attack	
	25	Interact Block	
	26	Predict Break	
	27	Continue Break	
	28	Start Item Use On	
	29	Stop Item Use On	
	30	Handled Teleport	
	31	Missed Swing	
	32	Start Crawling	

Field Name	Field Type		Notes
	33	Stop Crawling	
	34	Start Flying	
	35	Stop Flying	
	36	Received Server Data	Deprecated. Corresponds to Player Auth Input InputData::ClientAckServerData bit 44 Not sent when using server authoritative movement as specified in StartGamePacket. This is now deprecated because only server authoritative movement exists.
	37	Start Using Item	

Type **StackRequestSlotInfo** Datatype

Source and Destination point to the source slot from which Count of the item stack were taken and the destination slot to which this item was moved.

Field Name	Field Type	Notes
Slot Type	FullContainerName	ContainerID is the ID of the container that the slot was in.
Slot	u8	Slot is the index of the slot within the container with the ContainerID above.
Stack Id	zigzag32	StackNetworkID is the unique stack ID that the client assumes to be present in this slot. The server must check if these IDs match. If they do not match, servers should reject the stack request that the action holding this info was in.

Type **ItemStackRequest** Datatype

ItemStackRequest is sent by the client to change item stacks in an inventory. It is essentially a replacement of the InventoryTransaction packet added in 1.16 for inventory specific actions, such as moving items around or crafting. The InventoryTransaction packet is still used for actions such as placing blocks and interacting with entities.

Field Name	Field Type	Notes		
Request Id	zigzag32	RequestID is a unique ID for the request. This ID is used by the server to send a response for this specific request in the ItemStackResponse packet.		
Actions length	varint			
Actions array	**Type Id**	u8 enum		
		0	Take	TakeStackRequestAction is sent by the client to the server to take x amount of items from one slot in a container to the cursor.
		1	Place	PlaceStackRequestAction is sent by the client to the server to place x amount of items from one slot into another slot, such as when shift clicking an item in the inventory to move it around or when moving an item in the cursor into a slot.
		2	Swap	SwapStackRequestAction is sent by the client to swap the item in its cursor with an item present in another container. The two item stacks swap places.
		3	Drop	DropStackRequestAction is sent by the client when it drops an item out of the inventory when it has its inventory opened. This action is not sent when a player drops an item out of the hotbar using the Q button (or the equivalent on mobile). The InventoryTransaction packet is still used for that action, regardless of whether the item stack network IDs are used or not.
		4	Destroy	DestroyStackRequestAction is sent by the client when it destroys an item in creative mode by moving it back into the creative inventory.
		5	Consume	ConsumeStackRequestAction is sent by the client when it uses an item to craft another item. The original item is 'consumed'.
		6	Create	CreateStackRequestAction is sent by the client when an item is created through being used as part of a recipe. For example, when milk is used to craft a cake, the buckets are leftover. The buckets are moved to the slot sent by the client here. Note that before this is sent, an action for consuming all items in the crafting

Field Name	Field Type	Notes		
				table/grid is sent. Items that are not fully consumed when used for a recipe should not be destroyed there, but instead, should be turned into their respective resulting items.
		7	Place In Container	(as of 1.18.10) Not currently used
		8	Take Out Container	(as of 1.18.10) Not currently used
		9	Lab Table Combine	LabTableCombineStackRequestAction is sent by the client when it uses a lab table to combine item stacks.
		10	Beacon Payment	BeaconPaymentStackRequestAction is sent by the client when it submits an item to enable effects from a beacon. These items will have been moved into the beacon item slot in advance.
		11	Mine Block	MineBlockStackRequestAction is sent by the client when it breaks a block.
		12	Craft Recipe	CraftRecipeStackRequestAction is sent by the client the moment it begins crafting an item. This is the first action sent, before the Consume and Create item stack request actions. This action is also sent when an item is enchanted. Enchanting should be treated mostly the same way as crafting, where the old item is consumed.
		13	Craft Recipe Auto	AutoCraftRecipeStackRequestAction is sent by the client similarly to the CraftRecipeStackRequestAction. The only difference is that the recipe is automatically created and crafted by shift clicking the recipe book.
		14	Craft Creative	CraftCreativeStackRequestAction is sent by the client when it takes an item out fo the creative inventory. The item is thus not really crafted, but instantly created.
		15	Optional	CraftRecipeOptionalStackRequestAction is sent when using an anvil. When this action is sent, the CustomNames field in the respective stack request is non-empty and contains the name of the item created using the anvil.

Field Name	Field Type	Notes		
		16	Craft Grindstone Request	CraftGrindstoneRecipeStackRequestAction is sent when a grindstone recipe is crafted. It contains the RecipeNetworkID to identify the recipe crafted, and the cost for crafting the recipe.
		17	Craft Loom Request	CraftLoomRecipeStackRequestAction is sent when a loom recipe is crafted. It simply contains the pattern identifier to figure out what pattern is meant to be applied to the item.
		18	Non Implemented	CraftNonImplementedStackRequestAction is an action sent for inventory actions that aren't yet implemented in the new system. These include, for example, anvils.
		19	Results Deprecated	CraftResultsDeprecatedStackRequestAction is an additional, deprecated packet sent by the client after crafting. It holds the final results and the amount of times the recipe was crafted. It shouldn't be used. This action is also sent when an item is enchanted. Enchanting should be treated mostly the same way as crafting, where the old item is consumed.
	🔁 *if Type Id*	*Is Take Or Place Or Place In Container Or Take Out Container*	**Count**	u8
			Source	StackRequestSlotInfo
			Destination	StackRequestSlotInfo
		Is Swap	**Source**	StackRequestSlotInfo — Source and Destination point to the source slot from which Count of the item stack were taken and the destination slot to which this item was moved.
			Destination	StackRequestSlotInfo
		Is Drop	**Count**	u8 — Count is the count of the item in the

Field Name	Field Type	Notes			
					source slot that was taken towards the destination slot.
			Source	StackRequestSlotInfo	Source is the source slot from which items were dropped to the ground.
			Randomly	bool	Randomly seems to be set to false in most cases. I'm not entirely sure what this does, but this is what vanilla calls this field.
		Is Destroy Or Consume	**Count**	u8	Count is the count of the item in the source slot that was destroyed.
			Source	StackRequestSlotInfo	Source is the source slot from which items came that were destroyed by moving them into the creative inventory.
		Is Create	**Result Slot Id**	u8	ResultsSlot is the slot in the inventory in which the results of the crafting ingredients are to be placed.
		Is Beacon Payment	**Primary Effect**	zigzag32	PrimaryEffect and SecondaryEffect are the effects that were selected from the beacon.

Field Name	Field Type	Notes			
			Secondary Effect	zigzag32	
		Is Mine Block	**Hotbar Slot**	zigzag32	Current hotbar slot
			Predicted Durability	zigzag32	PredictedDurability is the durability of the item that the client assumes to be present at the time
			Network Id	zigzag32	StackNetworkID is the unique stack ID that the client assumes to be present at the time. The server must check if these IDs match. If they do not match, servers should reject the stack request that the action holding this info was in.
		Is Craft Recipe	**Recipe Network Id**	varint	RecipeNetworkID is the network ID of the recipe that is about to be crafted. This network ID matches one of the recipes sent in the CraftingData packet, where each of the recipes have a RecipeNetworkID as of 1.16.
			Times Crafted	u8	
		Is Craft Recipe Auto	**Recipe Network Id**	varint	RecipeNetworkID is the network ID of the recipe that is about to be crafted. This network ID matches one of the recipes sent in the CraftingData packet, where each of the recipes have a RecipeNetworkID as

Field Name	Field Type	Notes			
			Times Crafted 2	u8	TODO: Duplicate field?
			Times Crafted	u8	
			Ingredients length	varint	
			Ingredients array	RecipeIngredient	
		Is Craft Creative	**Item Id**	varint	The stack ID of the creative item that is being created. This is one of the creative item stack IDs sent in the CreativeContent packet.
			Times Crafted	u8	
		Is Optional	**Recipe Network Id**	varint	For the cartography table, if a certain MULTI recipe is being called, this points to the network ID that was assigned.
			Filtered String Index	li32	Most likely the index in the request's filter strings that this action is using
		Is Craft Grindstone Request	**Recipe Network Id**	varint	RecipeNetworkID is the network ID of the recipe that is about to be crafted. This network ID matches one of the recipes sent in the CraftingData packet, where each of the recipes have a RecipeNetworkID as
			Times Crafted	u8	NumberOfCrafts is how many times the recipe was crafted. This field appears to be boilerplate and has no effect.
			Cost	varint	Cost is the cost of the recipe that was crafted.

<table>
<tr><th>Field Name</th><th>Field Type</th><th colspan="4">Notes</th></tr>
<tr><td></td><td></td><td>Is Craft Loom Request</td><td>Pattern</td><td>string</td><td>Pattern is the pattern identifier for the loom recipe.</td></tr>
<tr><td></td><td></td><td></td><td>Times Crafted</td><td>u8</td><td>TimesCrafted is how many times the recipe was crafted.</td></tr>
<tr><td></td><td></td><td>Is Non Implemented</td><td colspan="3">void</td></tr>
<tr><td></td><td></td><td rowspan="3">Is Results Deprecated</td><td colspan="2">Result Items length</td><td>varint</td></tr>
<tr><td></td><td></td><td colspan="2">Result Items
array</td><td>ItemLegacy</td></tr>
<tr><td></td><td></td><td colspan="2">Times Crafted</td><td>u8</td></tr>
<tr><td>Custom Names length</td><td>varint</td><td colspan="4" rowspan="2">CustomNames is a list of custom names involved in the request. This is typically filled with one string when an anvil is used.
• Used for the server to determine which strings should be filtered. Used in anvils to verify a renamed item.</td></tr>
<tr><td>Custom Names
array</td><td>string</td></tr>
<tr><td>Cause</td><td colspan="5">li32 enum</td></tr>
<tr><td></td><td>0</td><td>Chat Public</td><td colspan="3">FilterCause represents the cause of any potential filtering. This is one of the constants above.</td></tr>
<tr><td></td><td>1</td><td>Chat Whisper</td><td colspan="3" rowspan="11"></td></tr>
<tr><td></td><td>2</td><td>Sign Text</td></tr>
<tr><td></td><td>3</td><td>Anvil Text</td></tr>
<tr><td></td><td>4</td><td>Book And Quill Text</td></tr>
<tr><td></td><td>5</td><td>Command Block Text</td></tr>
<tr><td></td><td>6</td><td>Block Actor Data Text</td></tr>
<tr><td></td><td>7</td><td>Join Event Text</td></tr>
<tr><td></td><td>8</td><td>Leave Event Text</td></tr>
<tr><td></td><td>9</td><td>Slash Command Chat</td></tr>
<tr><td></td><td>10</td><td>Cartography Text</td></tr>
</table>

Field Name	Field Type		Notes
	11	Kick Command	
	12	Title Command	
	13	Summon Command	

Type **ItemStackResponses** Datatype

ItemStackResponse is a response to an individual ItemStackRequest.

Field Name	Field Type					Notes
ItemStackResponses length	varint					
ItemStackResponses array	**Status**	u8 enum				
		0	Ok			Status specifies if the request with the RequestID below v successful. If this is the case, the ContainerInfo below will information on what slots ended up changing. If not, the cor info will be empty. A non-0 status means an error occurred a result in the action being reverted.
		1	Error			
	Request Id	zigzag32				RequestID is the unique ID of the request that this respons reaction to. If rejected, the client will undo the actions fro request with this ID.
	🔁 *if Status*	*Is Ok*	**Containers length**	varint		ContainerInfo holds information on the cont that had their contents changed as a result request.
			Containers array	**Slot Type**	FullContainerName	ContainerID is container ID of container that the that follow are in. I main inventory, this seems to be 0x1I the cursor, this v seems to be 0x3a the crafting grid, value seems to be • actually, this ContainerSlotT used by the inv

Field Name	Field Type						Notes
							system that sp the type of s
				Slots length	varint		SlotInfo hold information on wh stack should be p in specific slots i container.
				Slots array	**Slot**	u8	Slot and Hotb seem to be the value every The slot that actually chang not sure if th slots ever d
					Hotbar Slot	u8	
					Count	u8	Count is the count of the stack. This cou be shown clie after the respo sent to the c
					Item Stack Id	zigzag32	StackNetwor the network ID new stack a specific sl
					Custom Name	string	CustomName custom name item stack. It i in relation to filtering.
					Filtered Custom Name	string	FilteredCuston is a filtered ve of CustomNam all the profa removed. The will use this CustomName field is not e and they hav "Filter Profa setting enab
					Durability Correction	zigzag32	DurabilityCorr is the curre

Field Name	Field Type						Notes
							durability of th stack. This du will be shown side after t response is s the client

Type **CommandOrigin** Datatype

Field Name	Field Type		Notes
Type	varint enum		
	0	Player	Origin is one of the values above that specifies the origin of the command. The origin may change, depending on what part of the client actually called the command. The command may be issued by a websocket server, for example.
	1	Block	
	2	Minecart Block	
	3	Dev Console	
	4	Test	
	5	Automation Player	
	6	Client Automation	
	7	Dedicated Server	
	8	Entity	
	9	Virtual	
	10	Game Argument	
	11	Entity Server	
	12	Precompiled	

Field Name	Field Type		Notes
	13	Game Director Entity Server	
	14	Script	
	15	Executor	
Uuid	uuid		UUID is the UUID of the command called. This UUID is a bit odd as it is not specified by the server. It is not clear what exactly this UUID is meant to identify, but it is unique for each command called.
Request Id	string		RequestID is an ID that identifies the request of the client. The server should send a CommandOrigin with the same request ID to ensure it can be matched with the request by the caller of the command. This is especially important for websocket servers and it seems that this field is only non-empty for these websocket servers.
Player Entity Id *if Type*	*Is Dev Console Or Test*	**Player Entity Id**	zigzag64	PlayerUniqueID is an ID that identifies the player, the same as the one found in the AdventureSettings packet. Filling it out with 0 seems to work. PlayerUniqueID is only written if Origin is CommandOriginDevConsole or CommandOriginTest.

Type **TrackedObject** Datatype

MapTrackedObject is an object on a map that is 'tracked' by the client, such as an entity or a block. This object may move, which is handled client-side.

Field Name	Field Type		Notes
Type	li32 enum		
	0	Entity	Type is the type of the tracked object. It is either MapObjectTypeEntity or MapObjectTypeBlock.
	1	Block	

Field Name	Field Type		Notes
Entity Unique Id *if Type*	*Is Entity*	zigzag64	EntityUniqueID is the unique ID of the entity, if the tracked object was an entity. It needs not to be filled out if Type is not MapObjectTypeEntity.
Block Position *if Type*	*Is Block*	BlockCoordinates	BlockPosition is the position of the block, if the tracked object was a block. It needs not to be filled out if Type is not MapObjectTypeBlock.

Type **MapDecoration** Datatype

MapDecoration is a fixed decoration on a map: Its position or other properties do not change automatically client-side.

Field Name	Field Type	Notes
Type	u8 enum	
	0	Marker White
	1	Marker Green
	2	Marker Red
	3	Marker Blue
	4	Cross White
	5	Triangle Red
	6	Square White
	7	Marker Sign
	8	Marker Pink
	9	Marker Orange
	10	Marker Yellow
	11	Marker Teal
	12	Triangle Green
	13	Small Square White

Field Name	Field Type	Notes
	14	Mansion
	15	Monument
	16	No Draw
	17	Village Desert
	18	Village Plains
	19	Village Savanna
	20	Village Snowy
	21	Village Taiga
	22	Jungle Temple
	23	Witch Hut =>
	24	Marker White
	25	Marker Green
	26	Marker Red
	27	Marker Blue
	28	Cross White
	29	Triangle Red
	30	Square White
	31	Marker Sign
	32	Marker Pink
	33	Marker Orange
	34	Marker Yellow
	35	Marker Teal
	36	Triangle Green
	37	Small Square White
	38	Mansion
	39	Monument
	40	No Draw
	41	Village Desert
	42	Village Plains

Field Name	Field Type	Notes
	43	Village Savanna
	44	Village Snowy
	45	Village Taiga
	46	Jungle Temple
	47	Witch Hut
Rotation	u8	Rotation is the rotation of the map decoration. It is byte due to the 16 fixed directions that the map decoration may face.
X	u8	X is the offset on the X axis in pixels of the decoration.
Y	u8	Y is the offset on the Y axis in pixels of the decoration.
Label	string	Label is the name of the map decoration. This name may be of any value.
Color Abgr	varint	Colour is the colour of the map decoration. Some map decoration types have a specific colour set automatically, whereas others may be changed.

Type **StructureBlockSettings** Datatype

Field Name	Field Type	Notes
Palette Name	string	PaletteName is the name of the palette used in the structure. Currently, it seems that this field is always 'default'.
Ignore Entities	bool	IgnoreEntities specifies if the structure should ignore entities or include them. If set to false, entities will also show up in the exported structure.
Ignore Blocks	bool	IgnoreBlocks specifies if the structure should ignore blocks or include them. If set to false, blocks will show up in the exported structure.

Field Name	Field Type	Notes
Non Ticking Players And Ticking Areas	bool	
Size	BlockCoordinates	Size is the size of the area that is about to be exported. The area exported will start at the Position + Offset, and will extend as far as Size specifies.
Structure Offset	BlockCoordinates	Offset is the offset position that was set in the structure block. The area exported is offset by this position. **TODO**: This will be renamed to offset soon
Last Editing Player Unique Id	zigzag64	LastEditingPlayerUniqueID is the unique ID of the player that last edited the structure block that these settings concern.
Rotation	u8 enum	
	0	None	Rotation is the rotation that the structure block should obtain. See the constants above for available options.
	1	90 Deg	
	2	180 Deg	
	3	270 Deg	
Mirror	u8 enum	
	0	None	Mirror specifies the way the structure should be mirrored. It is either no mirror at all, mirror on the x/z axis or both.
	1	X Axis	
	2	Z Axis	
	3	Both Axes	
Animation Mode	u8 enum	
	0	None
	1	Layers

Field Name	Field Type	Notes
	2	Blocks
Animation Duration	lf32	How long the duration for this animation is
Integrity	lf32	Integrity is usually 1, but may be set to a number between 0 and 1 to omit blocks randomly, using the Seed that follows.
Seed	lu32	Seed is the seed used to omit blocks if Integrity is not equal to one. If the Seed is 0, a random seed is selected to omit blocks.
Pivot	vec3f	Pivot is the pivot around which the structure may be rotated.

Type

EducationSharedResourceURI

Datatype

EducationSharedResourceURI is an education edition feature that is used for transmitting education resource settings to clients. It contains a button name and a link URL.

Field Name	Field Type	Notes
Button Name	string	ButtonName is the button name of the resource URI.
Link Uri	string	LinkURI is the link URI for the resource URI.

Type

EducationExternalLinkSettings

Datatype

Field Name	Field Type	Notes
Url	string	URL is the external link URL.
Display Name	string	DisplayName is the display name in game.

Type **BlockUpdate** Datatype

Field Name	Field Type	Notes
Position	BlockCoordinates	
Runtime Id	varint	
Flags	varint	
Entity Unique Id	zigzag64	EntityUniqueID is the unique ID of the falling block entity that the block transitions to or that the entity transitions from. Note that for both possible values for TransitionType, the EntityUniqueID should point to the falling block entity involved.
Transition Type	varint enum	

Type **TransitionType** Datatype

TransitionType is the type of the transition that happened. It is either BlockToEntityTransition, when a block placed becomes a falling entity, or EntityToBlockTransition, when a falling entity hits the ground and becomes a solid block again.

Field Name	Field Type		Notes
TransitionType	enum		
	0	Entity	For falling sand, when a sand turns to an entity
	1	Create	When sand turns back to a new block
	2	Destroy	

Type **MaterialReducer** Datatype

Field Name	Field Type	Notes
Mix	zigzag32	

Field Name	Field Type	Notes
Items	**Network Id**	zigzag32
	Count	zigzag32

Type **PermissionLevel** Datatype

The permission level of a player, for example whether they are an Server Operator or not.

Field Name	Field Type	Notes
PermissionLevel	u8 enum	
	0	Visitor
	1	Member
	2	Operator
	3	Custom

Type **CommandPermissionLevel** Datatype

The command permission level, for example if being run by a Player, an Op, or a Command Block.

Field Name	Field Type	Notes
CommandPermissionLevel	u8 enum	
	0	Normal
	1	Operator
	2	Automation
	3	Host
	4	Owner
	5	Internal

Type **CommandPermissionLevelVarint** Datatype

The command permission level, for example if being run by a Player, an Op, or a Command Block.

Field Name	Field Type	Notes
CommandPermissionLevelVarint	u8 enum	
	0 Normal	
	1 Operator	
	2 Automation	
	3 Host	
	4 Owner	
	5 Internal	

Type **WindowID** Datatype

List of Window IDs. When a new container is opened (container_open), a new sequential Window ID is created. Below window IDs are hard-coded and created when the game starts and the server does not send a container_open for them.

Field Name	Field Type	Notes
WindowID	i8 enum	
	-100 Drop Contents	
	-24 Beacon	
	-23 Trading Output	
	-22 Trading Use Inputs	
	-21 Trading Input 2	
	-20 Trading Input 1	
	-17 Enchant Output	
	-16 Enchant Material	
	-15 Enchant Input	
	-13 Anvil Output	

Field Name	Field Type	Notes
	-12	Anvil Result
	-11	Anvil Material
	-10	Container Input
	-5	Crafting Use Ingredient
	-4	Crafting Result
	-3	Crafting Remove Ingredient
	-2	Crafting Add Ingredient
	-1	None
	0	Inventory
	1	First
	100	Last
	119	Offhand
	120	Armor
	121	Creative
	122	Hotbar
	123	Fixed Inventory
	124	Ui

Type **WindowIDVarint**

Datatype

Field Name	Field Type	Notes
WindowIDVarint	varint enum	
	-100	Drop Contents
	-24	Beacon
	-23	Trading Output
	-22	Trading Use Inputs
	-21	Trading Input 2
	-20	Trading Input 1

Field Name	Field Type		Notes
	-17	Enchant Output	
	-16	Enchant Material	
	-15	Enchant Input	
	-13	Anvil Output	
	-12	Anvil Result	
	-11	Anvil Material	
	-10	Container Input	
	-5	Crafting Use Ingredient	
	-4	Crafting Result	
	-3	Crafting Remove Ingredient	
	-2	Crafting Add Ingredient	
	-1	None	
	0	Inventory	
	1	First	
	100	Last	
	119	Offhand	
	120	Armor	
	121	Creative	
	122	Hotbar	
	123	Fixed Inventory	
	124	Ui	

Type **WindowType**

Datatype

Field Name	Field Type		Notes
WindowType	i8 enum		
	-9	None	
	-1	Inventory	

Field Name	Field Type		Notes
	0	Container	
	1	Workbench	
	2	Furnace	
	3	Enchantment	
	4	Brewing Stand	
	5	Anvil	
	6	Dispenser	
	7	Dropper	
	8	Hopper	
	9	Cauldron	
	10	Minecart Chest	
	11	Minecart Hopper	
	12	Horse	
	13	Beacon	
	14	Structure Editor	
	15	Trading	
	16	Command Block	
	17	Jukebox	
	18	Armor	
	19	Hand	
	20	Compound Creator	
	21	Element Constructor	
	22	Material Reducer	
	23	Lab Table	
	24	Loom	
	25	Lectern	
	26	Grindstone	
	27	Blast Furnace	
	28	Smoker	

Field Name	Field Type	Notes
	29 Stonecutter	
	30 Cartography	
	31 Hud	
	32 Jigsaw Editor	
	33 Smithing Table	
	34 Chest Boat	
	35 Decorated Pot	
	36 Crafter	

Type **ContainerSlotType**

Datatype

Used in inventory transactions.

Field Name	Field Type	Notes
ContainerSlotType	u8 enum	
	0 Anvil Input	
	1 Anvil Material	
	2 Anvil Result	
	3 Smithing Table Input	
	4 Smithing Table Material	
	5 Smithing Table Result	
	6 Armor	
	7 Container	
	8 Beacon Payment	
	9 Brewing Input	
	10 Brewing Result	
	11 Brewing Fuel	
	12 Hotbar And Inventory	
	13 Crafting Input	

Field Name	Field Type		Notes
	14	Crafting Output	
	15	Recipe Construction	
	16	Recipe Nature	
	17	Recipe Items	
	18	Recipe Search	
	19	Recipe Search Bar	
	20	Recipe Equipment	
	21	Recipe Book	
	22	Enchanting Input	
	23	Enchanting Lapis	
	24	Furnace Fuel	
	25	Furnace Ingredient	
	26	Furnace Output	
	27	Horse Equip	
	28	Hotbar	
	29	Inventory	
	30	Shulker	
	31	Trade Ingredient1	
	32	Trade Ingredient2	
	33	Trade Result	
	34	Offhand	
	35	Compcreate Input	
	36	Compcreate Output	
	37	Elemconstruct Output	
	38	Matreduce Input	
	39	Matreduce Output	
	40	Labtable Input	
	41	Loom Input	
	42	Loom Dye	

Field Name	Field Type	Notes
	43 Loom Material	
	44 Loom Result	
	45 Blast Furnace Ingredient	
	46 Smoker Ingredient	
	47 Trade2 Ingredient1	
	48 Trade2 Ingredient2	
	49 Trade2 Result	
	50 Grindstone Input	
	51 Grindstone Additional	
	52 Grindstone Result	
	53 Stonecutter Input	
	54 Stonecutter Result	
	55 Cartography Input	
	56 Cartography Additional	
	57 Cartography Result	
	58 Barrel	
	59 Cursor	
	60 Creative Output	
	61 Smithing Table Template	
	62 Crafter	
	63 Dynamic	
	64 Registry	

Type **SoundType**

Datatype

Field Name	Field Type	Notes
SoundType	varint enum	
	0 ItemUseOn	

Field Name	Field Type		Notes
	1	Hit	
	2	Step	
	3	Fly	
	4	Jump	
	5	Break	
	6	Place	
	7	HeavyStep	
	8	Gallop	
	9	Fall	
	10	Ambient	
	11	AmbientBaby	
	12	AmbientInWater	
	13	Breathe	
	14	Death	
	15	DeathInWater	
	16	DeathToZombie	
	17	Hurt	
	18	HurtInWater	
	19	Mad	
	20	Boost	
	21	Bow	
	22	SquishBig	
	23	SquishSmall	
	24	FallBig	
	25	FallSmall	
	26	Splash	
	27	Fizz	
	28	Flap	
	29	Swim	

Field Name	Field Type		Notes
	30	Drink	
	31	Eat	
	32	Takeoff	
	33	Shake	
	34	Plop	
	35	Land	
	36	Saddle	
	37	Armor	
	38	ArmorStandPlace	
	39	AddChest	
	40	Throw	
	41	Attack	
	42	AttackNoDamage	
	43	AttackStrong	
	44	Warn	
	45	Shear	
	46	Milk	
	47	Thunder	
	48	Explode	
	49	Fire	
	50	Ignite	
	51	Fuse	
	52	Stare	
	53	Spawn	
	54	Shoot	
	55	BreakBlock	
	56	Launch	
	57	Blast	
	58	LargeBlast	

Field Name	Field Type		Notes
	59	Twinkle	
	60	Remedy	
	61	Unfect	
	62	LevelUp	
	63	BowHit	
	64	BulletHit	
	65	ExtinguishFire	
	66	ItemFizz	
	67	ChestOpen	
	68	ChestClosed	
	69	ShulkerBoxOpen	
	70	ShulkerBoxClosed	
	71	EnderChestOpen	
	72	EnderChestClosed	
	73	PowerOn	
	74	PowerOff	
	75	Attach	
	76	Detach	
	77	Deny	
	78	Tripod	
	79	Pop	
	80	DropSlot	
	81	Note	
	82	Thorns	
	83	PistonIn	
	84	PistonOut	
	85	Portal	
	86	Water	
	87	LavaPop	

Field Name	Field Type		Notes
	88	Lava	
	89	Burp	
	90	BucketFillWater	
	91	BucketFillLava	
	92	BucketEmptyWater	
	93	BucketEmptyLava	
	94	ArmorEquipChain	
	95	ArmorEquipDiamond	
	96	ArmorEquipGeneric	
	97	ArmorEquipGold	
	98	ArmorEquipIron	
	99	ArmorEquipLeather	
	100	ArmorEquipElytra	
	101	Record13	
	102	RecordCat	
	103	RecordBlocks	
	104	RecordChirp	
	105	RecordFar	
	106	RecordMall	
	107	RecordMellohi	
	108	RecordStal	
	109	RecordStrad	
	110	RecordWard	
	111	Record11	
	112	RecordWait	
	113	StopRecord	
	114	Flop	
	115	GuardianCurse	
	116	MobWarning	

Field Name	Field Type		Notes
	117	MobWarningBaby	
	118	Teleport	
	119	ShulkerOpen	
	120	ShulkerClose	
	121	Haggle	
	122	HaggleYes	
	123	HaggleNo	
	124	HaggleIdle	
	125	ChorusGrow	
	126	ChorusDeath	
	127	Glass	
	128	PotionBrewed	
	129	CastSpell	
	130	PrepareAttackSpell	
	131	PrepareSummon	
	132	PrepareWololo	
	133	Fang	
	134	Charge	
	135	CameraTakePicture	
	136	LeashKnotPlace	
	137	LeashKnotBreak	
	138	AmbientGrowl	
	139	AmbientWhine	
	140	AmbientPant	
	141	AmbientPurr	
	142	AmbientPurreow	
	143	DeathMinVolume	
	144	DeathMidVolume	
	145	ImitateBlaze	

Field Name	Field Type		Notes
	146	ImitateCaveSpider	
	147	ImitateCreeper	
	148	ImitateElderGuardian	
	149	ImitateEnderDragon	
	150	ImitateEnderman	
	151	ImitateEndermite	
	152	ImitateEvocationIllager	
	153	ImitateGhast	
	154	ImitateHusk	
	155	ImitateIllusionIllager	
	156	ImitateMagmaCube	
	157	ImitatePolarBear	
	158	ImitateShulker	
	159	ImitateSilverfish	
	160	ImitateSkeleton	
	161	ImitateSlime	
	162	ImitateSpider	
	163	ImitateStray	
	164	ImitateVex	
	165	ImitateVindicationIllager	
	166	ImitateWitch	
	167	ImitateWither	
	168	ImitateWitherSkeleton	
	169	ImitateWolf	
	170	ImitateZombie	
	171	ImitateZombiePigman	
	172	ImitateZombieVillager	
	173	EnderEyePlaced	
	174	EndPortalCreated	

Field Name	Field Type		Notes
	175	AnvilUse	
	176	BottleDragonBreath	
	177	PortalTravel	
	178	TridentHit	
	179	TridentReturn	
	180	TridentRiptide1	
	181	TridentRiptide2	
	182	TridentRiptide3	
	183	TridentThrow	
	184	TridentThunder	
	185	TridentHitGround	
	186	Default	
	187	FletchingTableUse	
	188	ElemConstructOpen	
	189	IceBombHit	
	190	BalloonPop	
	191	LtReactionIceBomb	
	192	LtReactionBleach	
	193	LtReactionElephantToothpaste	
	194	LtReactionElephantToothpaste2	
	195	LtReactionGlowStick	
	196	LtReactionGlowStick2	
	197	LtReactionLuminol	
	198	LtReactionSalt	
	199	LtReactionFertilizer	
	200	LtReactionFireball	
	201	LtReactionMagnesiumSalt	
	202	LtReactionMiscFire	
	203	LtReactionFire	

Field Name	Field Type		Notes
	204	LtReactionMiscExplosion	
	205	LtReactionMiscMystical	
	206	LtReactionMiscMystical2	
	207	LtReactionProduct	
	208	SparklerUse	
	209	GlowStickUse	
	210	SparklerActive	
	211	ConvertToDrowned	
	212	BucketFillFish	
	213	BucketEmptyFish	
	214	BubbleColumnUpwards	
	215	BubbleColumnDownwards	
	216	BubblePop	
	217	BubbleUpInside	
	218	BubbleDownInside	
	219	HurtBaby	
	220	DeathBaby	
	221	StepBaby	
	222	SpawnBaby	
	223	Born	
	224	TurtleEggBreak	
	225	TurtleEggCrack	
	226	TurtleEggHatched	
	227	LayEgg	
	228	TurtleEggAttacked	
	229	BeaconActivate	
	230	BeaconAmbient	
	231	BeaconDeactivate	
	232	BeaconPower	

Field Name	Field Type		Notes
	233	ConduitActivate	
	234	ConduitAmbient	
	235	ConduitAttack	
	236	ConduitDeactivate	
	237	ConduitShort	
	238	Swoop	
	239	BambooSaplingPlace	
	240	PreSneeze	
	241	Sneeze	
	242	AmbientTame	
	243	Scared	
	244	ScaffoldingClimb	
	245	CrossbowLoadingStart	
	246	CrossbowLoadingMiddle	
	247	CrossbowLoadingEnd	
	248	CrossbowShoot	
	249	CrossbowQuickChargeStart	
	250	CrossbowQuickChargeMiddle	
	251	CrossbowQuickChargeEnd	
	252	AmbientAggressive	
	253	AmbientWorried	
	254	CantBreed	
	255	ShieldBlock	
	256	LecternBookPlace	
	257	GrindstoneUse	
	258	Bell	
	259	CampfireCrackle	
	260	Roar	
	261	Stun	

Field Name	Field Type		Notes
	262	SweetBerryBushHurt	
	263	SweetBerryBushPick	
	264	CartographyTableUse	
	265	StonecutterUse	
	266	ComposterEmpty	
	267	ComposterFill	
	268	ComposterFillLayer	
	269	ComposterReady	
	270	BarrelOpen	
	271	BarrelClose	
	272	RaidHorn	
	273	LoomUse	
	274	AmbientInRaid	
	275	UicartographyTableUse	
	276	UistonecutterUse	
	277	UiloomUse	
	278	SmokerUse	
	279	BlastFurnaceUse	
	280	SmithingTableUse	
	281	Screech	
	282	Sleep	
	283	FurnaceUse	
	284	MooshroomConvert	
	285	MilkSuspiciously	
	286	Celebrate	
	287	JumpPrevent	
	288	AmbientPollinate	
	289	BeehiveDrip	
	290	BeehiveEnter	

Field Name	Field Type		Notes
	291	BeehiveExit	
	292	BeehiveWork	
	293	BeehiveShear	
	294	HoneybottleDrink	
	295	AmbientCave	
	296	Retreat	
	297	ConvertToZombified	
	298	Admire	
	299	StepLava	
	300	Tempt	
	301	Panic	
	302	Angry	
	303	AmbientMoodWarpedForest	
	304	AmbientMoodSoulsandValley	
	305	AmbientMoodNetherWastes	
	306	AmbientMoodBasaltDeltas	
	307	AmbientMoodCrimsonForest	
	308	RespawnAnchorCharge	
	309	RespawnAnchorDeplete	
	310	RespawnAnchorSetSpawn	
	311	RespawnAnchorAmbient	
	312	SoulEscapeQuiet	
	313	SoulEscapeLoud	
	314	RecordPigstep	
	315	LinkCompassToLodestone	
	316	UseSmithingTable	
	317	EquipNetherite	
	318	AmbientLoopWarpedForest	
	319	AmbientLoopSoulsandValley	

Field Name	Field Type		Notes
	320	AmbientLoopNetherWastes	
	321	AmbientLoopBasaltDeltas	
	322	AmbientLoopCrimsonForest	
	323	AmbientAdditionWarpedForest	
	324	AmbientAdditionSoulsandValley	
	325	AmbientAdditionNetherWastes	
	326	AmbientAdditionBasaltDeltas	
	327	AmbientAdditionCrimsonForest	
	328	SculkSensorPowerOn	
	329	SculkSensorPowerOff	
	330	BucketFillPowderSnow	
	331	BucketEmptyPowderSnow	
	332	PointedDripstoneCauldronDripWater	
	333	PointedDripstoneCauldronDripLava	
	334	PointedDripstoneDripWater	
	335	PointedDripstoneDripLava	
	336	CaveVinesPickBerries	
	337	BigDripleafTiltDown	
	338	BigDripleafTiltUp	
	339	CopperWaxOn	
	340	CopperWaxOff	
	341	Scrape	
	342	PlayerHurtDrown	
	343	PlayerHurtOnFire	
	344	PlayerHurtFreeze	
	345	UseSpyglass	
	346	StopUsingSpyglass	
	347	AmethystBlockChime	
	348	AmbientScreamer	

Field Name	Field Type		Notes
	349	HurtScreamer	
	350	DeathScreamer	
	351	MilkScreamer	
	352	JumpToBlock	
	353	PreRam	
	354	PreRamScreamer	
	355	RamImpact	
	356	RamImpactScreamer	
	357	SquidInkSquirt	
	358	GlowSquidInkSquirt	
	359	ConvertToStray	
	360	CakeAddCandle	
	361	ExtinguishCandle	
	362	AmbientCandle	
	363	BlockClick	
	364	BlockClickFail	
	365	SculkCatalystBloom	
	366	SculkShriekerShriek	
	367	WardenNearbyClose	
	368	WardenNearbyCloser	
	369	WardenNearbyClosest	
	370	WardenSlightlyAngry	
	371	RecordOtherside	
	372	Tongue	
	373	CrackIronGolem	
	374	RepairIronGolem	
	375	Listening	
	376	Heartbeat	
	377	HornBreak	

Field Name	Field Type		Notes
	378	_	
	379	SculkSpread	
	380	SculkCharge	
	381	SculkSensorPlace	
	382	SculkShriekerPlace	
	383	GoatCall0	
	384	GoatCall1	
	385	GoatCall2	
	386	GoatCall3	
	387	GoatCall4	
	388	GoatCall5	
	389	GoatCall6	
	390	GoatCall7	
	391	GoatCall8	
	392	GoatCall9	
	393	GoatHarmony0	
	394	GoatHarmony1	
	395	GoatHarmony2	
	396	GoatHarmony3	
	397	GoatHarmony4	
	398	GoatHarmony5	
	399	GoatHarmony6	
	400	GoatHarmony7	
	401	GoatHarmony8	
	402	GoatHarmony9	
	403	GoatMelody0	
	404	GoatMelody1	
	405	GoatMelody2	
	406	GoatMelody3	

Field Name	Field Type		Notes
	407	GoatMelody4	
	408	GoatMelody5	
	409	GoatMelody6	
	410	GoatMelody7	
	411	GoatMelody8	
	412	GoatMelody9	
	413	GoatBass0	
	414	GoatBass1	
	415	GoatBass2	
	416	GoatBass3	
	417	GoatBass4	
	418	GoatBass5	
	419	GoatBass6	
	420	GoatBass7	
	421	GoatBass8	
	422	GoatBass9	
	423	_	
	424	_	
	425	_	
	426	ImitateWarden	
	427	ListeningAngry	
	428	ItemGiven	
	429	ItemTaken	
	430	Disappeared	
	431	Reappeared	
	432	DrinkMilk	
	433	FrogspawnHatched	
	434	LaySpawn	
	435	FrogspawnBreak	

Field Name	Field Type		Notes
	436	SonicBoom	
	437	SonicCharge	
	438	SoundeventItemThrown	
	439	Record5	
	440	ConvertToFrog	
	441	RecordPlaying	
	442	EnchantingTableUse	
	443	StepSand	
	444	DashReady	
	445	BundleDropContents	
	446	BundleInsert	
	447	BundleRemoveOne	
	448	PressurePlateClickOff	
	449	PressurePlateClickOn	
	450	ButtonClickOff	
	451	ButtonClickOn	
	452	DoorOpen	
	453	DoorClose	
	454	TrapdoorOpen	
	455	TrapdoorClose	
	456	FenceGateOpen	
	457	FenceGateClose	
	458	Insert	
	459	Pickup	
	460	InsertEnchanted	
	461	PickupEnchanted	
	462	Brush	
	463	BrushCompleted	
	464	ShatterDecoratedPot	

Field Name	Field Type		Notes
	465	BreakDecoratedPot	
	466	SnifferEggCrack	
	467	SnifferEggHatched	
	468	WaxedSignInteractFail	
	469	RecordRelic	
	470	Bump	
	471	PumpkinCarve	
	472	ConvertHuskToZombie	
	473	PigDeath	
	474	HoglinZombified	
	475	AmbientUnderwaterEnter	
	476	AmbientUnderwaterExit	
	477	BottleFill	
	478	BottleEmpty	
	479	CrafterCraft	
	480	CrafterFail	
	481	DecoratedPotInsert	
	482	DecoratedPotInsertFail	
	483	CrafterDisableSlot	
	484	TrialSpawnerOpenShutter	
	485	TrialSpawnerEjectItem	
	486	TrialSpawnerDetectPlayer	
	487	TrialSpawnerSpawnMob	
	488	TrialSpawnerCloseShutter	
	489	TrialSpawnerAmbient	
	490	CopperBulbTurnOn	
	491	CopperBulbTurnOff	
	492	AmbientInAir	
	493	BreezeWindChargeBurst	

Field Name	Field Type		Notes
	494	ImitateBreeze	
	495	ArmadilloBrush	
	496	ArmadilloScuteDrop	
	497	EquipWolf	
	498	UnequipWolf	
	499	Reflect	
	500	VaultOpenShutter	
	501	VaultCloseShutter	
	502	VaultEjectItem	
	503	VaultInsertItem	
	504	VaultInsertItemFail	
	505	VaultAmbient	
	506	VaultActivate	
	507	VaultDeactive	
	508	HurtReduced	
	509	WindChargeBurst	
	510	ImitateBogged	
	511	WolfArmourCrack	
	512	WolfArmourBreak	
	513	WolfArmourRepair	
	514	MaceSmashAir	
	515	MaceSmashGround	
	516	TrialSpawnerChargeActivate	
	517	TrialSpawnerAmbientOminous	
	518	OminiousItemSpawnerSpawnItem	
	519	OminousBottleEndUse	
	520	MaceHeavySmashGround	
	521	OminousItemSpawnerSpawnItemBegin	
	522	_	

Field Name	Field Type		Notes
	523	ApplyEffectBadOmen	
	524	ApplyEffectRaidOmen	
	525	ApplyEffectTrialOmen	
	526	OminousItemSpawnerAboutToSpawnItem	
	527	RecordCreator	
	528	RecordCreatorMusicBox	
	529	RecordPrecipice	
	530	VaultRejectRewardedPlayer	
	531	ImitateDrowned	
	532	ImitateCreaking	
	533	BundleInsertFailed	
	534	SpongeAbsorb	
	535	_	
	536	BlockCreakingHeartTrail	
	537	CreakingHeartSpawn	
	538	Activate	
	539	Deactivate	
	540	Freeze	
	541	Unfreeze	
	542	Open	
	543	OpenLong	
	544	Close	
	545	CloseLong	
	546	ImitatePhantom	
	547	ImitateZoglin	
	548	ImitateGuardian	
	549	ImitateRavager	
	550	ImitatePillager	
	551	PlaceInWater	

Field Name	Field Type	
	552	StateChange
	553	ImitateHappyGhast
	554	UniqueGeneric
	555	RecordTears
	556	TheEndLightFlash
	557	LeadLeash
	558	LeadUnleash
	559	LeadBreak
	560	Unsaddle
	561	EquipCopper
	562	RecordLavaChicken
	563	PlaceItem
	564	SingleItemSwap
	565	MultiItemSwap

Type **LegacyEntityType** Datatype

TODO: remove?

Field Name	Field Type	
LegacyEntityType	li32 enum	
	10	Chicken
	11	Cow
	12	Pig
	13	Sheep
	14	Wolf
	15	Villager
	16	Mooshroom
	17	Squid

Field Name	Field Type		Notes
	18	Rabbit	
	19	Bat	
	20	Iron Golem	
	21	Snow Golem	
	22	Ocelot	
	23	Horse	
	24	Donkey	
	25	Mule	
	26	Skeleton Horse	
	27	Zombie Horse	
	28	Polar Bear	
	29	Llama	
	30	Parrot	
	31	Dolphin	
	32	Zombie	
	33	Creeper	
	34	Skeleton	
	35	Spider	
	36	Zombie Pigman	
	37	Slime	
	38	Enderman	
	39	Silverfish	
	40	Cave Spider	
	41	Ghast	
	42	Magma Cube	
	43	Blaze	
	44	Zombie Villager	
	45	Witch	
	46	Stray	

Field Name	Field Type		Notes
	47	Husk	
	48	Wither Skeleton	
	49	Guardian	
	50	Elder Guardian	
	51	Npc	
	52	Wither	
	53	Ender Dragon	
	54	Shulker	
	55	Endermite	
	56	Agent	
	57	Vindicator	
	58	Phantom	
	61	Armor Stand	
	62	Tripod Camera	
	63	Player	
	64	Item	
	65	Tnt	
	66	Falling Block	
	67	Moving Block	
	68	Xp Bottle	
	69	Xp Orb	
	70	Eye Of Ender Signal	
	71	Ender Crystal	
	72	Fireworks Rocket	
	73	Thrown Trident	
	74	Turtle	
	75	Cat	
	76	Shulker Bullet	
	77	Fishing Hook	

Field Name	Field Type		Notes
	78	Chalkboard	
	79	Dragon Fireball	
	80	Arrow	
	81	Snowball	
	82	Egg	
	83	Painting	
	84	Minecart	
	85	Fireball	
	86	Splash Potion	
	87	Ender Pearl	
	88	Leash Knot	
	89	Wither Skull	
	90	Boat	
	91	Wither Skull Dangerous	
	93	Lightning Bolt	
	94	Small Fireball	
	95	Area Effect Cloud	
	96	Hopper Minecart	
	97	Tnt Minecart	
	98	Chest Minecart	
	100	Command Block Minecart	
	101	Lingering Potion	
	102	Llama Spit	
	103	Evocation Fang	
	104	Evocation Illager	
	105	Vex	
	106	Ice Bomb	
	107	Balloon	
	108	Pufferfish	

Field Name	Field Type	Notes
	109	Salmon
	110	Drowned
	111	Tropicalfish
	112	Cod
	113	Panda

Type **DeviceOS**

Datatype

Field Name	Field Type		Notes
DeviceOS	li32 enum		
	0	Undefined	
	1	Android	
	2	IOS	
	3	OSX	
	4	FireOS	
	5	GearVR	Deprecated
	6	Hololens	
	7	Win10	
	8	Win32	
	9	Dedicated	
	10	TVOS	
	11	Orbis	
	12	NintendoSwitch	
	13	Xbox	
	14	WindowsPhone	
	15	Linux	

Type **AbilityLayers** Datatype

AbilityLayer represents the abilities of a specific layer, such as the base layer or the spectator layer.

Field Name	Field Type		Notes
Type	lu16 enum		
	0	Cache	Type represents the type of the layer. This is one of the AbilityLayerType constants defined above.
	1	Base	
	2	Spectator	
	3	Commands	
	4	Editor	
	5	Loading Screen	
Allowed	AbilitySet		The abilities that can be toggled between
Enabled	AbilitySet		The abilities that are currently active
Fly Speed	lf32		FlySpeed is the default horizontal fly speed of the layer.
Vertical Fly Speed	lf32		VerticalFlySpeed is the default vertical fly speed of the layer.
Walk Speed	lf32		WalkSpeed is the default walk speed of the layer.

Type **CameraPresets** Datatype

Field Name	Field Type	Notes
Name	string	Name is the name of the preset. Each preset must have their own unique name.
Parent	string	Parent is the name of the preset that this preset extends upon. This can be left empty.

Field Name	Field Type	Notes
Position	Vec3fopts	
Rotation	Vec2fopts	
Rotation Speed **Optional**	lf32	RotationSpeed is the speed at which the camera should rotate.
Snap To Target **Optional**	bool	SnapToTarget determines whether the camera should snap to the target entity or not.
Horizontal Rotation Limit **Optional**	vec2f	horizontal*rotation*limit is the horizontal rotation limit of the camera.
Vertical Rotation Limit **Optional**	vec2f	vertical*rotation*limit is the vertical rotation limit of the camera.
Continue Targeting **Optional**	bool	continue_targeting determines whether the camera should continue targeting the entity or not.
Tracking Radius **Optional**	lf32	TrackingRadius is the radius around the camera that the aim assist should track targets.
Offset **Optional**	vec2f	ViewOffset is only used in a follow_orbit camera and controls an offset based on a pivot point to the player, causing it to be shifted in a certain direction.
Entity Offset **Optional**	vec3f	EntityOffset controls the offset from the entity that the camera should be rendered at.
Radius **Optional**	lf32	Radius is only used in a follow_orbit camera and controls how far away from the player the camera should be rendered.
Yaw Limit Min **Optional**	lf32	

Field Name	Field Type	Notes
Yaw Limit Max **Optional**	lf32	
Audio Listener **Optional**	u8	
Player Effects **Optional**	bool	
Aim Assist Optional	**Preset Id** **Optional**	string	Preset is the ID of the preset that has previously been defined in the CameraAimAssistPresets packet.
	Target Mode **Optional**	li32 enum 0 — Angle 1 — Distance
	Angle **Optional**	vec2f	Angle is the maximum angle around the playes's cursor that the aim assist should check for a target, if target_mode is set to angle.
	Distance **Optional**	lf32	Distance is the maximum distance from the player's cursor should check for a target, if TargetMode is set to target_mode distance.
Control Scheme **Optional**	u8 enum 0 — Locked Player Relative Strafe 1 — Camera Relative 2 — Camera Relative Strafe 3 — Player Relative 4 — Player Relative Strafe	

Type **DisconnectFailReason** Datatype

Field Name	Field Type	Notes
DisconnectFailReason	zigzag32 enum	

0	Unknown
1	Cant Connect No Internet
2	No Permissions
3	Unrecoverable Error
4	Third Party Blocked
5	Third Party No Internet
6	Third Party Bad Ip
7	Third Party No Server Or Server Locked
8	Version Mismatch
9	Skin Issue
10	Invite Session Not Found
11	Edu Level Settings Missing
12	Local Server Not Found
13	Legacy Disconnect
14	User Leave Game Attempted
15	Platform Locked Skins Error
16	Realms World Unassigned
17	Realms Server Cant Connect
18	Realms Server Hidden
19	Realms Server Disabled Beta
20	Realms Server Disabled
21	Cross Platform Disallowed
22	Cant Connect
23	Session Not Found
24	Client Settings Incompatible With Server
25	Server Full

Field Name	Field Type		Notes
	26	Invalid Platform Skin	
	27	Edition Version Mismatch	
	28	Edition Mismatch	
	29	Level Newer Than Exe Version	
	30	No Fail Occurred	
	31	Banned Skin	
	32	Timeout	
	33	Server Not Found	
	34	Outdated Server	
	35	Outdated Client	
	36	No Premium Platform	
	37	Multiplayer Disabled	
	38	No Wifi	
	39	World Corruption	
	40	No Reason	
	41	Disconnected	
	42	Invalid Player	
	43	Logged In Other Location	
	44	Server Id Conflict	
	45	Not Allowed	
	46	Not Authenticated	
	47	Invalid Tenant	
	48	Unknown Packet	
	49	Unexpected Packet	
	50	Invalid Command Request Packet	
	51	Host Suspended	
	52	Login Packet No Request	
	53	Login Packet No Cert	
	54	Missing Client	

Field Name	Field Type		Notes
	55	Kicked	
	56	Kicked For Exploit	
	57	Kicked For Idle	
	58	Resource Pack Problem	
	59	Incompatible Pack	
	60	Out Of Storage	
	61	Invalid Level	
	62	Disconnect Packet Deprecated	
	63	Block Mismatch	
	64	Invalid Heights	
	65	Invalid Widths	
	66	Connection Lost	deprecated
	67	Zombie Connection	
	68	Shutdown	
	69	Reason Not Set	
	70	Loading State Timeout	
	71	Resource Pack Loading Failed	
	72	Searching For Session Loading Screen Failed	
	73	Conn Protocol Version	
	74	Subsystem Status Error	
	75	Empty Auth From Discovery	
	76	Empty Url From Discovery	
	77	Expired Auth From Discovery	
	78	Unknown Signal Service Sign In Failure	
	79	Xbl Join Lobby Failure	
	80	Unspecified Client Instance Disconnection	
	81	Conn Session Not Found	

Field Name	Field Type		Notes
	82	Conn Create Peer Connection	
	83	Conn Ice	
	84	Conn Connect Request	
	85	Conn Connect Response	
	86	Conn Negotiation Timeout	
	87	Conn Inactivity Timeout	
	88	Stale Connection Being Replaced	
	89	Realms Session Not Found	
	90	Bad Packet	
	91	Conn Failed To Create Offer	
	92	Conn Failed To Create Answer	
	93	Conn Failed To Set Local Description	
	94	Conn Failed To Set Remote Description	
	95	Conn Negotiation Timeout Waiting For Response	
	96	Conn Negotiation Timeout Waiting For Accept	
	97	Conn Incoming Connection Ignored	
	98	Conn Signaling Parsing Failure	
	99	Conn Signaling Unknown Error	
	100	Conn Signaling Unicast Delivery Failed	
	101	Conn Signaling Broadcast Delivery Failed	
	102	Conn Signaling Generic Delivery Failed	
	103	Editor Mismatch Editor World	
	104	Editor Mismatch Vanilla World	
	105	World Transfer Not Primary Client	
	106	Server Shutdown	

Field Name	Field Type		Notes
	107	Game Setup Cancelled	
	108	Game Setup Failed	
	109	No Venue	
	110	Conn Signalling Sign In Failed	
	111	Session Access Denied	
	112	Service Sign In Issue	
	113	Conn No Signaling Channel	
	114	Conn Not Logged In	
	115	Conn Client Signalling Error	
	116	Sub Client Login Disabled	
	117	Deep Link Trying To Open Demo World While Signed In	
	118	Async Join Task Denied	
	119	Realms Timeline Required	
	120	Guest Withough Host	
	121	Failed To Join Experience	

Type **FullContainerName**

Datatype

Field Name	Field Type	Notes
Container Id	ContainerSlotType	
Dynamic Container Id **Optional**	u32	

Type **MovementEffectType**

Datatype

Field Name	Field Type	Notes
MovementEffectType	varint enum	

Field Name	Field Type	Notes
	-1	Invalid
	0	GLIDE BOOST

Type **BiomeDefinition** Datatype

BiomeDefinition represents a biome definition in the game. This can be a vanilla biome or a completely custom biome.

Field Name	Field Type	Notes
Name Index	li16	NameIndex represents the index of the biome name in the string list from BiomeDefinitionListPacket.
Biome Id	lu16	BiomeID is the biome ID. This is optional and can be empty.
Temperature	lf32	Temperature is the temperature of the biome, used for weather, biome behaviours and sky colour.
Downfall	lf32	Downfall is the amount that precipitation affects colours and block changes.
Snow Foliage	lf32	Changes leaves turning white in snow
Depth	lf32	Depth …
Scale	lf32	Scale …
Map Water Colour	li32	MapWaterColour is an ARGB value for the water colour on maps in the biome.
Rain	bool	Rain is true if the biome has rain, false if it is a dry biome.
Tags length	varint	Tags are a list of indices of tags in the string list. These are used to group biomes together for biome generation and other purposes.
Tags array	lu16	
Chunk Generation	BiomeChunkGeneration	ChunkGeneration is optional information to assist in client-side

Field Name	Field Type	Notes
Optional		chunk generation.

Type **BiomeChunkGeneration** Datatype

BiomeChunkGeneration represents the information required for the client to generate chunks itself to create the illusion of a larger render distance.

Field Name	Field Type	Notes
Climate **Optional**	BiomeClimate	Climate is optional information to specify the biome's climate.
Consolidated Features length	varint	ConsolidatedFeatures is a list of features that are consolidated into a single feature.
Consolidated Features array	BiomeConsolidatedFeature	
Mountain Parameters **Optional**	BiomeMountainParameters	MountainParameters is optional information to specify the biome's mountain parameters.
Surface Material Adjustments length	varint	SurfaceMaterialAdjustments is a list of surface material adjustments.
Surface Material Adjustments array	BiomeElementData	
Surface Materials **Optional**	BiomeSurfaceMaterial	SurfaceMaterials is a set of materials to use for the surface layers of the biome.
Has Default Overworld Surface **Optional**	bool	HasDefaultOverworldSurface is true if the biome has a default overworld surface.

Field Name	Field Type	Notes
Has Swamp Surface	bool	HasSwampSurface is true if the biome has a swamp surface.
Has Frozen Ocean Surface	bool	HasFrozenOceanSurface is true if the biome has a frozen ocean surface.
Has End Surface	bool	HasEndSurface is true if the biome has an end surface.
Mesa Surface **Optional**	BiomeMesaSurface	MesaSurface is optional information to specify the biome's mesa surface.
Capped Surface **Optional**	BiomeCappedSurface	CappedSurface is optional information to specify the biome's capped surface, i.e. in the Nether.
Overworld Rules **Optional**	BiomeOverworldRules	OverworldRules is optional information to specify the biome's overworld rules, such as rivers and hills.
Multi Noise Rules **Optional**	BiomeMultiNoiseRules	MultiNoiseRules is optional information to specify the biome's multi-noise rules.
Legacy Rules length	varint	LegacyRules is a list of legacy rules for the biomes using an older format, which is just a list of weighted biomes.
Legacy Rules **array**	BiomeConditionalTransformation	

Type **BiomeClimate** Datatype

BiomeClimate represents the climate of a biome, mainly for ambience but also defines certain behaviours.

Field Name	Field Type	Notes
Temperature	lf32	Temperature is the temperature of the biome, used for weather, biome behaviours and sky colour.
Downfall	lf32	Downfall is the amount that precipitation affects colours and block changes.
Snow Accumulation Min	lf32	SnowAccumulationMin is the minimum amount of snow that can accumulate in the biome, every 0.125 is another layer of snow.
Snow Accumulation Max	lf32	SnowAccumulationMax is the maximum amount of snow that can accumulate in the biome, every 0.125 is another layer of snow.

Type **BiomeMountainParameters** Datatype

BiomeMountainParameters specifies the parameters for a mountain biome.

Field Name	Field Type	Notes
Steep Block	li32	SteepBlock is the runtime ID of the block to use for steep slopes.
North Slopes	bool	NorthSlopes is true if the biome has north slopes.
South Slopes	bool	SouthSlopes is true if the biome has south slopes.
West Slopes	bool	WestSlopes is true if the biome has west slopes.
East Slopes	bool	EastSlopes is true if the biome has east slopes.
Top Slide Enabled	bool	TopSlideEnabled is true if the biome has top slide enabled.

Type **BiomeSurfaceMaterial** Datatype

BiomeSurfaceMaterial specifies the materials to use for the surface layers of the biome.

Field Name	Field Type	Notes
Top Block	li32	TopBlock is the runtime ID of the block to use for the top layer.
Mid Block	li32	MidBlock is the runtime ID to use for the middle layers.
Sea Floor Block	li32	SeaFloorBlock is the runtime ID to use for the sea floor.
Foundation Block	li32	FoundationBlock is the runtime ID to use for the foundation layers.
Sea Block	li32	SeaBlock is the runtime ID to use for the sea layers.
Sea Floor Depth	li32	SeaFloorDepth is the depth of the sea floor, in blocks.

Type **BiomeMesaSurface** Datatype

BiomeMesaSurface specifies the materials to use for the mesa biome.

Field Name	Field Type	Notes
Clay Material	lu32	ClayMaterial is the runtime ID of the block to use for clay layers.
Hard Clay Material	lu32	HardClayMaterial is the runtime ID of the block to use for hard clay layers.
Bryce Pillars	bool	BrycePillars is true if the biome has bryce pillars, which are tall spire-like structures.
Has Forest	bool	HasForest is true if the biome has a forest.

Type **BiomeCappedSurface** Datatype

BiomeCappedSurface specifies the materials to use for the capped surface of a biome, such as in the Nether.

Field Name	Field Type	Notes
Floor Blocks length	varint	FloorBlocks is a list of runtime IDs to use for the floor blocks.
Floor Blocks array	li32	
Ceiling Blocks length	varint	CeilingBlocks is a list of runtime IDs to use for the ceiling blocks.
Ceiling Blocks array	li32	
Sea Block Optional	lu32	SeaBlock is an optional runtime ID to use for the sea block.
Foundation Block Optional	lu32	FoundationBlock is an optional runtime ID to use for the foundation block.
Beach Block Optional	lu32	BeachBlock is an optional runtime ID to use for the beach block.

Type **BiomeMultiNoiseRules** Datatype

BiomeMultiNoiseRules specifies the rules for multi-noise biomes, which are biomes that are defined by multiple noise parameters instead of just temperature and humidity.

Field Name	Field Type	Notes
Temperature	lf32	Temperature is the temperature level of the biome.
Humidity	lf32	Humidity is the humidity level of the biome.
Altitude	lf32	Altitude is the altitude level of the biome.
Weirdness	lf32	Weirdness is the weirdness level of the biome.
Weight	lf32	Weight is the weight of the biome, with a higher weight being more likely to be selected.

Type **BiomeWeight** Datatype

BiomeWeight defines the weight for a biome, used for weighted randomness.

Field Name	Field Type	Notes
Biome	li16	Biome is the index of the biome name in the string list.
Weight	lu32	Weight is the weight of the biome, with a higher weight being more likely to be selected.

Type **BiomeTemperatureWeight** Datatype

BiomeTemperatureWeight defines the weight for a temperature, used for weighted randomness.

Field Name	Field Type	Notes
Temperature	zigzag32	Temperature is the temperature that can be selected.
Weight	lu32	Weight is the weight of the temperature, with a higher weight being more likely to be selected.

Type **BiomeConsolidatedFeature** Datatype

BiomeConsolidatedFeature represents a feature that is consolidated into a single feature for the biome.

Field Name	Field Type	Notes
Scatter	BiomeScatterParameter	Scatter defines how the feature is scattered in the biome.
Feature	li16	Feature is the index of the feature's name in the string list.
Identifier	li16	Identifier is the index of the feature's identifier in the string list.

Field Name	Field Type	Notes
Pass	li16	Pass is the index of the feature's pass in the string list.
Can Use Internal	bool	CanUseInternal is true if the feature can use internal features.

Type **BiomeScatterParameter** Datatype

Field Name	Field Type	Notes
Coordinates length	varint	Coordinates is a list of coordinate rules to scatter the feature within.
Coordinates array	BiomeCoordinate	
Evaluation Order	zigzag32 enum	
	0 Xyz	EvaluationOrder is the order in which the coordinates are evaluated.
	1 Xzy	
	2 Yxz	
	3 Yzx	
	4 Zxy	
	5 Zyx	
Chance Percent Type	zigzag32	ChancePercentType is the type of expression operation to use for the chance percent.
Chance Percent	li16	ChancePercent is the index of the chance expression in the string list.
Chance Numerator	li32	ChanceNumerator is the numerator of the chance expression.
Chance Denominator	li32	ChanceDenominator is the denominator of the chance expression.
Iterations Type	zigzag32	IterationsType is the type of expression operation to use for the iterations.

Field Name	Field Type	Notes
Iterations	li16	Iterations is the index of the iterations expression in the string list.

Type **BiomeCoordinate** Datatype

BiomeCoordinate specifies coordinate rules for where features can be scattered in the biome.

Field Name	Field Type	Notes
Min Value Type	zigzag32	MinValueType is the type of expression operation to use for the minimum value.
Min Value	li16	MinValue is the index of the minimum value expression in the string list.
Max Value Type	zigzag32	MaxValueType is the type of expression operation to use for the maximum value.
Max Value	li16	MaxValue is the index of the maximum value expression in the string list.
Grid Offset	lu32	GridOffset is the offset of the grid, used for fixed grid and jittered grid distributions.
Grid Step Size	lu32	GridStepSize is the step size of the grid, used for fixed grid and jittered grid distributions.
Distribution	zigzag32 enum	
	0 Single Valued	Distribution is the type of distribution to use for the coordinate.
	1 Uniform	
	2 Gaussian	
	3 Inverse Gaussian	
	4 Fixed Grid	
	5 Jittered Grid	
	6 Triangle	

Type **BiomeElementData** Datatype

BiomeElementData are set rules to adjust the surface materials of the biome.

Field Name	Field Type	Notes
Noise Frequency Scale	lf32	NoiseFrequencyScale is the frequency scale of the noise used to adjust the surface materials.
Noise Lower Bound	lf32	NoiseLowerBound is the minimum noise value required to be selected.
Noise Upper Bound	lf32	NoiseUpperBound is the maximum noise value required to be selected.
Height Min Type	zigzag32	HeightMinType is the type of expression operation to use for the minimum height.
Height Min	li16	HeightMin is the index of the minimum height expression in the string list.
Height Max Type	zigzag32	HeightMaxType is the type of expression operation to use for the maximum height.
Height Max	li16	HeightMax is the index of the maximum height expression in the string list.
Adjusted Materials	BiomeSurfaceMaterial	AdjustedMaterials is the materials to use for the surface layers of the biome if selected.

Type **BiomeOverworldRules** Datatype

BiomeOverworldRules specifies a list of transformation rules to apply to different parts of the overworld.

Field Name	Field Type	Notes
Hills Transformations length	varint	HillsTransformations is a list of weighted biome transformations to apply to hills.

Field Name	Field Type	Notes
Hills Transformations array	BiomeWeight	
Mutate Transformations length	varint	MutateTransformations is a list of weighted biome transformations to apply to mutated biomes.
Mutate Transformations array	BiomeWeight	
River Transformations length	varint	RiverTransformations is a list of weighted biome transformations to apply to rivers.
River Transformations array	BiomeWeight	
Shore Transformations length	varint	ShoreTransformations is a list of weighted biome transformations to apply to shores.
Shore Transformations array	BiomeWeight	
Pre Hills Edge Transformations length	varint	PreHillsEdgeTransformations is a list of conditional transformations to apply to the edges of hills.
Pre Hills Edge Transformations array	BiomeConditionalTransformation	
Post Shore Edge Transformations length	varint	PostShoreEdgeTransformations is a list of conditional transformations to apply to the edges of shores.
Post Shore Edge Transformations array	BiomeConditionalTransformation	
Climate Transformations length	varint	ClimateTransformations is a list of weighted temperature transformations to apply to the biome's climate.
Climate Transformations	BiomeTemperatureWeight	

Field Name	Field Type	Notes
array		

Type **BiomeConditionalTransformation** Datatype

BiomeConditionalTransformation is the legacy method of transforming biomes.

Field Name	Field Type	Notes
Weighted Biomes length	varint	WeightedBiomes is a list of biomes and their weights.
Weighted Biomes array	BiomeWeight	
Condition Json	li16	ConditionJSON is an index of the condition JSON data in the string list.
Min Passing Neighbours	lu32	MinPassingNeighbours is the minimum number of neighbours that must pass the condition for the transformation to be applied.

Type **EaseType** Datatype

Field Name	Field Type	Notes
EaseType	u8 enum	
	0	Linear
	1	Spring
	2	InQuad
	3	OutQuad
	4	InOutQuad
	5	InCubic
	6	OutCubic
	7	InOutCubic
	8	InQuart

Field Name	Field Type		Notes
	9	OutQuart	
	10	InOutQuart	
	11	InQuint	
	12	OutQuint	
	13	InOutQuint	
	14	InSine	
	15	OutSine	
	16	InOutSine	
	17	InExpo	
	18	OutExpo	
	19	InOutExpo	
	20	InCirc	
	21	OutCirc	
	22	InOutCirc	
	23	InBounce	
	24	OutBounce	
	25	InOutBounce	
	26	InBack	
	27	OutBack	
	28	InOutBack	
	29	InElastic	
	30	OutElastic	
	31	InOutElastic	

Login Sequence

The login process is as follows:

- C→S: Login
- S→C: Server To Client Handshake
- C→S: Client To Server Handshake
- S→C: Play Status (Login success)
- To spawn, the following packets should be sent, in order, after the ones above:

- S→C: Resource Packs Info
- C→S: Resource Pack Client Response
- S→C: Resource Pack Stack
- C→S: Resource Pack Client Response
- S→C: Start Game
- S→C: Item Registry
- S→C: Creative Content
- S→C: Biome Definition List
- S→C: Chunks
- S→C: Play Status (Player spawn)

  If there are no resource packs being sent, a Resource Pack Stack can be sent directly after Resource Packs Info to avoid the client responses.

Serverbound

1 Packet Login

Field Name	Field Type	Notes
Protocol Version	i32	Protocol version (Big Endian!)
Tokens	["encapsulated", { "lengthType": "varint", "type": "LoginTokens" }]	The structure of the login tokens has changed in 1.21.90. The encapsulated data is now a JSON object with a stringified Certificate.

Datatype

Type LoginTokens

Field Name	Field Type	Notes
Identity	LittleString	JSON array of JWT data: contains the display name, UUID and XUID It should be signed by the Mojang public key For 1.21.90+, the 'identity' field is a Little-Endian length-prefixed JSON-encoded string. This JSON object must contain a 'Certificate' key, whose value is a *stringified* JSON object that holds the actual JWT 'chain' array.
Client	LittleString	JWT containing skin and other client data.

Clientbound

2 Packet Play Status

Field Name	Field Type		Notes
Status	i32 enum		
	0	Login Success	Sent after Login has been successfully decoded and the player has logged in
	1	Failed Client	Displays "Could not connect: Outdated client!"
	2	Failed Spawn	Displays "Could not connect: Outdated server!"
	3	Player Spawn	Sent after world data to spawn the player
	4	Failed Invalid Tenant	Displays "Unable to connect to world. Your school does not have access to this server."
	5	Failed Vanilla Edu	Displays "The server is not running Minecraft: Education Edition. Failed to connect."
	6	Failed Edu Vanilla	Displays "The server is running an incompatible edition of Minecraft. Failed to connect."
	7	Failed Server Full	Displays "Wow this server is popular! Check back later to see if space opens up. Server Full"
	8	Failed Editor Vanilla Mismatch	Cannot join a vanilla game on editor
	9	Failed Vanilla Editor Mismatch	Cannot join an editor game on vanilla

3

Clientbound

Packet Server To Client Handshake

Field Name	Field Type	Notes
Token	string	Contains the salt to complete the Diffie-Hellman key exchange

4 Serverbound

Packet Client To Server Handshake

Sent by the client in response to a Server To Client Handshake packet sent by the server. It is the first encrypted packet in the login handshake and serves as a confirmation that encryption is correctly initialized client side. It has no fields.

Field Name	Field Type	Notes

5 Clientbound

Packet Disconnect

Sent by the server to disconnect a client.

<table>
<tr><th>Field Name</th><th colspan="4">Field Type</th><th>Notes</th></tr>
<tr><td>Reason</td><td colspan="4">DisconnectFailReason</td><td>Reason is the reason for the disconnection. It seems as if this field has no use other than for telemetry reasons as it does not affect the message that gets displayed on the disconnect screen.</td></tr>
<tr><td>Hide Disconnect Reason</td><td colspan="4">bool</td><td>Specifies if the disconnection screen should be hidden when the client is disconnected, meaning it will be sent directly to the main menu.</td></tr>
<tr><td rowspan="3">🔁
if Hide Disconnect Reason</td><td>Is True</td><td colspan="3">void</td></tr>
<tr><td rowspan="2">Default</td><td>Message</td><td>string</td><td>An optional message to show when disconnected.</td></tr>
<tr><td>Filtered Message</td><td>string</td><td>FilteredMessage is a filtered version of Message with all the profanity removed. The client will use this over Message if this field is not empty and they have the "Filter Profanity" setting enabled.</td></tr>
</table>

6 Clientbound

Packet Resource Packs Info

Field Name	Field Type		Notes
Must Accept	bool		If the resource pack requires the client accept it.
Has Addons	bool		HasAddons specifies if any of the resource packs contain addons in them. If set to true, only clients that support addons will be able to download them.
Has Scripts	bool		If scripting is enabled.
Disable Vibrant Visuals	bool		ForceDisableVibrantVisuals specifies if the vibrant visuals feature should be forcibly disabled on the server. If set to true, the server will ensure that vibrant visuals are not enabled, regardless of the client's settings.
World Template	**Uuid**	uuid	WorldTemplateUUID is teh UUID of the template that has been used to generate the world. Templates can be downloaded from the marketplace or installed via '.mctemplate' files. If the world was not generated from a template, this field is empty.
	Version	string	WorldTemplateVersion is the version of the world template that has been used to generate the world. If the world was not generated from a template, this field is empty.
Texture Packs	TexturePackInfos		A list of resource packs that the client needs to download before joining the server. The order of these resource packs is not relevant in this packet. It is however important in the Resource Pack Stack packet.

7 ## Packet Resource Pack Stack

Clientbound

Field Name	Field Type	Notes
Must Accept	bool	If the resource pack must be accepted for the player to join the server.
Behavior Packs	ResourcePackIdVersions	[inline]

Field Name	Field Type	Notes
Resource Packs	ResourcePackIdVersions	[inline]
Game Version	string	
Experiments	Experiments	
Experiments Previously Used	bool	
Has Editor Packs	bool	

8 Serverbound

Packet Resource Pack Client Response

Field Name	Field Type		Notes
Response Status	u8 enum		
	0	None	
	1	Refused	
	2	Send Packs	
	3	Have All Packs	
	4	Completed	
Resourcepackids	ResourcePackIds		All of the pack IDs.

9 Bidirectional

Packet Text

Sent by the client to the server to send chat messages, and by the server to the client to forward or send messages, which may be chat, popups, tips etc.

Field Name	Field Type		Notes
Type	u8 enum		
	0	Raw	TextType is the type of the text sent. When a client sends this to the server, it should always be TextTypeChat. If the

<table>
<tr><th>Field Name</th><th colspan="2">Field Type</th><th>Notes</th></tr>
<tr><td></td><td colspan="2"></td><td>server sends it, it may be one of the other text types above.</td></tr>
<tr><td></td><td>1</td><td>Chat</td><td></td></tr>
<tr><td></td><td>2</td><td>Translation</td><td></td></tr>
<tr><td></td><td>3</td><td>Popup</td><td></td></tr>
<tr><td></td><td>4</td><td>Jukebox Popup</td><td></td></tr>
<tr><td></td><td>5</td><td>Tip</td><td></td></tr>
<tr><td></td><td>6</td><td>System</td><td></td></tr>
<tr><td></td><td>7</td><td>Whisper</td><td></td></tr>
<tr><td></td><td>8</td><td>Announcement</td><td></td></tr>
<tr><td></td><td>9</td><td>Json Whisper</td><td></td></tr>
<tr><td></td><td>10</td><td>Json</td><td></td></tr>
<tr><td></td><td>11</td><td>Json Announcement</td><td></td></tr>
<tr><td>Needs Translation</td><td colspan="2">bool</td><td>NeedsTranslation specifies if any of the messages need to be translated. It seems that where % is found in translatable text types, these are translated regardless of this bool. Translatable text types include TextTypeTip, TextTypePopup and TextTypeJukeboxPopup.</td></tr>
<tr><td>🔁
if Type</td><td colspan="2">Is Chat Or Whisper Or Announcement</td><td>Source Name string
Message string</td></tr>
<tr><td></td><td colspan="2">Is Raw Or Tip Or System Or Json Whisper Or Json Or Json Announcement</td><td>Message string</td></tr>
<tr><td></td><td colspan="2">Is Translation Or Popup Or Jukebox Popup</td><td>Message string
Parameters length varint
Parameters array string</td></tr>
<tr><td>Xuid</td><td colspan="2">string</td><td>The XUID of the player who sent this message.</td></tr>
</table>

Field Name	Field Type	Notes
Platform Chat Id	string	PlatformChatID is an identifier only set for particular platforms when chatting (presumably only for Nintendo Switch). It is otherwise an empty string, and is used to decide which players are able to chat with each other.
Filtered Message	string	FilteredMessage is a filtered version of Message with all the profanity removed. The client will use this over Message if this field is not empty and they have the "Filter Profanity" setting enabled.

10 Packet Set Time

Clientbound

For additional information and examples of all the chat types above, see here: https://imgur.com/a/KhcFscg Sent by the server to update the current time client-side. The client actually advances time client-side by itself, so this packet does not need to be sent each tick. It is merely a means of synchronizing time between server and client.

Field Name	Field Type	Notes
Time	zigzag32	Time is the current time. The time is not limited to 24000 (time of day), but continues progressing after that.

11 Packet Start Game

Clientbound

Sent by the server to send information about the world the player will be spawned in.

Field Name	Field Type	Notes
Entity Id	zigzag64	The unique ID of the player. The unique ID is a value that remains consistent across different sessions of the same world, but most unofficial servers simply fill the runtime ID of the entity out for this field.
Runtime Entity Id	varint64	The runtime ID of the player. The runtime ID is unique for each world session, and entities are generally identified in packets using this runtime ID.

Field Name	Field Type	Notes
Player Gamemode	GameMode	PlayerGameMode is the game mode the player currently has. It is a value from 0-4, with 0 being survival mode, 1 being creative mode, 2 being adventure mode, 3 being survival spectator and 4 being creative spectator. This field may be set to 5 to make the client fall back to the game mode set in the WorldGameMode field.
Player Position	vec3f	The spawn position of the player in the world. In servers this is often the same as the world's spawn position found below.
Rotation	vec2f	The pitch and yaw of the player
Seed	lu64	The seed used to generate the world.
Biome Type	li16	
Biome Name	string	
Dimension	zigzag32 enum	
	0 Overworld	Dimension is the ID of the dimension that the player spawns in. It is a value from 0-2, with 0 being the overworld, 1 being the nether and 2 being the end.
	1 Nether	
	2 End	
Generator	zigzag32	Generator is the generator used for the world. It is a value from 0-4, with 0 being old limited worlds, 1 being infinite worlds, 2 being flat worlds, 3 being nether worlds and 4 being end worlds. A value of 0 will actually make the client stop rendering chunks you send beyond the world limit. As of 1.21.80, protocol.PlayerMovementModeServer is the minimum requirement for MovementType.
World Gamemode	GameMode	The world game mode that a player gets when it first spawns in the world. It is shown in the settings and is used if the Player Gamemode is set to 5.

Field Name	Field Type	Notes
Hardcore	bool	Specifies if the game is locked to "hardcore" mode or not, meaning the world will be unplayable after player dies in survival game mode. Persists even after switching player or world game modes.
Difficulty	zigzag32	Difficulty is the difficulty of the world. It is a value from 0-3, with 0 being peaceful, 1 being easy, 2 being normal and 3 being hard.
Spawn Position	BlockCoordinates	The block on which the world spawn of the world. This coordinate has no effect on the place that the client spawns, but it does have an effect on the direction that a compass poInts.
Achievements Disabled	bool	Defines if achievements are disabled in the world. The client crashes if this value is set to true while the player's or the world's game mode is creative, and it's recommended to simply always set this to false as a server.
Editor World Type	zigzag32 enum	
	0 Not Editor	EditorWorldType is a value to dictate the type of editor mode, a special mode recently introduced adding "powerful tools for editing worlds, intended for experienced creators."
	1 Project	
	2 Test Level	
	3 Realms Upload	
Created In Editor	bool	Whether the world was created in editor mode
Exported From Editor	bool	Whether the world was exported from editor mode

Field Name	Field Type	Notes
Day Cycle Stop Time	zigzag32	The time at which the day cycle was locked if the day cycle is disabled using the respective game rule. The client will maIntain this time as Boolean as the day cycle is disabled.
Edu Offer	zigzag32	Some Minecraft: Education Edition field that specifies what 'region' the world was from, with 0 being None, 1 being RestOfWorld, and 2 being China. The actual use of this field is unknown.
Edu Features Enabled	bool	Specifies if the world has education edition features enabled, such as the blocks or entities specific to education edition.
Edu Product Uuid	string	
Rain Level	lf32	The level specifying the Intensity of the rain falling. When set to 0, no rain falls at all.
Lightning Level	lf32	
Has Confirmed Platform Locked Content	bool	The level specifying the Intensity of the thunder. This may actually be set independently from the rain level, meaning dark clouds can be produced without rain.
Is Multiplayer	bool	Specifies if the world is a multi-player game. This should always be set to true for servers.
Broadcast To Lan	bool	Specifies if LAN broadcast was Intended to be enabled for the world.
Xbox Live Broadcast Mode	varint	The mode used to broadcast the joined game across XBOX Live.
Platform Broadcast Mode	varint	The mode used to broadcast the joined game across the platform.
Enable Commands	bool	If commands are enabled for the player. It is recommended to always set this to true on the server, as

Field Name	Field Type	Notes
		setting it to false means the player cannot, under any circumstance, use a command.
Is Texturepacks Required	bool	Specifies if the texture pack the world might hold is required, meaning the client was forced to download it before joining.
Gamerules length	varint	Defines game rules currently active with their respective values. The value of these game rules may be either 'bool', 'Int32' or 'Float32'. Some game rules are server side only, and don't necessarily need to be sent to the client.
Gamerules array	GameRuleVarint	
Experiments	Experiments	
Experiments Previously Used	bool	
Bonus Chest	bool	Specifies if the world had the bonus map setting enabled when generating it. It does not have any effect client-side.
Map Enabled	bool	Specifies if the world has the start with map setting enabled, meaning each joining player obtains a map. This should always be set to false, because the client obtains a map all on its own accord if this is set to true.
Permission Level	PermissionLevel	The permission level of the player. It is a value from 0-3, with 0 being visitor, 1 being member, 2 being operator and 3 being custom.
Server Chunk Tick Range	li32	The radius around the player in which chunks are ticked. Most servers set this value to a fixed number, as it does not necessarily affect anything client-side.

Field Name	Field Type	Notes
Has Locked Behavior Pack	bool	Specifies if the texture pack of the world is locked, meaning it cannot be disabled from the world. This is typically set for worlds on the marketplace that have a dedicated texture pack.
Has Locked Resource Pack	bool	Specifies if the texture pack of the world is locked, meaning it cannot be disabled from the world. This is typically set for worlds on the marketplace that have a dedicated texture pack.
Is From Locked World Template	bool	Specifies if the world from the server was from a locked world template. For servers this should always be set to false.
Msa Gamertags Only	bool	
Is From World Template	bool	Specifies if the world from the server was from a locked world template. For servers this should always be set to false.
Is World Template Option Locked	bool	Specifies if the world was a template that locks all settings that change properties above in the settings GUI. It is recommended to set this to true for servers that do not allow things such as setting game rules through the GUI.
Only Spawn V1 Villagers	bool	A hack that Mojang put in place to preserve backwards compatibility with old villagers. The his never actually read though, so it has no functionality.
Persona Disabled	bool	PersonaDisabled is true if persona skins are disabled for the current game session.
Custom Skins Disabled	bool	CustomSkinsDisabled is true if custom skins are disabled for the current game session.
Emote Chat Muted	bool	EmoteChatMuted specifies if players will be sent a chat message when

Field Name	Field Type	Notes
		using certain emotes.
Game Version	string	The version of the game from which Vanilla features will be used. The exact function of this field isn't clear.
Limited World Width	li32	
Limited World Length	li32	
Is New Nether	bool	
Edu Resource Uri	EducationSharedResourceURI	
Experimental Gameplay Override	bool	
Chat Restriction Level	u8 enum	
	0 None	ChatRestrictionLevel specifies the level of restriction on in-game chat.
	1 Dropped	
	2 Disabled	
Disable Player Interactions	bool	DisablePlayerInteractions is true if the client should ignore other players when interacting with the world.
Server Identifier	string	
World Identifier	string	
Scenario Identifier	string	
Owner Identifier	string	
Level Id	string	A base64 encoded world ID that is used to identify the world.
World Name	string	The name of the world that the player is joining. Note that this field shows up above the player list for the rest of the game session, and cannot be

Field Name	Field Type	Notes
		changed. Setting the server name to this field is recommended.
Premium World Template Id	string	A UUID specific to the premium world template that might have been used to generate the world. Servers should always fill out an empty String for this.
Is Trial	bool	Specifies if the world was a trial world, meaning features are limited and there is a time limit on the world.
Rewind History Size	zigzag32	RewindHistorySize is the amount of history to keep at maximum
Server Authoritative Block Breaking	bool	ServerAuthoritativeBlockBreaking specifies if block breaking should be sent through packet.PlayerAuthInput or not. This field is somewhat redundant as it is always enabled if server authoritative movement is enabled.
Current Tick	li64	The total time in ticks that has elapsed since the start of the world.
Enchantment Seed	zigzag32	The seed used to seed the random used to produce enchantments in the enchantment table. Note that the exact correct random implementation must be used to produce the correct results both client- and server-side.
Block Properties	BlockProperties	BlockProperties is a list of all the custom blocks registered on the server.
Multiplayer Correlation Id	string	A unique ID specifying the multi-player session of the player. A random UUID should be filled out for this field.
Server Authoritative Inventory	bool	ServerAuthoritativeInventory specifies if the server authoritative inventory system is enabled. This is a new system introduced in 1.16. Backwards compatibility with the inventory transactions has to some extent been preserved, but will eventually be removed.

Field Name	Field Type	Notes
Engine	string	The server's engine version, used for telemetry
Property Data	nbt	PropertyData contains properties that should be applied on the player. These properties are the same as the ones that are sent in the SyncActorProperty packet.
Block Pallette Checksum	lu64	A checksum to ensure block types between the server and client match
World Template Id	uuid	WorldTemplateID is a UUID that identifies the template that was used to generate the world. Servers that do not use a world based off of a template can set this to an empty UUID.
Client Side Generation	bool	ClientSideGeneration is true if the client should use the features registered in the FeatureRegistry packet to generate terrain client-side to save on bandwidth.
Block Network Ids Are Hashes	bool	UseBlockNetworkIDHashes is true if the client should use the hash of a block's name as its network ID rather than its index in the expected block palette. This is useful for servers that wish to support multiple protocol versions and custom blocks, but it will result in extra bytes being written for every block in a sub chunk palette.
Tick Death Systems	bool	
Server Controlled Sound	bool	

12 **Packet Add Player** Clientbound

Field Name	Field Type	Notes
Uuid	uuid	UUID is the UUID of the player. It is the same UUID that the client sent in the Login packet at the start of the session. A player with this UUID must exist in the player list (built up using the Player List packet) for it to show up in-game.
Username	string	Username is the name of the player. This username is the username that will be set as the initial name tag of the player.
Runtime Id	varint64	The runtime ID of the player. The runtime ID is unique for each world session, and entities are generally identified in packets using this runtime ID.
Platform Chat Id	string	An identifier only set for particular platforms when chatting (presumably only for Nintendo Switch). It is otherwise an empty string, and is used to decide which players are able to chat with each other.
Position	vec3f	Position is the position to spawn the player on. If the player is on a distance that the viewer cannot see it, the player will still show up if the viewer moves closer.
Velocity	vec3f	Velocity is the initial velocity the player spawns with. This velocity will initiate client side movement of the player.
Pitch	lf32	Pitch is the vertical rotation of the player. Facing straight forward yields a pitch of 0. Pitch is measured in degrees.
Yaw	lf32	Yaw is the horizontal rotation of the player. Yaw is also measured in degrees.
Head Yaw	lf32	HeadYaw is the same as Yaw, except that it applies specifically to the head of the player. A different value for HeadYaw than Yaw means

Field Name	Field Type	Notes
		that the player will have its head turned.
Held Item	Item	HeldItem is the item that the player is holding. The item is shown to the viewer as soon as the player itself shows up. Needless to say that this field is rather pointless, as additional packets still must be sent for armour to show up.
Gamemode	GameMode	GameType is the game type of the player. If set to GameTypeSpectator, the player will not be shown to viewers.
Metadata	MetadataDictionary	EntityMetadata is a map of entity metadata, which includes flags and data properties that alter in particular the way the player looks. Flags include ones such as 'on fire' and 'sprinting'. The metadata values are indexed by their property key.
Properties	EntityProperties	EntityProperties holds lists of entity properties that define specific attributes of an entity. As of v1.19.40, the vanilla server does not use these properties, however they are still supported by the protocol.
Unique Id	li64	The unique ID of the player. The unique ID is a value that remains consistent across different sessions of the same world, but most unoffical servers simply fill the runtime ID of the player out for this field.
Permission Level	PermissionLevel	
Command Permission	CommandPermissionLevel	
Abilities length	u8	AbilityLayer represents the abilities of a specific layer, such as the base layer or the spectator layer.
Abilities array	AbilityLayers	

Field Name	Field Type	Notes
Links	Links	EntityLinks is a list of entity links that are currently active on the player. These links alter the way the player shows up when first spawned in terms of it shown as riding an entity. Setting these links is important for new viewers to see the player is riding another entity.
Device Id	string	DeviceID is the device ID set in one of the files found in the storage of the device of the player. It may be changed freely, so it should not be relied on for anything.
Device Os	DeviceOS	BuildPlatform is the build platform/device OS of the player that is about to be added, as it sent in the Login packet when joining.

13

Packet Add Entity

Clientbound

Field Name	Field Type	Notes
Unique Id	zigzag64	EntityUniqueID is the unique ID of the entity. The unique ID is a value that remains consistent across different sessions of the same world, but most servers simply fill the runtime ID of the entity out for
Runtime Id	varint64	EntityRuntimeID is the runtime ID of the entity. The runtime ID is unique for each world session, and entities are generally identified in packets using this runtime ID.
Entity Type	string	EntityType is the string entity type of the entity, for example 'minecraft:skeleton'. A list of these entities may be found online.
Position	vec3f	Position is the position to spawn the entity on. If the entity is on a distance that the player cannot see it, the entity will still show up if the player moves closer.
Velocity	vec3f	Velocity is the initial velocity the entity spawns with. This velocity will initiate client

Field Name	Field Type	Notes
		side movement of the entity.
Pitch	lf32	Pitch is the vertical rotation of the entity. Facing straight forward yields a pitch of 0. Pitch is measured in degrees.
Yaw	lf32	Yaw is the horizontal rotation of the entity. Yaw is also measured in degrees.
Head Yaw	lf32	HeadYaw is the same as Yaw, except that it applies specifically to the head of the entity. A different value for HeadYaw than Yaw means that the entity will have its head turned.
Body Yaw	lf32	BodyYaw is the same as Yaw, except that it applies specifically to the body of the entity. A different value for BodyYaw than HeadYaw means that the entity will have its body turned, although it is unclear what the difference between BodyYaw and Yaw is.
Attributes	EntityAttributes	Attributes is a slice of attributes that the entity has. It includes attributes such as its health, movement speed, etc.
Metadata	MetadataDictionary	EntityMetadata is a map of entity metadata, which includes flags and data properties that alter in particular the way the entity looks. Flags include ones such as 'on fire' and 'sprinting'. The metadata values are indexed by their property key.
Properties	EntityProperties	EntityProperties holds lists of entity properties that define specific attributes of an entity. As of v1.19.40, the vanilla server does not use these properties, however they are still supported by the protocol.
Links	Links	EntityLinks is a list of entity links that are currently active on the entity. These links alter the way the entity shows up when first spawned in terms of it shown as riding an entity. Setting these links is important for new viewers to see the entity is riding another entity.

14 ## Packet Remove Entity

Clientbound

Field Name	Field Type	Notes
Entity Id Self	zigzag64	

15 ## Packet Add Item Entity

Clientbound

Field Name	Field Type	Notes
Entity Id Self	zigzag64	
Runtime Entity Id	varint64	
Item	Item	
Position	vec3f	
Velocity	vec3f	
Metadata	MetadataDictionary	
Is From Fishing	bool	

17 ## Packet Take Item Entity

Clientbound

Field Name	Field Type	Notes
Runtime Entity Id	varint64	
Target	varint	

18 ## Packet Move Entity

Bidirectional

MoveActorAbsolute is sent by the server to move an entity to an absolute position. It is typically used for movements where high accuracy isn't needed, such as for long range teleporting.

Field Name	Field Type	Notes
Runtime Entity Id	varint64	EntityRuntimeID is the runtime ID of the entity. The runtime ID is unique for each world session, and entities are generally identified in packets using this runtime ID.
Flags	u8	Flags is a combination of flags that specify details of the movement. It is a combination of the flags above.
Position	vec3f	Position is the position to spawn the entity on. If the entity is on a distance that the player cannot see it, the entity will still show up if the player moves closer.
Rotation	Rotation	Rotation is a Vec3 holding the X, Y and Z rotation of the entity after the movement. This is a Vec3 for the reason that projectiles like arrows don't have yaw/pitch, but do have roll.

19

Packet Move Player

Bidirectional

MovePlayer is sent by players to send their movement to the server, and by the server to update the movement of player entities to other players.

Field Name	Field Type	Notes
Runtime Id	varint	EntityRuntimeID is the runtime ID of the player. The runtime ID is unique for each world session, and entities are generally identified in packets using this runtime ID.
Position	vec3f	Position is the position to spawn the player on. If the player is on a distance that the viewer cannot see it, the player will still show up if the viewer moves closer.
Pitch	lf32	Pitch is the vertical rotation of the player. Facing straight forward yields a pitch of 0. Pitch is measured in degrees.
Yaw	lf32	Yaw is the horizontal rotation of the player. Yaw is also measured in degrees
Head Yaw	lf32	HeadYaw is the same as Yaw, except that it applies specifically to the head of the player. A different value for HeadYaw than Yaw means that the player will have its head turned
Mode	u8 enum	

<table>
<tr><th>Field Name</th><th colspan="3">Field Type</th><th colspan="3">Notes</th></tr>
<tr><td></td><td>0</td><td>Normal</td><td colspan="3">Mode is the mode of the movement. It specifies the way the player's movement should be shown to other players. It is one of the constants below.</td></tr>
<tr><td></td><td>1</td><td>Reset</td><td colspan="3"></td></tr>
<tr><td></td><td>2</td><td>Teleport</td><td colspan="3"></td></tr>
<tr><td></td><td>3</td><td>Rotation</td><td colspan="3"></td></tr>
<tr><td>On Ground</td><td colspan="3">bool</td><td colspan="3">OnGround specifies if the player is considered on the ground. Note that proxies or hacked clients could fake this to always be true, so it should not be taken for granted.</td></tr>
<tr><td>Ridden Runtime Id</td><td colspan="3">varint</td><td colspan="3">RiddenEntityRuntimeID is the runtime ID of the entity that the player might currently be riding. If not riding, this should be left 0.</td></tr>
<tr><td rowspan="7">Teleport
if Mode</td><td rowspan="7">Is Teleport</td><td rowspan="5">Cause</td><td colspan="3">li32 enum</td></tr>
<tr><td>0</td><td>Unknown</td><td>TeleportCause is written only if Mode is MoveModeTeleport. It specifies the cause of the teleportation, which is one of the constants above.</td></tr>
<tr><td>1</td><td>Projectile</td><td></td></tr>
<tr><td>2</td><td>Chorus Fruit</td><td></td></tr>
<tr><td>3</td><td>Command</td><td></td></tr>
<tr><td>4</td><td>Behavior</td><td></td></tr>
<tr><td>Source Entity Type</td><td colspan="2">LegacyEntityType</td><td>TeleportSourceEntityType is the entity type that caused the teleportation, for example an ender pearl. TODO: is this still a integer and not a string?</td></tr>
<tr><td>Tick</td><td colspan="3">varint64</td><td colspan="3"></td></tr>
</table>

Packet Rider Jump

Bidirectional

Removed in 1.21.80

Field Name	Field Type	Notes
Jump Strength	zigzag32	

21 Packet Update Block

Clientbound

UpdateBlock is sent by the server to update a block client-side, without resending the entire chunk that the block is located in. It is particularly useful for small modifications like block breaking/placing.

Field Name	Field Type	Notes
Position	BlockCoordinates	Position is the block position at which a block is updated.
Block Runtime Id	varint	NewBlockRuntimeID is the runtime ID of the block that is placed at Position after sending the packet to the client.
Flags	UpdateBlockFlags	Flags is a combination of flags that specify the way the block is updated client-side. It is a combination of the flags above, but typically sending only the BlockUpdateNetwork flag is sufficient.
Layer	varint	Layer is the world layer on which the block is updated. For most blocks, this is the first layer, as that layer is the default layer to place blocks on, but for blocks inside of each other, this differs.

22 Packet Add Painting

Clientbound

Field Name	Field Type	Notes
Entity Id Self	zigzag64	
Runtime Entity Id	varint64	
Coordinates	vec3f	
Direction	zigzag32	

Field Name	Field Type	Notes
Title	string	

23 Bidirectional

Packet Tick Sync

TickSync is sent by the client and the server to maintain a synchronized, server-authoritative tick between the client and the server. The client sends this packet first, and the server should reply with another one of these packets, including the response time.

Field Name	Field Type	Notes
Request Time	li64	ClientRequestTimestamp is the timestamp on which the client sent this packet to the server. The server should fill out that same value when replying. The ClientRequestTimestamp is always 0
Response Time	li64	ServerReceptionTimestamp is the timestamp on which the server received the packet sent by the client. When the packet is sent by the client, this value is 0. ServerReceptionTimestamp is generally the current tick of the server. It isn't an actual timestamp, as the field implies

24 Bidirectional

Packet Level Sound Event Old

Field Name	Field Type	Notes
Sound Id	u8	
Position	vec3f	
Block Id	zigzag32	
Entity Type	zigzag32	
Is Baby Mob	bool	
Is Global	bool	

25 Clientbound

Packet Level Event

TODO: Check and verify old versions

Field Name	Field Type	Notes
Event	zigzag32 enum	

1000	Sound Click
1001	Sound Click Fail
1002	Sound Shoot
1003	Sound Door
1004	Sound Fizz
1005	Sound Ignite
1007	Sound Ghast
1008	Sound Ghast Shoot
1009	Sound Blaze Shoot
1010	Sound Door Bump
1012	Sound Door Crash
1018	Sound Enderman Teleport
1020	Sound Anvil Break
1021	Sound Anvil Use
1022	Sound Anvil Fall
1030	Sound Pop
1032	Sound Portal
1040	Sound Itemframe Add Item
1041	Sound Itemframe Remove
1042	Sound Itemframe Place
1043	Sound Itemframe Remove Item
1044	Sound Itemframe Rotate Item
1050	Sound Camera
1051	Sound Orb
1052	Sound Totem
1060	Sound Armor Stand Break
1061	Sound Armor Stand Hit

Field Name	Field Type		Notes
	1062	Sound Armor Stand Fall	
	1063	Sound Armor Stand Place	
	1064	Pointed Dripstone Land	
	1065	Dye Used	
	1066	Ink Sack Used	
	2000	Particle Shoot	
	2001	Particle Destroy	
	2002	Particle Splash	
	2003	Particle Eye Despawn	
	2004	Particle Spawn	
	2005	Particle Crop Growth	
	2006	Particle Guardian Curse	
	2007	Particle Death Smoke	
	2008	Particle Block Force Field	
	2009	Particle Projectile Hit	
	2010	Particle Dragon Egg Teleport	
	2011	Particle Crop Eaten	
	2012	Particle Critical	
	2013	Particle Enderman Teleport	
	2014	Particle Punch Block	
	2015	Particle Bubble	
	2016	Particle Evaporate	
	2017	Particle Destroy Armor Stand	
	2018	Particle Breaking Egg	
	2019	Particle Destroy Egg	
	2020	Particle Evaporate Water	
	2021	Particle Destroy Block No Sound	
	2022	Particle Knockback Roar	
	2023	Particle Teleport Trail	

Field Name	Field Type		Notes
	2024	Particle Point Cloud	
	2025	Particle Explosion	
	2026	Particle Block Explosion	
	2027	Particle Vibration Signal	
	2028	Particle Dripstone Drip	
	2029	Particle Fizz Effect	
	2030	Particle Wax On	
	2031	Particle Wax Off	
	2032	Particle Scrape	
	2033	Particle Electric Spark	
	2034	Particle Turtle Egg	
	2035	Particle Sculk Shriek	
	2036	Sculk Catalyst Bloom	
	2037	Sculk Charge	
	2038	Sculk Charge Pop	
	2039	Sonic Explosion	
	2040	Dust Plume	
	3001	Start Rain	
	3002	Start Thunder	
	3003	Stop Rain	
	3004	Stop Thunder	
	3005	Pause Game	
	3006	Pause Game No Screen	
	3007	Set Game Speed	
	3500	Redstone Trigger	
	3501	Cauldron Explode	
	3502	Cauldron Dye Armor	
	3503	Cauldron Clean Armor	
	3504	Cauldron Fill Potion	

Field Name	Field Type		Notes
	3505	Cauldron Take Potion	
	3506	Cauldron Fill Water	
	3507	Cauldron Take Water	
	3508	Cauldron Add Dye	
	3509	Cauldron Clean Banner	
	3600	Block Start Break	
	3601	Block Stop Break	
	3602	Block Break Speed	
	3603	Particle Punch Block Down	
	3604	Particle Punch Block Up	
	3605	Particle Punch Block North	
	3606	Particle Punch Block South	
	3607	Particle Punch Block West	
	3608	Particle Punch Block East	
	3609	Particle Shoot White Smoke	
	3610	Particle Breeze Wind Explosion	
	3611	Particle Trial Spawner Detection	
	3612	Particle Trial Spawner Spawning	
	3613	Particle Trial Spawner Ejecting	
	3614	Particle Wind Explosion	
	3615	Particle Wolf Armor Break	
	3616	Ominous Item Spawner	
	3617	Creaking Crumble	
	3618	Pale Oak Leaves	
	3619	Eyeblossom Open	
	3620	Eyeblossom Close	
	3621	Green Flame	
	4000	Set Data	
	9800	Players Sleeping	

Field Name	Field Type		Notes
	9801	Sleeping Players	
	9810	Jump Prevented	
	9811	Animation Vault Activate	
	9812	Animation Vault Deactivate	
	9813	Animation Vault Eject Item	
	9814	Animation Spawn Cobweb	
	9815	Add Particle Smash Attack Ground Dust	
	9816	Add Particle Creaking Heart Trail	
	16384	Add Particle Mask	
	16385	Add Particle Bubble	0x4000 | + particle ID
	16386	Add Particle Bubble Manual	
	16387	Add Particle Critical	
	16388	Add Particle Block Force Field	
	16389	Add Particle Smoke	
	16390	Add Particle Explode	
	16391	Add Particle Evaporation	
	16392	Add Particle Flame	
	16393	Add Particle Candle Flame	
	16394	Add Particle Lava	
	16395	Add Particle Large Smoke	
	16396	Add Particle Redstone	
	16397	Add Particle Rising Red Dust	
	16398	Add Particle Item Break	
	16399	Add Particle Snowball Poof	
	16400	Add Particle Huge Explode	
	16401	Add Particle Huge Explode Seed	
	16402	Add Particle Mob Flame	

Field Name	Field Type		Notes
	16403	Add Particle Heart	
	16404	Add Particle Terrain	
	16405	Add Particle Town Aura	
	16406	Add Particle Portal	
	16408	Add Particle Water Splash	
	16409	Add Particle Water Splash Manual	
	16410	Add Particle Water Wake	
	16411	Add Particle Drip Water	
	16412	Add Particle Drip Lava	
	16413	Add Particle Drip Honey	
	16414	Add Particle Stalactite Drip Water	
	16415	Add Particle Stalactite Drip Lava	
	16416	Add Particle Falling Dust	
	16417	Add Particle Mob Spell	
	16418	Add Particle Mob Spell Ambient	
	16419	Add Particle Mob Spell Instantaneous	
	16420	Add Particle Ink	
	16421	Add Particle Slime	
	16422	Add Particle Rain Splash	
	16423	Add Particle Villager Angry	
	16424	Add Particle Villager Happy	
	16425	Add Particle Enchantment Table	
	16426	Add Particle Tracking Emitter	
	16427	Add Particle Note	
	16428	Add Particle Witch Spell	
	16429	Add Particle Carrot	
	16430	Add Particle Mob Appearance	
	16431	Add Particle End Rod	

Field Name	Field Type		Notes
	16432	Add Particle Dragons Breath	
	16433	Add Particle Spit	
	16434	Add Particle Totem	
	16435	Add Particle Food	
	16436	Add Particle Fireworks Starter	
	16437	Add Particle Fireworks Spark	
	16438	Add Particle Fireworks Overlay	
	16439	Add Particle Balloon Gas	
	16440	Add Particle Colored Flame	
	16441	Add Particle Sparkler	
	16442	Add Particle Conduit	
	16443	Add Particle Bubble Column Up	
	16444	Add Particle Bubble Column Down	
	16445	Add Particle Sneeze	
	16446	Add Particle Shulker Bullet	
	16447	Add Particle Bleach	
	16448	Add Particle Dragon Destroy Block	
	16449	Add Particle Mycelium Dust	
	16450	Add Particle Falling Red Dust	
	16451	Add Particle Campfire Smoke	
	16452	Add Particle Tall Campfire Smoke	
	16453	Add Particle Dragon Breath Fire	
	16454	Add Particle Dragon Breath Trail	
	16455	Add Particle Blue Flame	
	16456	Add Particle Soul	
	16457	Add Particle Obsidian Tear	
	16458	Add Particle Portal Reverse	
	16459	Add Particle Snowflake	
	16460	Add Particle Vibration Signal	

Field Name	Field Type		Notes
	16461	Add Particle Sculk Sensor Redstone	
	16462	Add Particle Spore Blossom Shower	
	16463	Add Particle Spore Blossom Ambient	
	16464	Add Particle Wax	
	16465	Add Particle Electric Spark	
Position	vec3f		
Data	zigzag32		

26

Packet Block Event

Clientbound

Field Name	Field Type		Notes
Position	BlockCoordinates		Position is the position of the block that an event occurred at.
Type	zigzag32 enum		
	0	Sound	EventType is the type of the block event. The event type decides the way the event data that follows is used
	1	Change State	
Data	zigzag32		EventData holds event type specific data. For chests for example, opening the chest means the data must be 1

27

Packet Entity Event

Bidirectional

Field Name	Field Type		Notes
Runtime Entity Id	varint64		
Event Id	u8 enum		
	1	Jump	
	2	Hurt Animation	
	3	Death Animation	
	4	Arm Swing	
	5	Stop Attack	
	6	Tame Fail	
	7	Tame Success	
	8	Shake Wet	
	9	Use Item	
	10	Eat Grass Animation	
	11	Fish Hook Bubble	
	12	Fish Hook Position	
	13	Fish Hook Hook	
	14	Fish Hook Tease	
	15	Squid Ink Cloud	
	16	Zombie Villager Cure	
	18	Respawn	
	19	Iron Golem Offer Flower	
	20	Iron Golem Withdraw Flower	
	21	Love Particles	
	22	Villager Angry	
	23	Villager Happy	
	24	Witch Spell Particles	
	25	Firework Particles	
	26	In Love Particles	
	27	Silverfish Spawn Animation	
	28	Guardian Attack	

Field Name	Field Type		Notes
	29	Witch Drink Potion	
	30	Witch Throw Potion	
	31	Minecart Tnt Prime Fuse	
	32	Creeper Prime Fuse	
	33	Air Supply Expired	
	34	Player Add Xp Levels	
	35	Elder Guardian Curse	
	36	Agent Arm Swing	
	37	Ender Dragon Death	
	38	Dust Particles	
	39	Arrow Shake	
	57	Eating Item	
	60	Baby Animal Feed	
	61	Death Smoke Cloud	
	62	Complete Trade	
	63	Remove Leash	
	64	Caravan	
	65	Consume Totem	
	66	Player Check Treasure Hunter Achievement	
	67	Entity Spawn	
	68	Dragon Puke	
	69	Item Entity Merge	
	70	Start Swim	
	71	Balloon Pop	
	72	Treasure Hunt	
	73	Agent Summon	
	74	Charged Item	
	75	Fall	
	76	Grow Up	

Field Name	Field Type	Notes
	77 Vibration Detected	
	78 Drink Milk	
Data	zigzag32	

Clientbound

28 Packet Mob Effect

Field Name	Field Type	Notes
Runtime Entity Id	varint64	
Event Id	u8 enum	
	1 Add	
	2 Update	
	3 Remove	
Effect Id	zigzag32	
Amplifier	zigzag32	
Particles	bool	
Duration	zigzag32	
Tick	varint64	

Clientbound

29 Packet Update Attributes

Field Name	Field Type	Notes
Runtime Entity Id	varint64	
Attributes	PlayerAttributes	
Tick	varint64	

Bidirectional

30 Packet Inventory Transaction

InventoryTransaction is a packet sent by the client. It essentially exists out of multiple sub-packets, each of which have something to do with the inventory in one way or another. Some of these sub-packets directly relate to the inventory, others relate to interaction with the world, that could potentially result in a change in the inventory.

Field Name	Field Type	Notes
Transaction	Transaction	

31 Packet Mob Equipment

Bidirectional

Field Name	Field Type	Notes
Runtime Entity Id	varint64	
Item	Item	
Slot	u8	
Selected Slot	u8	
Window Id	WindowID	

32 Packet Mob Armor Equipment

Bidirectional

Field Name	Field Type	Notes
Runtime Entity Id	varint64	
Helmet	Item	
Chestplate	Item	
Leggings	Item	
Boots	Item	
Body	Item	

33 Packet Interact

Bidirectional

Interact is sent by the client when it interacts with another entity in some way. It used to be used for normal entity and block interaction, but this is no longer the case now.

Field Name	Field Type		Notes
Action Id	u8 enum		
	3	Leave Vehicle	Action type is the ID of the action that was executed by the player. It is one of the constants that may be found above.
	4	Mouse Over Entity	
	5	Npc Open	
	6	Open Inventory	
Target Entity Id	varint64		TargetEntityRuntimeID is the runtime ID of the entity that the player interacted with. This is empty for the InteractActionOpenInventory action type.
Position *if Action Id*	*Is Mouse Over Entity Or Leave Vehicle*	vec3f	Position associated with the ActionType above. For the InteractActionMouseOverEntity, this is the position relative to the entity moused over over which the player hovered with its mouse/touch. For the InteractActionLeaveVehicle, this is the position that the player spawns at after leaving the vehicle.

34

Packet Block Pick Request

Serverbound

Field Name	Field Type	Notes
X	zigzag32	
Y	zigzag32	
Z	zigzag32	
Add User Data	bool	
Selected Slot	u8	

Serverbound

35 Packet Entity Pick Request

Field Name	Field Type	Notes
Runtime Entity Id	lu64	
Selected Slot	u8	
With Data	bool	WithData is true if the pick request requests the entity metadata.

Serverbound

36 Packet Player Action

PlayerAction is sent by the client when it executes any action, for example starting to sprint, swim, starting the breaking of a block, dropping an item, etc.

Field Name	Field Type	Notes
Runtime Entity Id	varint64	EntityRuntimeID is the runtime ID of the player. The runtime ID is unique for each world session, and entities are generally identified in packets using this runtime ID.
Action	Action	ActionType is the ID of the action that was executed by the player. It is one of the constants that may be found above.
Position	BlockCoordinates	BlockPosition is the position of the target block, if the action with the ActionType set concerned a block. If that is not the case, the block position will be zero.
Result Position	BlockCoordinates	ResultPosition is the position of the action's result. When a UseItemOn action is sent, this is the position of the block clicked, but when a block is placed, this is the position at which the block will be placed.
Face	zigzag32	BlockFace is the face of the target block that was touched. If the action with the ActionType set concerned a block. If not, the face is always 0.

38 Clientbound

Packet Hurt Armor

Field Name	Field Type	Notes
Cause	zigzag32	
Damage	zigzag32	
Armor Slots	zigzag64	

39 Bidirectional

Packet Set Entity Data

Field Name	Field Type	Notes
Runtime Entity Id	varint64	
Metadata	MetadataDictionary	
Properties	EntityProperties	EntityProperties holds lists of entity properties that define specific attributes of an entity. As of v1.19.40, the vanilla server does not use these properties, however they are still supported by the protocol.
Tick	varint64	

40 Bidirectional

Packet Set Entity Motion

SetActorMotion is sent by the server to change the client-side velocity of an entity. It is usually used in combination with server-side movement calculation.

Field Name	Field Type	Notes
Runtime Entity Id	varint64	EntityRuntimeID is the runtime ID of the entity. The runtime ID is unique for each world session, and entities are generally identified in packets using this runtime ID.
Velocity	vec3f	Velocity is the new velocity the entity gets. This velocity will initiate the client-side movement of the entity.

Field Name	Field Type	Notes
Tick	varint64	

41 Packet Set Entity Link

Clientbound

SetActorLink is sent by the server to initiate an entity link client-side, meaning one entity will start riding another.

Field Name	Field Type	Notes
Link	Link	

42 Packet Set Health

Clientbound

Field Name	Field Type	Notes
Health	zigzag32	

43 Packet Set Spawn Position

Clientbound

Field Name	Field Type		Notes
Spawn Type	zigzag32 enum		
	0	Player	
	1	World	
Player Position	BlockCoordinates		
Dimension	zigzag32		
World Position	BlockCoordinates		

44 Packet Animate

Bidirectional

Field Name	Field Type		Notes
Action Id	zigzag32 enum		
	0	None	Unused
	1	Swing Arm	Server bound notification to swing the player's arm. Server is expected to rebroadcast to all that should see the arm move. See also PlayerAuthInputPacket::InputData::MissedSwing for a very similar action
	2	Unknown	
	3	Wake Up	Client bound notification to stop sleeping in a bed
	4	Critical Hit	Client-bound notification to play critical hit particles
	5	Magic Critical Hit	Unused
	128	Row Right	Sent every tick the client is in a boat exclusively in legacy client authoritative movement. See Player Auth Input for how to compute this in the latest protocol.
	129	Row Left	Sent every tick the client is in a boat exclusively in legacy client authoritative movement. See Player Auth Input for how to compute this in the latest protocol.
Runtime Entity Id	varint64		
🔁 *if Action Id*	*Is Row Right Or Row Left*	**Boat Rowing Time**	lf32

45

Packet Respawn

Bidirectional

Field Name	Field Type	Notes
Position	vec3f	

Field Name	Field Type	Notes
State	u8	
Runtime Entity Id	varint64	

46 Packet Container Open

Clientbound

ContainerOpen is sent by the server to open a container client-side. This container must be physically present in the world, for the packet to have any effect. Unlike Java Edition, Bedrock Edition requires that chests for example must be present and in range to open its inventory.

Field Name	Field Type	Notes
Window Id	WindowID	WindowID is the ID representing the window that is being opened. It may be used later to close the container using a ContainerClose packet.
Window Type	WindowType	ContainerType is the type ID of the container that is being opened when opening the container at the position of the packet. It depends on the block/entity, and could, for example, be the window type of a chest or a hopper, but also a horse inventory.
Coordinates	BlockCoordinates	ContainerPosition is the position of the container opened. The position must point to a block entity that actually has a container. If that is not the case, the window will not be opened and the packet will be ignored, if a valid ContainerEntityUniqueID has not also been provided.
Runtime Entity Id	zigzag64	ContainerEntityUniqueID is the unique ID of the entity container that was opened. It is only used if the ContainerType is one that points to an entity, for example a horse.

47 Packet Container Close

Bidirectional

ContainerClose is sent by the server to close a container the player currently has opened, which was opened using the ContainerOpen packet, or by the client to tell the server it closed a particular container, such as the crafting grid.

Field Name	Field Type	Notes
Window Id	WindowID	WindowID is the ID representing the window of the container that should be closed. It must be equal to the one sent in the ContainerOpen packet to close the designated window.
Window Type	WindowType	ContainerType is the type ID of the container that is being opened when opening the container at the position of the packet. It depends on the block/entity, and could, for example, be the window type of a chest or a hopper, but also a horse inventory.
Server	bool	ServerSide determines whether or not the container was force-closed by the server. If this value is not set correctly, the client may ignore the packet and respond with a PacketViolationWarning.

48

Packet Player Hotbar

Bidirectional

PlayerHotBar is sent by the server to the client. It used to be used to link hot bar slots of the player to actual slots in the inventory, but as of 1.2, this was changed and hot bar slots are no longer a free floating part of the inventory. Since 1.2, the packet has been re-purposed, but its new functionality is not clear.

Field Name	Field Type	Notes
Selected Slot	varint	
Window Id	WindowID	
Select Slot	bool	

49

Packet Inventory Content

Bidirectional

InventoryContent is sent by the server to update the full content of a particular inventory. It is usually sent for the main inventory of the player, but also works for other inventories that are currently opened by the player.

Field Name	Field Type	Notes
Window Id	WindowIDVarint	WindowID is the ID that identifies one of the windows that the client currently has opened, or one of the consistent windows such as the main inventory.
Input	ItemStacks	Content is the new content of the inventory. The length of this slice must be equal to the full size of the inventory window updated.
Container	FullContainerName	Container is the protocol.FullContainerName that describes the container that the content is for.
Storage Item	Item	storage_item is the item that is acting as the storage container for the inventory. If the inventory is not a dynamic container then this field should be left empty. When set, only the item type is used by the client and none of the other stack info.

50

Packet Inventory Slot

Bidirectional

InventorySlot is sent by the server to update a single slot in one of the inventory windows that the client currently has opened. Usually this is the main inventory, but it may also be the off hand or, for example, a chest inventory.

Field Name	Field Type	Notes
Window Id	WindowIDVarint	WindowID is the ID of the window that the packet modifies. It must point to one of the windows that the client currently has opened.
Slot	varint	Slot is the index of the slot that the packet modifies. The new item will be set to the slot at this index.
Container	FullContainerName	Container is the protocol.FullContainerName that describes the container that the content is for.
Storage Item	Item	storage_item is the item that is acting as the storage container for the inventory. If the inventory is not a dynamic container then this field should be left empty. When set, only the

Field Name	Field Type	Notes
		item type is used by the client and none of the other stack info.
Item	Item	NewItem is the item to be put in the slot at Slot. It will overwrite any item that may currently be present in that slot.

51 Packet Container Set Data

Clientbound

ContainerSetData is sent by the server to update specific data of a single container, meaning a block such as a furnace or a brewing stand. This data is usually used by the client to display certain features client-side.

Field Name	Field Type	Notes
Window Id	WindowID	WindowID is the ID of the window that should have its data set. The player must have a window open with the window ID passed, or nothing will happen.
Property	zigzag32	Key is the key of the property. It is one of the constants that can be found above. Multiple properties share the same key, but the functionality depends on the type of the container that the data is set to. IF FURNACE: 0: furnace*tick*count 1: furnace*lit*time 2: furnace*lit*duration 3: furnace*stored*xp 4: furnace*fuel*aux IF BREWING STAND: 0: brew*time 1: brew*fuel*amount 2: brew*fuel_total
Value	zigzag32	Value is the value of the property. Its use differs per property.

52 Packet Crafting Data

Clientbound

Field Name	Field Type	Notes
Recipes	Recipes	
Potion Type Recipes	PotionTypeRecipes	PotionContainerChangeRecipes is a list of all recipes to convert a potion from one type to another, such as from a drinkable potion

Field Name	Field Type	Notes
		to a splash potion, or from a splash potion to a lingering potion.
Potion Container Recipes	PotionContainerChangeRecipes	
Material Reducers length	varint	MaterialReducers is a list of all material reducers which is used in education edition chemistry.
Material Reducers array	MaterialReducer	
Clear Recipes	bool	ClearRecipes indicates if all recipes currently active on the client should be cleaned. Doing this means that the client will have no recipes active by itself: Any CraftingData packets previously sent will also be discarded, and only the recipes in this CraftingData packet will be used.

53 **Packet Crafting Event** Bidirectional

CraftingEvent is sent by the client when it crafts a particular item. Note that this packet may be fully ignored, as the InventoryTransaction packet provides all the information required.

Field Name	Field Type		Notes
Window Id	WindowID		WindowID is the ID representing the window that the player crafted in.
Recipe Type	zigzag32 enum		
	0	Inventory	CraftingType is a type that indicates the way the crafting was done, for example if a crafting table was used.
	1	Crafting	

Field Name	Field Type	Notes
	2 Workbench	
Recipe Id	uuid	RecipeUUID is the UUID of the recipe that was crafted. It points to the UUID of the recipe that was sent earlier in the CraftingData packet.
Input length	varint	Input is a list of items that the player put into the recipe so that it could create the Output items. These items are consumed in the process.
Input array	Item	
Result length	varint	Output is a list of items that were obtained as a result of crafting the recipe.
Result array	Item	

54 ## Packet Gui Data Pick Item

Clientbound

GUIDataPickItem is sent by the server to make the client 'select' a hot bar slot. It currently appears to be broken however, and does not actually set the selected slot to the hot bar slot set in the packet.

Field Name	Field Type	Notes
Item Name	string	ItemName is the name of the item that shows up in the top part of the popup that shows up when selecting an item. It is shown as if an item was selected by the player itself.
Item Effects	string	ItemEffects is the line under the ItemName, where the effects of the item are usually situated.
Hotbar Slot	li32	HotBarSlot is the hot bar slot to be selected/picked. This does not currently work, so it does not matter what number this is.

55 ## Packet Adventure Settings

Bidirectional

AdventureSettings is sent by the server to update game-play related features, in particular permissions to access these features for the client. It includes allowing the

player to fly, build and mine, and attack entities. Most of these flags should be checked server-side instead of using this packet only. The client may also send this packet to the server when it updates one of these settings through the in-game settings interface. The server should verify if the player actually has permission to update those settings.

Field Name	Field Type	Notes
Flags	AdventureFlags	Flags is a set of flags that specify certain properties of the player, such as whether or not it can fly and/or move through blocks. It is one of the AdventureFlag constants above.
Command Permission	CommandPermissionLevelVarint	CommandPermissionLevel is a permission level that specifies the kind of commands that the player is allowed to use.
Action Permissions	ActionPermissions	ActionPermissions is, much like Flags, a set of flags that specify actions that the player is allowed to undertake, such as whether it is allowed to edit blocks, open doors etc. It is a combination of the ActionPermission constants above.
Permission Level	PermissionLevel	PermissionLevel is the permission level of the player as it shows up in the player list built up using the PlayerList packet. It is one of the PermissionLevel constants above.
Custom Stored Permissions	varint	Custom permissions
User Id	li64	PlayerUniqueID is a unique identifier of the player. It appears it is not required to fill this field out with a correct value. Simply writing 0 seems to work.

56 Packet Block Entity Data

Bidirectional

Field Name	Field Type	Notes
Position	BlockCoordinates	
Nbt	nbt	

57 Packet Player Input

Serverbound

Removed in 1.21.80

Field Name	Field Type	Notes
Motion X	lf32	
Motion Z	lf32	
Jumping	bool	
Sneaking	bool	

58 Packet Level Chunk

Clientbound

LevelChunk is sent by the server to provide the client with a chunk of a world data (16xYx16 blocks). Typically a certain amount of chunks is sent to the client before sending it the spawn PlayStatus packet, so that the client spawns in a loaded world.

Field Name	Field Type	Notes
X	zigzag32	ChunkX is the X coordinate of the chunk sent. (To translate a block's X to a chunk's X: x >> 4)
Z	zigzag32	ChunkZ is the Z coordinate of the chunk sent. (To translate a block's Z to a chunk's Z: z >> 4)
Dimension	zigzag32	
Sub Chunk Count	varint	SubChunkCount is the amount of sub chunks that are part of the chunk sent. Depending on if the cache is enabled, a list of blob hashes will be sent, or, if disabled, the sub chunk data. On newer versions, if this is a negative value it indicates to use the Subchunk Polling mechanism

Field Name	Field Type		Notes
Highest Subchunk Count *if Sub Chunk Count*			
	Is -2	lu16	HighestSubChunk is the highest sub-chunk at the position that is not all air. It is only set if the RequestMode is set to protocol.SubChunkRequestModeLimited.
Cache Enabled	bool		CacheEnabled specifies if the client blob cache should be enabled. This system is based on hashes of blobs which are consistent and saved by the client in combination with that blob, so that the server does not have to send the same chunk multiple times. If the client does not yet have a blob with the hash sent, it will send a ClientCacheBlobStatus packet containing the hashes is does not have the data of.
Blobs *if Cache Enabled*	*Is True*	**Hashes length**	varint	BlobHashes is a list of all blob hashes used in the chunk. It is composed of SubChunkCount + 1 hashes, with the first SubChunkCount hashes being those of the sub chunks and the last one that of the biome of the chunk. If CacheEnabled is set to false, BlobHashes can be left empty.
		Hashes array	lu64	
Payload	ByteArray		RawPayload is a serialised string of chunk data. The data held depends on if CacheEnabled is set to true. If set to false, the payload is composed of multiple sub-chunks, each of which carry a version which indicates the way they are serialised, followed by biomes, border blocks and tile entities. If CacheEnabled is true, the payload consists out of the border blocks and tile entities only.

59 **Packet Set Commands Enabled** Clientbound

Field Name	Field Type	Notes
Enabled	bool	

60 **Packet Set Difficulty** Clientbound

Field Name	Field Type	Notes
Difficulty	varint	

61 ## Packet Change Dimension

Clientbound

Field Name	Field Type	Notes
Dimension	zigzag32	
Position	vec3f	
Respawn	bool	
Loading Screen Id **Optional**	lu32	

62 ## Packet Set Player Game Type

Bidirectional

SetPlayerGameType is sent by the server to update the game type (game mode) of the player

Field Name	Field Type	Notes
Gamemode	GameMode	The new gamemode for the player. Some of these game types require additional flags to be set in an AdventureSettings packet for the game mode to obtain its full functionality.

63 ## Packet Player List

Clientbound

Field Name	Field Type	Notes
Records	PlayerRecords	

64 ## Packet Simple Event

Clientbound

Field Name	Field Type	Notes
Event Type	lu16 enum	
	0	Uninitialized Subtype
	1	Enable Commands
	2	Disable Commands
	3	Unlock World Template Settings

65

Packet Event

Clientbound

Event is sent by the server to send an event with additional data. It is typically sent to the client for telemetry reasons, much like the SimpleEvent packet.

Field Name	Field Type	Notes
Runtime Id	varint64	
Event Type	zigzag32 enum	
	0	Achievement Awarded
	1	Entity Interact
	2	Portal Built
	3	Portal Used
	4	Mob Killed
	5	Cauldron Used
	6	Player Death
	7	Boss Killed
	8	Agent Command
	9	Agent Created
	10	Banner Pattern Removed
	11	Command Executed
	12	Fish Bucketed
	13	Mob Born
	14	Pet Died

Field Name	Field Type		Notes
	15	Cauldron Block Used	
	16	Composter Block Used	
	17	Bell Block Used	
	18	Actor Definition	
	19	Raid Update	
	20	Player Movement Anomaly	
	21	Player Movement Corrected	
	22	Honey Harvested	
	23	Target Block Hit	
	24	Piglin Barter	
	25	Waxed Or Unwaxed Copper	
	26	Code Builder Runtime Action	
	27	Code Builder Scoreboard	
	28	Strider Ridden In Lava In Overworld	
	29	Sneak Close To Sculk Sensor	
	30	Careful Restoration	
	31	Item Used	
Use Player Id	u8		
Event Data	restBuffer		

66

Packet Spawn Experience Orb

Clientbound

Field Name	Field Type	Notes
Position	vec3f	
Count	zigzag32	

Packet Clientbound Map Item Data

ClientBoundMapItemData is sent by the server to the client to update the data of a map shown to the client. It is sent with a combination of flags that specify what data is updated. The ClientBoundMapItemData packet may be used to update specific parts of the map only. It is not required to send the entire map each time when updating one part.

Field Name	Field Type		Notes
Map Id	zigzag64		MapID is the unique identifier that represents the map that is updated over network. It remains consistent across sessions.
Update Flags	UpdateMapFlags		UpdateFlags is a combination of flags found above that indicate what parts of the map should be updated client-side.
Dimension	u8		Dimension is the dimension of the map that should be updated, for example the overworld (0), the nether (1) or the end (2).
Locked	bool		LockedMap specifies if the map that was updated was a locked map, which may be done using a cartography table.
Origin	vec3i		Origin is the center position of the map being updated.
Included In *if Update Flags.Initialisation*	*Is True* **length**	varint	The following fields apply only for the MapUpdateFlagInitialisation. MapsIncludedIn holds an array of map IDs that the map updated is included in. This has to do with the scale of the map: Each map holds its own map ID and all map IDs of maps that include this map and have a bigger scale. This means that a scale 0 map will have 5 map IDs in this slice, whereas a scale 4 map will have only 1 (its own). The actual use of this field remains unknown.
	Is True array	zigzag64	
Scale *if Update Flags.Initialisation || Update Flags.Decoration || Update Flags.Texture*	*Is True*	u8	Scale is the scale of the map as it is shown in-game. It is written when any of the MapUpdateFlags are set to the UpdateFlags field.

Field Name	Field Type			Notes
Tracked *if Update Flags.Decoration*	*Is True*	**Objects length**	varint	The following fields apply only for the MapUpdateFlagDecoration. TrackedObjects is a list of tracked objects on the map, which may either be entities or blocks. The client makes sure these tracked objects are actually tracked. (position updated etc.)
		Objects array	TrackedObject	
		Decorations length	varint	
		Decorations array	MapDecoration	
Texture *if Update Flags.Texture*	*Is True*	**Width**	zigzag32	Updates to the map contents itself (texture) Width is the width of the texture area that was updated. The width may be a subset of the total width of the map.
		Height	zigzag32	Height is the height of the texture area that was updated. The height may be a subset of the total height of the map
		X Offset	zigzag32	XOffset is the X offset in pixels at which the updated texture area starts. From this X, the updated texture will extend exactly Width pixels to the right.
		Y Offset	zigzag32	YOffset is the Y offset in pixels at which the updated texture area starts. From this Y, the updated texture will extend exactly Height pixels up.
		Pixels length	varint	Pixels is a list of pixel colours for the new texture of the map. It is indexed as Pixels[y][x], with the length of the outer slice having to be exactly Height long and the inner slices exactly Width long. To access this array, use $width * y + x
		Pixels array	varint	

68 ## Packet Map Info Request

Bidirectional

Field Name	Field Type	Notes
Map Id	zigzag64	
Client Pixels length	lu32	ClientPixels is a map of pixels sent from the client to notify the server about the pixels that it isn't aware of.
Client Pixels array	**Rgba**	li32
	Index	lu16

69 ## Packet Request Chunk Radius

Bidirectional

RequestChunkRadius is sent by the client to the server to update the server on the chunk view radius that it has set in the settings. The server may respond with a ChunkRadiusUpdated packet with either the chunk radius requested, or a different chunk radius if the server chooses so.

Field Name	Field Type	Notes
Chunk Radius	zigzag32	ChunkRadius is the requested chunk radius. This value is always the value set in the settings of the player.
Max Radius	u8	

70 ## Packet Chunk Radius Update

Clientbound

ChunkRadiusUpdated is sent by the server in response to a RequestChunkRadius packet. It defines the chunk radius that the server allows the client to have. This may be lower than the chunk radius requested by the client in the RequestChunkRadius packet.

Field Name	Field Type	Notes
Chunk Radius	zigzag32	ChunkRadius is the final chunk radius that the client will adapt when it receives the packet. It does not have to be the same as the requested chunk radius.

72 ## Packet Game Rules Changed

Clientbound

Field Name	Field Type	Notes
Rules length	varint	
Rules array	GameRuleI32	

73 ## Packet Camera

Clientbound

Camera is sent by the server to use an Education Edition camera on a player. It produces an image client-side.

Field Name	Field Type	Notes
Camera Entity Unique Id	zigzag64	CameraEntityUniqueID is the unique ID of the camera entity from which the picture was taken.
Target Player Unique Id	zigzag64	TargetPlayerUniqueID is the unique ID of the target player. The unique ID is a value that remains consistent across different sessions of the same world, but most servers simply fill the runtime ID of the player out for this field.

74 ## Packet Boss Event

Bidirectional

Field Name	Field Type		Notes
Boss Entity Id	zigzag64		
Type	varint enum		
	0	Show Bar	S2C: Shows the boss-bar to the player.

Field Name	Field Type		Notes
	1	Register Player	C2S: Registers a player to a boss fight.
	2	Hide Bar	S2C: Removes the boss-bar from the client.
	3	Unregister Player	C2S: Unregisters a player from a boss fight.
	4	Set Bar Progress	S2C: Sets the bar percentage.
	5	Set Bar Title	S2C: Sets title of the bar.
	6	Update Properties	S2C: darkens the sky
	7	Texture	S2C: Not implemented :(Intended to alter bar appearance, but these currently produce no effect on client-side whatsoever.
	8	Query	C2S: Client asking the server to resend all boss data.
🔁 *if Type*	*Is Show Bar*	**Title**	string	BossBarTitle is the title shown above the boss bar. It currently does not function, and instead uses the name tag of the boss entity at all times. It is only set if the EventType is BossEventShow or BossEventTitle.
		Filtered Title	string	FilteredBossBarTitle is a filtered version of BossBarTitle with all the profanity removed. The client will use this over BossBarTitle if this field is not empty and they have the "Filter Profanity" setting enabled.
		Progress	lf32	HealthPercentage is the percentage of health that is shown in the boss bar. It currently does not function, and instead uses the health percentage of the boss entity at all times. It is only set if the EventType is BossEventShow or BossEventHealthPercentage.
		Screen Darkening	li16	ScreenDarkening currently seems not to do anything.

Field Name	Field Type			Notes
		Color	varint	Colour is the colour of the boss bar that is shown when a player is subscribed. It currently does not function. It is only set if the EventType is BossEventShow, BossEventAppearanceProperties or BossEventTexture. Format is ARGB
		Overlay	varint	Overlay is the overlay of the boss bar that is shown on top of the boss bar when a player is subscribed. It currently does not function. It is only set if the EventType is BossEventShow, BossEventAppearanceProperties or BossEventTexture.
	Is Register Player Or Unregister Player Or Query	**Player Id**	zigzag64	PlayerUniqueID is the unique ID of the player that is registered to or unregistered from the boss fight. It is set if EventType is either BossEventRegisterPlayer or BossEventUnregisterPlayer.
	Is Set Bar Progress	**Progress**		lf32
	Is Set Bar Title	**Title**	string	
		Filtered Title	string	FilteredBossBarTitle is a filtered version of BossBarTitle with all the profanity removed. The client will use this over BossBarTitle if this field is not empty and they have the "Filter Profanity" setting enabled.
	Is Update Properties	**Screen Darkening**		li16
		Color		varint
		Overlay		varint
	Is Texture	**Color**		varint
		Overlay		varint

Field Name	Field Type	Notes
Runtime Entity Id	varint64	
Status	zigzag32	

76

Packet Available Commands

Clientbound

This packet sends a list of commands to the client. Commands can have arguments, and some of those arguments can have 'enum' values, which are a list of possible values for the argument. The serialization is rather complex and involves palettes like chunks.

Field Name	Field Type	Notes
Values Len	varint	The length of the enums for all the command parameters in this packet
_enum_type	["enum_size_based_on_values_len"]	Not read from stream: instead calculated from the values_len field If the values*len < 0xff => byte, If the values*len < 0xffff => short, If the values_len < 0xffffff => int
*Length for Enum Values below is **Values Len** from above*		Here all the enum values for all of the possible commands are stored to one array palette
Enum Values array	string	
Chained Subcommand Values length	varint	chained*subcommand*values is a slice of all chained subcommand names. chained*subcommand*values generally should contain each possible value only once. chained_subcommands are built by pointing to entries in this slice.
Chained Subcommand Values array	string	
Suffixes length	varint	Integer parameters may sometimes have a prefix, such as the XP command: /xp [player: target] <- here, the xp command gives experience points /xp L [player: target] <- here, the xp command gives experience levels This is the palette of suffixes
Suffixes array	string	
Enums length	varint	The list of enum objects

<table>
<tr><th>Field Name</th><th colspan="5">Field Type</th><th>Notes</th></tr>
<tr><td rowspan="5">Enums
array</td><td>Name</td><td colspan="4">string</td><td>The name of the enum</td></tr>
<tr><td>Values length</td><td colspan="4">varint</td><td>The values in the enum</td></tr>
<tr><td rowspan="3">Values
array</td><td rowspan="3">🔁
if ../ Enum Type</td><td>Is Byte</td><td>u8</td><td rowspan="3">The indexes to value in the palette</td></tr>
<tr><td>Is Short</td><td>lu16</td></tr>
<tr><td>Is Int</td><td>lu32</td></tr>
<tr><td>Chained Subcommands length</td><td colspan="5">varint</td><td>chained_subcommands is a slice of all subcommands that are followed by a chained command. An example usage of this is /execute which allows you to run another command as another entity or at a different position etc.</td></tr>
<tr><td rowspan="4">Chained Subcommands
array</td><td>Name</td><td colspan="2">string</td><td colspan="2">ChainedSubcommandValue represents the value for a chained subcommand argument. name is the name of the chained subcommand and shows up in the list as a regular subcommand enum.</td></tr>
<tr><td>Values length</td><td colspan="2">varint</td><td colspan="2">values contains the index and parameter type of the chained subcommand.</td></tr>
<tr><td rowspan="2">Values
array</td><td>Index</td><td>lu16</td><td colspan="2">index is the index of the argument in the ChainedSubcommandValues slice from the AvailableCommands packet. This is then used to set the type specified by the Value field below.</td></tr>
<tr><td>Value</td><td>lu16</td><td colspan="2">value is a combination of the flags above and specified the type of argument. Unlike regular parameter types, this should NOT contain any of the special flags (valid, enum, suffixed or soft enum) but only the basic types.</td></tr>
<tr><td>Command Data length</td><td colspan="5">varint</td><td></td></tr>
<tr><td rowspan="5">Command Data
array</td><td>Name</td><td colspan="5">string</td></tr>
<tr><td>Description</td><td colspan="5">string</td></tr>
<tr><td>Flags</td><td colspan="5">lu16</td></tr>
<tr><td>Permission Level</td><td colspan="5">u8</td></tr>
<tr><td>Alias</td><td colspan="5">li32</td></tr>
</table>

Field Name	Field Type		Notes
	Chained Subcommand Offsets length	varint	chained*subcommand*offsets is a slice of offsets that all point to a different chained*subcommand from the chained*subcommands slice in the available_commands packet.
	Chained Subcommand Offsets array	lu16	
	Overloads length	varint	The list of overload parameters for this command
	Overloads array	**Chaining**	bool — chaining determines if the parameters use chained subcommands or not.
		Parameters length	varint — Each of the parameters gets an array of posible overloads
		Parameters array	**Parameter Name** — string — The name of the parameter shown to the user (the amount in /xp <amount: int>)
			Value Type — lu16 enum

1	Int
3	Float
4	Value
5	Wildcard Int
6	Operator
7	Command Operator
8	Target
10	Wildcard Target
17	File Path
23	Integer Range
43	Equipment Slots
44	String
52	Block Position

Field Name	Field Type		Notes		
				53	Position
				55	Message
				58	Raw Text
				62	Json
				71	Block States
				74	Command
			Enum Type	lu16 enum	
				16	Valid	In MC, this + prior field are combined to one 32bit bitfield
				48	Enum	
				256	Suffixed	
				1040	Soft Enum	
			Optional	bool	Is this parameter required?
			Options	CommandFlags	Additinal options for this command (thanks macroshaft…)
Dynamic Enums length	varint		There are two types of enums: static enums which cannot be changed after sending AvaliableCommands, (unless you resend the whole packet) and 'soft' or 'dynamic' enums like below which is an array that can be updated with the UpdateSoftEnum packet		
Dynamic Enums array	**Name**		string		
	Values length		varint		
	Values array		string		

Field Name	Field Type	Notes
Enum Constraints length	varint	
Enum Constraints array	**Value Index**	li32
	Enum Index	li32
	Constraints length	varint
	Constraints array	**Constraint**: u8 enum 0 Cheats Enabled 1 Operator Permissions 2 Host Permissions

77

Packet Command Request

Serverbound

ParamOptionCollapseEnum specifies if the enum (only if the Type is actually an enum type. If not, setting this to true has no effect) should be collapsed. This means that the options of the enum are never shown in the actual usage of the command, but only as auto-completion, like it automatically does with enums that have a big amount of options. To illustrate, it can make <$Name: bool>. enum*size*based*on*values_len: native CommandRequest is sent by the client to request the execution of a server-side command. Although some servers support sending commands using the Text packet, this packet is guaranteed to have the correct result.

Field Name	Field Type	Notes
Command	string	CommandLine is the raw entered command line. The client does no parsing of the command line by itself (unlike it did in the early stages), but lets the server do that.
Origin	CommandOrigin	Origin holds information about the command sender that will be returnd back in the command response
Internal	bool	Internal specifies if the command request internal. Setting it to false seems to work and the usage of this field is not known.

Field Name	Field Type	Notes
Version	varint	Specifies the version of the command to run, relative to the current Minecraft version. Should be set to 52 as of 1.19.62

78

Packet Command Block Update

Serverbound

CommandBlockUpdate is sent by the client to update a command block at a specific position. The command block may be either a physical block or an entity.

Field Name	Field Type	Notes		
Is Block	bool	Block specifies if the command block updated was an actual physical block. If false, the command block is in a minecart and has an entity runtime ID instead.		
🔁 *if Is Block*	*Is True*	**Position**	BlockCoordinates	Position is the position of the command block updated. It is only set if Block is set to true. Nothing happens if no command block is set at this position. Position is the position of the command block updated. It is only set if Block is set to true. Nothing happens if no command block is set at this position.
		Mode	varint enum	
			0 Impulse	Mode is the mode of the command block. It is either CommandBlockImpulse, CommandBlockChain or CommandBlockRepeat. It is only set if Block is set to true.
			1 Repeat	
			2 Chain	

Field Name	Field Type	Notes		
		Needs Redstone	bool	NeedsRedstone specifies if the command block needs to be powered by redstone to be activated. If false, the command block is always active. The field is only set if Block is set to true.
		Conditional	bool	Conditional specifies the behaviour of the command block if the command block before it (the opposite side of the direction the arrow if facing) fails to execute. If set to false, it will activate at all times, whereas if set to true, it will activate only if the previous command block executed successfully. The field is only set if Block is set to true.
	Is False	**Minecart Entity Runtime Id**		varint64
Command	string	Command is the command currently entered in the command block. This is the command that is executed when the command block is activated.		
Last Output	string	LastOutput is the output of the last command executed by the command block. It may be left empty to show simply no output at all, in combination with setting ShouldTrackOutput to false.		
Name	string	Name is the name of the command block updated. If not empty, it will show this name hovering above the command block when hovering over the block with the cursor.		
Filtered Name	string	FilteredName is a filtered version of Name with all the profanity removed. The client will use this over Name if this field is not empty and they have the "Filter Profanity" setting enabled.		

Field Name	Field Type	Notes
Should Track Output	bool	ShouldTrackOutput specifies if the command block tracks output. If set to false, the output box won't be shown within the command block.
Tick Delay	li32	TickDelay is the delay in ticks between executions of a command block, if it is a repeating command block.
Execute On First Tick	bool	ExecuteOnFirstTick specifies if the command block should execute on the first tick, AKA as soon as the command block is enabled.

79 Packet Command Output

Clientbound

Field Name	Field Type		Notes
Origin	CommandOrigin		CommandOrigin is the data specifying the origin of the command. In other words, the source that the command request was from, such as the player itself or a websocket server. The client forwards the messages in this packet to the right origin, depending on what is sent here.
Output Type	i8 enum		
	1	Last	OutputType specifies the type of output that is sent.
	2	Silent	
	3	All	
	4	Data Set	
Success Count	varint		SuccessCount is the amount of times that a command was executed successfully as a result of the command that was requested. For servers, this is usually a rather meaningless fields, but for vanilla, this is applicable for commands created with Functions.
Output length	varint		OutputMessages is a list of all output messages that should be sent to the player. Whether they are shown or not, depends on the type of the messages.

Field Name	Field Type		Notes
Output array	**Success**	bool	Success indicates if the output message was one of a successful command execution. If set to true, the output message is by default coloured white, whereas if set to false, the message is by default coloured red.
	Message Id	string	Message is the message that is sent to the client in the chat window. It may either be simply a message or a translated built-in string like 'commands.tp.success.coordinates', combined with specific parameters below.
	Parameters length	varint	Parameters is a list of parameters that serve to supply the message sent with additional information, such as the position that a player was teleported to or the effect that was applied to an entity. These parameters only apply for the Minecraft built-in command output.
	Parameters array	string	
Data Set *if Output Type*			
	Is Data Set		string
	Default		void

80

Packet Update Trade

Clientbound

UpdateTrade is sent by the server to update the trades offered by a villager to a player. It is sent at the moment that a player interacts with a villager.

Field Name	Field Type	Notes
Window Id	WindowID	WindowID is the ID that identifies the trading window that the client currently has opened.
Window Type	WindowType	WindowType is an identifier specifying the type of the window opened. In vanilla, it appears this is always filled out with 15.
Size	varint	Size is the amount of trading options that the villager has.

Field Name	Field Type	Notes
Trade Tier	varint	TradeTier is the tier of the villager that the player is trading with. The tier starts at 0 with a first two offers being available, after which two additional offers are unlocked each time the tier becomes one higher.
Villager Unique Id	varint64	VillagerUniqueID is the unique ID of the villager entity that the player is trading with. The TradeTier sent above applies to this villager.
Entity Unique Id	varint64	EntityUniqueID is the unique ID of the entity (usually a player) for which the trades are updated. The updated trades may apply only to this entity.
Display Name	string	DisplayName is the name displayed at the top of the trading UI. It is usually used to represent the profession of the villager in the UI.
New Trading Ui	bool	NewTradeUI specifies if the villager should be using the new trade UI (The one added in 1.11.) rather than the old one. This should usually be set to true.
Economic Trades	bool	Trading based on Minecraft economy - specifies if the prices of the villager's offers are modified by an increase in demand for the item. (A mechanic added in 1.11.) Buying more of the same item will increase the price of that particular item. https://minecraft.wiki/w/Trading#Economics
Offers	nbt	NBT serialised compound of offers that the villager has.

81

Packet Update Equipment

Clientbound

UpdateEquip is sent by the server to the client upon opening a horse inventory. It is used to set the content of the inventory and specify additional properties, such as the items that are allowed to be put in slots of the inventory.

Field Name	Field Type	Notes
Window Id	WindowID	WindowID is the identifier associated with the window that the UpdateEquip packet concerns. It is the ID sent for the horse inventory that was opened before this packet was sent.

Field Name	Field Type	Notes
Window Type	WindowType	WindowType is the type of the window that was opened. Generally, this is the type of a horse inventory, as the packet is specifically made for that.
Size	u8	Size is the size of the horse inventory that should be opened. A bigger size does, in fact, change the amount of slots displayed.
Entity Id	zigzag64	EntityUniqueID is the unique ID of the entity whose equipment was 'updated' to the player. It is typically the horse entity that had its inventory opened.
Inventory	nbt	inventory is a network NBT serialised compound holding the content of the inventory of the entity (the equipment) and additional data such as the allowed items for a particular slot, used to make sure only saddles can be put in the saddle slot etc.

82

Packet Resource Pack Data Info

Clientbound

ResourcePackDataInfo is sent by the server to the client to inform the client about the data contained in one of the resource packs that are about to be sent.

Field Name	Field Type	Notes
Pack Id	string	UUID is the unique ID of the resource pack that the info concerns.
Max Chunk Size	lu32	DataChunkSize is the maximum size in bytes of the chunks in which the total size of the resource pack to be sent will be divided. A size of 1MB (1024*1024) means that a resource pack of 15.5MB will be split into 16 data chunks.
Chunk Count	lu32	ChunkCount is the total amount of data chunks that the sent resource pack will exist out of. It is the total size of the resource pack divided by the DataChunkSize field. The client doesn't actually seem to use this field. Rather, it divides the size by the chunk size to calculate it itself.
Size	lu64	Size is the total size in bytes that the resource pack occupies. This is the size of the compressed archive (zip) of the resource pack.

Field Name	Field Type	Notes
Hash	ByteArray	Hash is a SHA256 hash of the content of the resource pack.
Is Premium	bool	Premium specifies if the resource pack was a premium resource pack, meaning it was bought from the Minecraft store.
Pack Type	u8 enum	
	1 Addon	PackType is the type of the resource pack. It is one of the resource pack types listed.
	2 Cached	
	3 Copy Protected	
	4 Behavior	
	5 Persona Piece	
	6 Resources	
	7 Skins	
	8 World Template	

83

Clientbound

Packet Resource Pack Chunk Data

ResourcePackChunkData is sent to the client so that the client can download the resource pack. Each packet holds a chunk of the compressed resource pack, of which the size is defined in the ResourcePackDataInfo packet sent before.

Field Name	Field Type	Notes
Pack Id	string	UUID is the unique ID of the resource pack that the chunk of data is taken out of.
Chunk Index	lu32	ChunkIndex is the current chunk index of the chunk. It is a number that starts at 0 and is incremented for each resource pack data chunk sent to the client.

Field Name	Field Type	Notes
Progress	lu64	DataOffset is the current progress in bytes or offset in the data that the resource pack data chunk is taken from.
Payload	ByteArray	RawPayload is a byte slice containing a chunk of data from the resource pack. It must be of the same size or less than the DataChunkSize set in the ResourcePackDataInfo packet.

84 Serverbound

Packet Resource Pack Chunk Request

ResourcePackChunkRequest is sent by the client to request a chunk of data from a particular resource pack, that it has obtained information about in a ResourcePackDataInfo packet.

Field Name	Field Type	Notes
Pack Id	string	UUID is the unique ID of the resource pack that the chunk of data is requested from.
Chunk Index	lu32	ChunkIndex is the requested chunk index of the chunk. It is a number that starts at 0 and is incremented for each resource pack data chunk requested.

85 Clientbound

Packet Transfer

Field Name	Field Type	Notes
Server Address	string	
Port	lu16	
Reload World	bool	

86 Clientbound

Packet Play Sound

Field Name	Field Type	Notes
Name	string	
Coordinates	BlockCoordinates	
Volume	lf32	
Pitch	lf32	

87 Packet Stop Sound

Clientbound

Field Name	Field Type	Notes
Name	string	
Stop All	bool	
Stop Music Legacy	bool	

88 Packet Set Title

Clientbound

SetTitle is sent by the server to make a title, subtitle or action bar shown to a player. It has several fields that allow setting the duration of the titles.

<table>
<tr><th>Field Name</th><th colspan="2">Field Type</th><th>Notes</th></tr>
<tr><td rowspan="7">Type</td><td colspan="3">zigzag32 enum</td></tr>
<tr><td>0</td><td>Clear</td><td rowspan="6">ActionType is the type of the action that should be executed upon the title of a player. It is one of the constants above and specifies the response of the client to the packet.</td></tr>
<tr><td>1</td><td>Reset</td></tr>
<tr><td>2</td><td>Set Title</td></tr>
<tr><td>3</td><td>Set Subtitle</td></tr>
<tr><td>4</td><td>Action Bar Message</td></tr>
<tr><td>5</td><td>Set Durations</td></tr>
</table>

Field Name	Field Type	Notes
	6 Set Title Json 7 Set Subtitle Json 8 Action Bar Message Json	
Text	string	Text is the text of the title, which has a different meaning depending on the ActionType that the packet has. The text is the text of a title, subtitle or action bar, depending on the type set.
Fade In Time	zigzag32	FadeInDuration is the duration that the title takes to fade in on the screen of the player. It is measured in 20ths of a second (AKA in ticks).
Stay Time	zigzag32	RemainDuration is the duration that the title remains on the screen of the player. It is measured in 20ths of a second (AKA in ticks).
Fade Out Time	zigzag32	FadeOutDuration is the duration that the title takes to fade out of the screen of the player. It is measured in 20ths of a second (AKA in ticks).
Xuid	string	XUID is the XBOX Live user ID of the player, which will remain consistent as long as the player is logged in with the XBOX Live account. It is empty if the user is not logged into its XBL account.
Platform Online Id	string	PlatformOnlineID is either a uint64 or an empty string.
Filtered Message	string	FilteredMessage is a filtered version of Message with all the profanity removed. The client will use this over Message if this field is not empty and they have the "Filter Profanity" setting enabled.

89

Packet Add Behavior Tree

Clientbound

Field Name	Field Type	Notes
Behaviortree	string	

Packet Structure Block Update

StructureBlockUpdate is sent by the client when it updates a structure block using the in-game UI. The data it contains depends on the type of structure block that it is. In Minecraft Bedrock Edition v1.11, there is only the Export structure block type, but in v1.13 the ones present in Java Edition will, according to the wiki, be added too.

Field Name	Field Type	Notes
Position	BlockCoordinates	Position is the position of the structure block that is updated.
Structure Name	string	StructureName is the name of the structure that was set in the structure block's UI. This is the name used to export the structure to a file.
Filtered Structure Name	string	FilteredStructureName is a filtered version of StructureName with all the profanity removed. The client will use this over StructureName if this field is not empty and they have the "Filter Profanity" setting enabled.
Data Field	string	DataField is the name of a function to run, usually used during natural generation. A description can be found here: https://minecraft.wiki/w/Structure_Block#Data.
Include Players	bool	IncludePlayers specifies if the 'Include Players' toggle has been enabled, meaning players are also exported by the structure block.
Show Bounding Box	bool	ShowBoundingBox specifies if the structure block should have its bounds outlined. A thin line will encapsulate the bounds of the structure if set to true.
Structure Block Type	zigzag32	StructureBlockType is the type of the structure block updated. A list of structure block types that will be used can be found in the constants above.
Settings	StructureBlockSettings	Settings is a struct of settings that should be used for exporting the structure. These settings are identical to the last sent in the StructureBlockUpdate packet by the client.

Field Name	Field Type	Notes
Redstone Save Mode	zigzag32	RedstoneSaveMode is the mode that should be used to save the structure when used with redstone. In Java Edition, this is always stored in memory, but in Bedrock Edition it can be stored either to disk or memory. See the constants above for the options.
Should Trigger	bool	ShouldTrigger specifies if the structure block should be triggered immediately after this packet reaches the server.
Water Logged	bool	Waterlogged specifies if the structure block is waterlogged at the time of the packet being sent.

91

Packet Show Store Offer

Clientbound

ShowStoreOffer is sent by the server to show a Marketplace store offer to a player. It opens a window client-side that displays the item. The ShowStoreOffer packet only works on the partnered servers: Servers that are not partnered will not have a store buttons show up in the in-game pause menu and will, as a result, not be able to open store offers on the client side. Sending the packet does therefore not work when using a proxy that is not connected to with the domain of one of the partnered servers.

Field Name	Field Type		Notes
Offer Id	string		OfferID is a string that identifies the offer for which a window should be opened. While typically a UUID, the ID could be anything.
Redirect Type	u8 enum		
	0	Marketplace	ShowAll specifies if all other offers of the same 'author' as the one of the offer associated with the OfferID should also be displayed, alongside the target offer.
	1	Dressing Room	
	2	Third Party Server Page	

92

Packet Purchase Receipt

Serverbound

PurchaseReceipt is sent by the client to the server to notify the server it purchased an item from the Marketplace store that was offered by the server. The packet is only used for partnered servers.

Field Name	Field Type	Notes
Receipts length	varint	Receipts is a list of receipts, or proofs of purchases, for the offers that have been purchased by the player.
Receipts array	string	

93

Packet Player Skin

Bidirectional

Field Name	Field Type	Notes
Uuid	uuid	
Skin	Skin	
Skin Name	string	
Old Skin Name	string	
Is Verified	bool	

94

Packet Sub Client Login

Serverbound

SubClientLogin is sent when a sub-client joins the server while another client is already connected to it. The packet is sent as a result of split-screen game play, and allows up to four players to play using the same network connection. After an initial Login packet from the 'main' client, each sub-client that connects sends a SubClientLogin to request their own login.

Field Name	Field Type	Notes
Tokens	["encapsulated", { "lengthType": "varint", "type": "LoginTokens" }]	ConnectionRequest is a string containing information about the player and JWTs that may be used to verify if the player is connected to XBOX Live. The connection request also contains the necessary client public key to

Field Name	Field Type	Notes
		initiate encryption. The ConnectionRequest in this packet is identical to the one found in the Login packet.

95 Clientbound

Packet Initiate Web Socket Connection

AutomationClientConnect is used to make the client connect to a websocket server. This websocket server has the ability to execute commands on the behalf of the client and it can listen for certain events fired by the client.

Field Name	Field Type	Notes
Server	string	ServerURI is the URI to make the client connect to. It can be, for example, 'localhost:8000/ws' to connect to a websocket server on the localhost at port 8000.

96 Clientbound

Packet Set Last Hurt By

SetLastHurtBy is sent by the server to let the client know what entity type it was last hurt by. At this moment, the packet is useless and should not be used. There is no behaviour that depends on if this packet is sent or not.

Field Name	Field Type	Notes
Entity Type	varint	

97 Serverbound

Packet Book Edit

BookEdit is sent by the client when it edits a book. It is sent each time a modification was made and the player stops its typing 'session', rather than simply after closing the book.

Field Name	Field Type	Notes
Type	u8 enum	

Field Name	Field Type			Notes
	0	Replace Page		
	1	Add Page		
	2	Delete Page		
	3	Swap Pages		
	4	Sign		
Slot	u8			
🔁 *if Type*	*Is Replace Page Or Add Page*	**Page Number**	u8	
		Text	string	
		Photo Name	string	Only available on Education Edition.
	Is Delete Page	**Page Number**	u8	
	Is Swap Pages	**Page1**	u8	
		Page2	u8	
	Is Sign	**Title**	string	
		Author	string	
		Xuid	string	

98

Packet Npc Request

Bidirectional

NPCRequest is sent by the client when it interacts with an NPC. The packet is specifically made for Education Edition, where NPCs are available to use.

Field Name	Field Type		Notes
Runtime Entity Id	varint64		EntityRuntimeID is the runtime ID of the NPC entity that the player interacted with. It is the same as sent by the server when spawning the entity.
Request Type	u8 enum		
	0	Set Actions	RequestType is the type of the request, which depends on the permission that the player has. It will be either a type that

Field Name	Field Type		Notes
			indicates that the NPC should show its dialog, or that it should open the editing window.
	1	Execute Action	
	2	Execute Closing Commands	
	3	Set Name	
	4	Set Skin	
	5	Set Interaction Text	
	6	Execute Opening Commands	
Command	string		CommandString is the command string set in the NPC. It may consist of multiple commands, depending on what the player set in it.
Action Type	u8 enum		
	0	Set Actions	ActionType is the type of the action to execute.
	1	Execute Action	
	2	Execute Closing Commands	
	3	Set Name	
	4	Set Skin	
	5	Set Interact Text	
	6	Execute Opening Commands	
Scene Name	string		SceneName is the name of the scene.

Serverbound

99 Packet Photo Transfer

PhotoTransfer is sent by the server to transfer a photo (image) file to the client. It is typically used to transfer photos so that the client can display it in a portfolio in Education Edition. While previously usable in the default Bedrock Edition, the displaying of photos in books was disabled and the packet now has little use anymore.

Field Name	Field Type	Notes
Image Name	string	PhotoName is the name of the photo to transfer. It is the exact file name that the client will download the photo as, including the extension of the file.
Image Data	string	PhotoData is the raw data of the photo image. The format of this data may vary: Formats such as JPEG or PNG work, as long as PhotoName has the correct extension.
Book Id	string	BookID is the ID of the book that the photo is associated with. If the PhotoName in a book with this ID is set to PhotoName, it will display the photo (provided Education Edition is used). The photo image is downloaded to a sub-folder with this book ID.
Photo Type	u8	PhotoType is one of the three photo types above.
Source Type	u8	SourceType is the source photo type. It is one of the three photo types above.
Owner Entity Unique Id	li64	OwnerEntityUniqueID is the entity unique ID of the photo's owner.
New Photo Name	string	NewPhotoName is the new name of the photo.

Clientbound

100 Packet Modal Form Request

ModalFormRequest is sent by the server to make the client open a form. This form may be either a modal form which has two options, a menu form for a selection of options and a custom form for properties.

Field Name	Field Type	Notes
Form Id	varint	FormID is an ID used to identify the form. The ID is saved by the client and sent back when the player submits the form, so that the server can identify which form was submitted.
Data	string	FormData is a JSON encoded object of form data. The content of the object differs, depending on the type of the form sent, which is also set in the JSON.

101

Packet Modal Form Response

Serverbound

ModalFormResponse is sent by the client in response to a ModalFormRequest, after the player has submitted the form sent. It contains the options/properties selected by the player, or a JSON encoded 'null' if the form was closed by clicking the X at the top right corner of the form.

Field Name	Field Type			Notes
Form Id	varint			FormID is the form ID of the form the client has responded to. It is the same as the ID sent in the ModalFormRequest, and may be used to identify which form was submitted.
Has Response Data	bool			HasResponseData is true if the client provided response data.
Data *if Has Response Data*	*Is True*	string		ResponseData is a JSON encoded value representing the response of the player. For a modal form, the response is either true or false, for a menu form, the response is an integer specifying the index of the button clicked, and for a custom form, the response is an array containing a value for each element.
Has Cancel Reason	bool			HasCancelReason is true if the client provided a reason for the form being cancelled.
🔁 *if Has Cancel Reason*	*Is True*	**Cancel Reason**	u8 enum	
			0 Closed	CancelReason represents the reason why the form was cancelled.
			1 Busy	

102 Packet Server Settings Request

Serverbound

ServerSettingsRequest is sent by the client to request the settings specific to the server. These settings are shown in a separate tab client-side, and have the same structure as a custom form. ServerSettingsRequest has no fields.

Field Name	Field Type	Notes

103 Packet Server Settings Response

Clientbound

ServerSettingsResponse is optionally sent by the server in response to a ServerSettingsRequest from the client. It is structured the same as a ModalFormRequest packet, and if filled out correctly, will show a specific tab for the server in the settings of the client. A ModalFormResponse packet is sent by the client in response to a ServerSettingsResponse, when the client fills out the settings and closes the settings again.

Field Name	Field Type	Notes
Form Id	varint	FormID is an ID used to identify the form. The ID is saved by the client and sent back when the player submits the form, so that the server can identify which form was submitted.
Data	string	FormData is a JSON encoded object of form data. The content of the object differs, depending on the type of the form sent, which is also set in the JSON.

104 Packet Show Profile

Clientbound

ShowProfile is sent by the server to show the XBOX Live profile of one player to another.

Field Name	Field Type	Notes
Xuid	string	XUID is the XBOX Live User ID of the player whose profile should be shown to the player. If it is not a valid XUID, the client ignores the packet.

Serverbound

105 Packet Set Default Game Type

SetDefaultGameType is sent by the client when it toggles the default game type in the settings UI, and is sent by the server when it actually changes the default game type, resulting in the toggle being changed in the settings UI.

Field Name	Field Type	Notes
Gamemode	GameMode	GameType is the new game type that is set. When sent by the client, this is the requested new default game type.

Clientbound

106 Packet Remove Objective

RemoveObjective is sent by the server to remove a scoreboard objective. It is used to stop showing a scoreboard to a player.

Field Name	Field Type	Notes
Objective Name	string	ObjectiveName is the name of the objective that the scoreboard currently active has. This name must be identical to the one sent in the SetDisplayObjective packet.

Clientbound

107 Packet Set Display Objective

SetDisplayObjective is sent by the server to display an object as a scoreboard to the player. Once sent, it should be followed up by a SetScore packet to set the lines of the packet.

Field Name	Field Type	Notes
Display Slot	string	DisplaySlot is the slot in which the scoreboard should be displayed. Available options can be found in the constants above.
Objective Name	string	ObjectiveName is the name of the objective that the scoreboard displays. Filling out a random unique value for this field works: It is not displayed in the scoreboard.

Field Name	Field Type	Notes
Display Name	string	DisplayName is the name, or title, that is displayed at the top of the scoreboard.
Criteria Name	string	CriteriaName is the name of the criteria that need to be fulfilled in order for the score to be increased. This can be any kind of string and does not show up client-side.
Sort Order	zigzag32	SortOrder is the order in which entries on the scoreboard should be sorted. It is one of the constants that may be found above.

108 Packet Set Score

Clientbound

SetScore is sent by the server to send the contents of a scoreboard to the player. It may be used to either add, remove or edit entries on the scoreboard.

Field Name	Field Type			Notes
Action	u8 enum			
	0	Change		ActionType is the type of the action to execute upon the scoreboard with the entries that the packet has. If ActionType is ScoreboardActionModify, all entries will be added to the scoreboard if not yet present, or modified if already present. If set to ScoreboardActionRemove, all scoreboard entries set will be removed from the scoreboard.
	1	Remove		
Entries length	varint			
Entries array	**Scoreboard Id**	zigzag64		
	Objective Name	string		
	Score	li32		
	🔁 *if ../Action*	*Is Change*	**Entry Type**	i8 enum
				1 Player

Field Name	Field Type			Notes	
				2	Entity
				3	Fake Player
			Entity Unique Id *if Entry Type*	*Is Player Or Entity*	zigzag64
			Custom Name *if Entry Type*	*Is Fake Player*	string

109 Bidirectional

Packet Lab Table

LabTable is sent by the client to let the server know it started a chemical reaction in Education Edition, and is sent by the server to other clients to show the effects. The packet is only functional if Education features are enabled.

Field Name	Field Type		Notes
Action Type	u8 enum		
	0	Combine	ActionType is the type of the action that was executed. It is one of the constants above. Typically, only LabTableActionCombine is sent by the client, whereas LabTableActionReact is sent by the server.
	1	React	
	2	Reset	
Position	vec3i		Position is the position at which the lab table used was located.
Reaction Type	u8		ReactionType is the type of the reaction that took place as a result of the items put into the lab table. The reaction type can be either that of an item or a particle, depending on whatever the result was of the reaction.

Clientbound

110 Packet Update Block Synced

UpdateBlockSynced is sent by the server to synchronise the falling of a falling block entity with the transitioning back and forth from and to a solid block. It is used to prevent the entity from flickering, and is used in places such as the pushing of blocks with pistons.

Field Name	Field Type	Notes
Position	BlockCoordinates	Position is the block position at which a block is updated.
Block Runtime Id	varint	NewBlockRuntimeID is the runtime ID of the block that is placed at Position after sending the packet to the client.
Flags	UpdateBlockFlags	Flags is a combination of flags that specify the way the block is updated client-side. It is a combination of the flags above, but typically sending only the BlockUpdateNetwork flag is sufficient.
Layer	varint	Layer is the world layer on which the block is updated. For most blocks, this is the first layer, as that layer is the default layer to place blocks on, but for blocks inside of each other, this differs.
Entity Unique Id	zigzag64	EntityUniqueID is the unique ID of the falling block entity that the block transitions to or that the entity transitions from. Note that for both possible values for TransitionType, the EntityUniqueID should point to the falling block entity involved.
Transition Type	varint enum	

Clientbound

111 Packet Move Entity Delta

TransitionType is the type of the transition that happened. It is either BlockToEntityTransition, when a block placed becomes a falling entity, or EntityToBlockTransition, when a falling entity hits the ground and becomes a solid block again. MoveActorDelta is sent by the server to move an entity. The packet is specifically optimised to save as much space as possible, by only writing non-zero fields. As of 1.16.100, this packet no longer actually contains any deltas.

Field Name	Field Type	Notes
Runtime Entity Id	varint64	EntityRuntimeID is the runtime ID of the entity that is being moved. The packet works provided a non-player entity with this runtime ID is present.
Flags	DeltaMoveFlags	Flags is a list of flags that specify what data is in the packet.
X *if Flags.Has X*	*Is True*	lf32
Y *if Flags.Has Y*	*Is True*	lf32
Z *if Flags.Has Z*	*Is True*	lf32
Rot X *if Flags.Has Rot X*	*Is True*	u8
Rot Y *if Flags.Has Rot Y*	*Is True*	u8
Rot Z *if Flags.Has Rot Z*	*Is True*	u8

112

Packet Set Scoreboard Identity

Clientbound

SetScoreboardIdentity is sent by the server to change the identity type of one of the entries on a scoreboard. This is used to change, for example, an entry pointing to a

player, to a fake player when it leaves the server, and to change it back to a real player when it joins again. In non-vanilla situations, the packet is quite useless.

Field Name	Field Type		Notes
Action	i8 enum		
	0	Register Identity	ActionType is the type of the action to execute. The action is either ScoreboardIdentityActionRegister to associate an identity with the entry, or ScoreboardIdentityActionClear to remove associations with an entity.
	1	Clear Identity	
Entries length	varint		Entries is a list of all entries in the packet. Each of these entries points to one of the entries on a scoreboard. Depending on ActionType, their identity will either be registered or cleared.
Entries array	**Scoreboard Id**		zigzag64
	Entity Unique Id		
	if ../Action	*Is Register Identity*	zigzag64
		Default	void

113 Serverbound

Packet Set Local Player As Initialized

SetLocalPlayerAsInitialised is sent by the client in response to a PlayStatus packet with the status set to spawn. The packet marks the moment at which the client is fully initialised and can receive any packet without discarding it.

Field Name	Field Type	Notes
Runtime Entity Id	varint64	EntityRuntimeID is the entity runtime ID the player was assigned earlier in the login sequence in the StartGame packet.

114 Clientbound

Packet Update Soft Enum

UpdateSoftEnum is sent by the server to update a soft enum, also known as a dynamic enum, previously sent in the AvailableCommands packet. It is sent whenever the enum should get new options or when some of its options should be removed. The UpdateSoftEnum packet will apply for enums that have been set in the AvailableCommands packet with the 'Dynamic' field of the CommandEnum set to true.

Field Name	Field Type	Notes
Enum Type	string	EnumType is the type of the enum. This type must be identical to the one set in the AvailableCommands packet, because the client uses this to recognise which enum to update.
Options length	varint	Options is a list of options that should be updated. Depending on the ActionType field, either these options will be added to the enum, the enum options will be set to these options or all of these options will be removed from the enum.
Options array	string	
Action Type	u8 enum	
	0 Add	ActionType is the type of the action to execute on the enum. The Options field has a different result, depending on what ActionType is used.
	1 Remove	
	2 Update	

115 **Packet Network Stack Latency** Bidirectional

NetworkStackLatency is sent by the server (and the client, on development builds) to measure the latency over the entire Minecraft stack, rather than the RakNet latency. It has other usages too, such as the ability to be used as some kind of acknowledgement packet, to know when the client has received a certain other packet.

Field Name	Field Type	Notes
Timestamp	lu64	Timestamp is the timestamp of the network stack latency packet. The client will, if NeedsResponse is set to true, send a NetworkStackLatency packet with this same timestamp packet in response.
Needs Response	u8	NeedsResponse specifies if the sending side of this packet wants a response to the packet, meaning that the

Field Name	Field Type	Notes
		other side should send a NetworkStackLatency packet back.

117

Packet Script Custom Event

Bidirectional

ScriptCustomEvent is sent by both the client and the server. It is a way to let scripts communicate with the server, so that the client can let the server know it triggered an event, or the other way around. It is essentially an RPC kind of system. Deprecated: ScriptCustomEvent is deprecated as of 1.20.10.

Field Name	Field Type	Notes
Event Name	string	EventName is the name of the event. The script and the server will use this event name to identify the data that is sent.
Event Data	string	EventData is the data of the event. This data is typically a JSON encoded string, that the script is able to encode and decode too.

118

Packet Spawn Particle Effect

Clientbound

SpawnParticleEffect is sent by the server to spawn a particle effect client-side. Unlike other packets that result in the appearing of particles, this packet can show particles that are not hardcoded in the client. They can be added and changed through behaviour packs to implement custom particles.

Field Name	Field Type	Notes
Dimension	u8	Dimension is the dimension that the particle is spawned in. Its exact usage is not clear, as the dimension has no direct effect on the particle.
Entity Id	zigzag64	EntityUniqueID is the unique ID of the entity that the spawned particle may be attached to. If this ID is not -1, the Position below will be interpreted as relative to the position of the entity associated with this unique ID.
Position	vec3f	Position is the position that the particle should be spawned at. If the position is too far away from the

Field Name	Field Type	Notes
		player, it will not show up. If EntityUniqueID is not -1, the position will be relative to the position of the entity.
Particle Name	string	ParticleName is the name of the particle that should be shown. This name may point to a particle effect that is built-in, or to one implemented by behaviour packs.
Molang Variables **Optional**	string	MoLangVariables is an encoded JSON map of MoLang variables that may be applicable to the particle spawn. This can just be left empty in most cases.

119 Packet Available Entity Identifiers

Clientbound

AvailableActorIdentifiers is sent by the server at the start of the game to let the client know all entities that are available on the server.

Field Name	Field Type	Notes
Nbt	nbt	SerialisedEntityIdentifiers is a network NBT serialised compound of all entity identifiers that are available in the server.

120 Packet Level Sound Event V2

Bidirectional

Not used. Use packet_level_sound_event.

Field Name	Field Type	Notes
Sound Id	u8	
Position	vec3f	
Block Id	zigzag32	
Entity Type	string	
Is Baby Mob	bool	
Is Global	bool	

Packet Network Chunk Publisher Update

NetworkChunkPublisherUpdate is sent by the server to change the point around which chunks are and remain loaded. This is useful for mini-game servers, where only one area is ever loaded, in which case the NetworkChunkPublisherUpdate packet can be sent in the middle of it, so that no chunks ever need to be additionally sent during the course of the game. In reality, the packet is not extraordinarily useful, and most servers just send it constantly at the position of the player. If the packet is not sent at all, no chunks will be shown to the player, regardless of where they are sent.

Field Name	Field Type		Notes
Coordinates	BlockCoordinates		Position is the block position around which chunks loaded will remain shown to the client. Most servers set this position to the position of the player itself.
Radius	varint		Radius is the radius in blocks around Position that chunks sent show up in and will remain loaded in. Unlike the RequestChunkRadius and ChunkRadiusUpdated packets, this radius is in blocks rather than chunks, so the chunk radius needs to be multiplied by 16. (Or shifted to the left by 4.)
Saved Chunks length	lu32		
Saved Chunks array	**X**	zigzag32	ChunkX is the X coordinate of the chunk sent. (To translate a block's X to a chunk's X: x >> 4)
	Z	zigzag32	ChunkZ is the Z coordinate of the chunk sent. (To translate a block's Z to a chunk's Z: z >> 4)

Packet Biome Definition List

BiomeDefinitionList is sent by the server to let the client know all biomes that are available and implemented on the server side. It is much like the AvailableActorIdentifiers packet, but instead functions for biomes.

Field Name	Field Type	Notes
Biome Definitions length	varint	BiomeDefinitions is a list of biomes that are available on the server.
Biome Definitions array	BiomeDefinition	
String List length	varint	StringList is a makeshift dictionary implementation Mojang created to try and reduce the size of the overall packet. It is a list of common strings that are used in the biome definitions.
String List array	string	

123

Packet Level Sound Event

Bidirectional

LevelSoundEvent is sent by the server to make any kind of built-in sound heard to a player. It is sent to, for example, play a stepping sound or a shear sound. The packet is also sent by the client, in which case it could be forwarded by the server to the other players online. If possible, the packets from the client should be ignored however, and the server should play them on its own accord.

Field Name	Field Type	Notes
Sound Id	SoundType	SoundType is the type of the sound to play. Some of the sound types require additional data, which is set in the EventData field.
Position	vec3f	Position is the position of the sound event. The player will be able to hear the direction of the sound based on what position is sent here.
Extra Data	zigzag32	ExtraData is a packed integer that some sound types use to provide extra data. An example of this is the note sound, which is composed of a pitch and an instrument type.
Entity Type	string	EntityType is the string entity type of the entity that emitted the sound, for example 'minecraft:skeleton'. Some sound types use this entity type for additional data.
Is Baby Mob	bool	BabyMob specifies if the sound should be that of a baby mob. It is most notably used for parrot imitations, which will change based on if this field is set to true or not.

Field Name	Field Type	Notes
Is Global	bool	DisableRelativeVolume specifies if the sound should be played relatively or not. If set to true, the sound will have full volume, regardless of where the Position is, whereas if set to false, the sound's volume will be based on the distance to Position.
Entity Unique Id	li64	EntityUniqueID is the unique ID of a source entity. The unique ID is a value that remains consistent across different sessions of the same world, but most servers simply fill the runtime ID of the entity out for this field.

124 Clientbound

Packet Level Event Generic

LevelEventGeneric is sent by the server to send a 'generic' level event to the client. This packet sends an NBT serialised object and may for that reason be used for any event holding additional data.

Field Name	Field Type	Notes
Event Id	varint	EventID is a unique identifier that identifies the event called. The data that follows has fields in the NBT depending on what event it is.
Nbt	nbtLoop	SerialisedEventData is a network little endian serialised object of event data, with fields that vary depending on EventID. Unlike many other NBT structures, this data is not actually in a compound but just loosely floating NBT tags. To decode using the nbt package, you would need to append 0x0a00 at the start (compound id and name length) and add 0x00 at the end, to manually wrap it in a compound. Likewise, you would have to remove these bytes when encoding.

125 Serverbound

Packet Lectern Update

LecternUpdate is sent by the client to update the server on which page was opened in a book on a lectern, or if the book should be removed from it.

Field Name	Field Type	Notes
Page	u8	Page is the page number in the book that was opened by the player on the lectern.
Page Count	u8	PageCount is the number of pages that the book opened in the lectern has.
Position	vec3i	Position is the position of the lectern that was updated. If no lectern is at the block position, the packet should be ignored.

126 Packet Video Stream Connect

Clientbound

This packet was removed.

Field Name	Field Type	Notes
Server Uri	string	
Frame Send Frequency	lf32	
Action	u8 enum	
	1 None	
	2 Close	
Resolution X	li32	
Resolution Y	li32	

129 Packet Client Cache Status

Bidirectional

ClientCacheStatus is sent by the client to the server at the start of the game. It is sent to let the server know if it supports the client-side blob cache. Clients such as Nintendo Switch do not support the cache, and attempting to use it anyway will fail.

Field Name	Field Type	Notes
Enabled	bool	Enabled specifies if the blob cache is enabled. If false, the server should not attempt to use the blob cache. If true, it may do so, but it may also choose not to use it.

130 Clientbound

Packet On Screen Texture Animation

OnScreenTextureAnimation is sent by the server to show a certain animation on the screen of the player. The packet is used, as an example, for when a raid is triggered and when a raid is defeated.

Field Name	Field Type	Notes
Animation Type	lu32	AnimationType is the type of the animation to show. The packet provides no further extra data to allow modifying the duration or other properties of the animation.

131 Clientbound

Packet Map Create Locked Copy

MapCreateLockedCopy is sent by the server to create a locked copy of one map into another map. In vanilla, it is used in the cartography table to create a map that is locked and cannot be modified.

Field Name	Field Type	Notes
Original Map Id	zigzag64	OriginalMapID is the ID of the map that is being copied. The locked copy will obtain all content that is visible on this map, except the content will not change.
New Map Id	zigzag64	NewMapID is the ID of the map that holds the locked copy of the map that OriginalMapID points to. Its contents will be impossible to change.

132 Serverbound

Packet Structure Template Data Export Request

StructureTemplateDataRequest is sent by the client to request data of a structure.

Field Name	Field Type	Notes
Name	string	StructureName is the name of the structure that was set in the structure block's UI. This is the name used to export the structure to a file.
Position	BlockCoordinates	Position is the position of the structure block that has its template data requested.
Settings	StructureBlockSettings	Settings is a struct of settings that should be used for exporting the structure. These settings are identical to the last sent in the StructureBlockUpdate packet by the client.
Request Type	u8 enum	
	1 Export From Save	RequestType specifies the type of template data request that the player sent.
	2 Export From Load	
	3 Query Saved Structure	
	4 Import From Save	

133

Clientbound

Packet Structure Template Data Export Response

StructureTemplateDataResponse is sent by the server to send data of a structure to the client in response to a StructureTemplateDataRequest packet.

Field Name	Field Type	Notes
Name	string	
Success	bool	
Nbt		
if Success	*Is True*	nbt

Field Name	Field Type		Notes
Response Type	u8 enum		
	1	Export	ResponseType specifies the response type of the packet. This depends on the RequestType field sent in the StructureTemplateDataRequest packet and is one of the constants above.
	2	Query	
	3	Import	

134 Packet Update Block Properties

Clientbound

No longer used.

Field Name	Field Type	Notes
Nbt	nbt	

135 Packet Client Cache Blob Status

Serverbound

ClientCacheBlobStatus is part of the blob cache protocol. It is sent by the client to let the server know what blobs it needs and which blobs it already has, in an ACK type system.

Field Name	Field Type	Notes
Misses	varint	The number of MISSes in this packet
Haves	varint	The number of HITs in this packet
*Length for Missing below is **Misses** from above*		A list of blob hashes that the client does not have a blob available for. The server should send the blobs matching these hashes as soon as possible.
Missing array	lu64	
*Length for Have below is **Haves** from above*		A list of hashes that the client does have a cached blob for. Server doesn't need to send.
Have array	lu64	

136 Clientbound

Packet Client Cache Miss Response

ClientCacheMissResponse is part of the blob cache protocol. It is sent by the server in response to a ClientCacheBlobStatus packet and contains the blob data of all blobs that the client acknowledged not to have yet.

Field Name	Field Type	Notes
Blobs length	varint	
Blobs array	Blob	

137 Clientbound

Packet Education Settings

EducationSettings is a packet sent by the server to update Minecraft: Education Edition related settings. It is unused by the normal base game.

Field Name	Field Type	Notes
CodeBuilderDefaultURI	string	CodeBuilderDefaultURI is the default URI that the code builder is ran on. Using this, a Code Builder program can make code directly affect the server.
CodeBuilderTitle	string	CodeBuilderTitle is the title of the code builder shown when connected to the CodeBuilderDefaultURI.
CanResizeCodeBuilder	bool	CanResizeCodeBuilder specifies if clients connected to the world should be able to resize the code builder when it is opened.
Disable Legacy Title Bar	bool	
Post Process Filter	string	
Screenshot Border Path	string	
Has Agent Capabilities	bool	
Agent Capabilities	*Is True*	**Has** bool

Field Name	Field Type	Notes
if Has Agent Capabilities		**Can Modify Blocks** bool
HasOverrideURI	bool	
OverrideURI *if HasOverrideURI*	*Is True*	string
HasQuiz	bool	HasQuiz specifies if the world has a quiz connected to it.
Has External Link Settings	bool	
External Link Settings *if Has External Link Settings*	*Is True*	**Has** bool **Url** string **Display Name** string

138

Packet Emote

Bidirectional

Emote is sent by both the server and the client. When the client sends an emote, it sends this packet to the server, after which the server will broadcast the packet to other players online.

Field Name	Field Type	Notes
Entity Id	varint64	EntityRuntimeID is the entity that sent the emote. When a player sends this packet, it has this field set as its own entity runtime ID.
Emote Id	string	EmoteID is the ID of the emote to send.
Emote Length Ticks	varint	EmoteLength is the number of ticks that the emote lasts for.
Xuid	string	XUID is the Xbox User ID of the player that sent the emote. It is only set when the emote is used by a player that is authenticated with Xbox Live.
Platform Id	string	PlatformID is an identifier only set for particular platforms when using an emote (presumably only for Nintendo Switch). It is otherwise an empty string, and is

Field Name	Field Type		Notes
			used to decide which players are able to emote with each other.
Flags	u8 enum		
	1	Server Side	Flags is a combination of flags that change the way the Emote packet operates. When the server sends this packet to other players, EmoteFlagServerSide must be present.
	2	Mute Chat	

139 Serverbound

Packet Multiplayer Settings

MultiPlayerSettings is sent by the client to update multi-player related settings server-side and sent back to online players by the server. The MultiPlayerSettings packet is a Minecraft: Education Edition packet. It has no functionality for the base game.

Field Name	Field Type		Notes
Action Type	zigzag32 enum		
	0	Enable Multiplayer	ActionType is the action that should be done when this packet is sent. It is one of the constants that may be found above.
	1	Disable Multiplayer	
	2	Refresh Join Code	

140 Serverbound

Packet Settings Command

SettingsCommand is sent by the client when it changes a setting in the settings that results in the issuing of a command to the server, such as when Show Coordinates is enabled.

Field Name	Field Type	Notes
Command Line	string	CommandLine is the full command line that was sent to the server as a result of the setting that the client changed.
Suppress Output	bool	SuppressOutput specifies if the client requests the suppressing of the output of the command that was executed. Generally this is set to true, as the client won't need a message to confirm the output of the change.

141

Packet Anvil Damage

Serverbound

AnvilDamage is sent by the client to request the dealing damage to an anvil. This packet is completely pointless and the server should never listen to it.

Field Name	Field Type	Notes
Damage	u8	Damage is the damage that the client requests to be dealt to the anvil.
Position	BlockCoordinates	AnvilPosition is the position in the world that the anvil can be found at.

142

Packet Completed Using Item

Clientbound

CompletedUsingItem is sent by the server to tell the client that it should be done using the item it is currently using.

Field Name	Field Type		Notes
Used Item Id	li16		UsedItemID is the item ID of the item that the client completed using. This should typically be the ID of the item held in the hand.
Use Method	li32 enum		
	0	Equip Armor	UseMethod is the method of the using of the item that was completed. It is one of the constants that may be found above.
	1	Eat	

Field Name	Field Type	Notes
	2 Attack 3 Consume 4 Throw 5 Shoot 6 Place 7 Fill Bottle 8 Fill Bucket 9 Pour Bucket 10 Use Tool 11 Interact 12 Retrieved 13 Dyed 14 Traded 15 Brushing Completed 16 Opened Vault	

143 **Packet Network Settings** Clientbound

NetworkSettings is sent by the server to update a variety of network settings. These settings modify the way packets are sent over the network stack.

Field Name	Field Type	Notes
Compression Threshold	lu16	CompressionThreshold is the minimum size of a packet that is compressed when sent. If the size of a packet is under this value, it is not compressed. When set to 0, all packets will be left uncompressed.
Compression Algorithm	lu16 enum	
	0 Deflate	CompressionAlgorithm is the algorithm that is used to compress packets.

Field Name	Field Type	Notes
	1 Snappy	
Client Throttle	bool	ClientThrottle regulates whether the client should throttle players when exceeding of the threshold. Players outside threshold will not be ticked, improving performance on low-end devices.
Client Throttle Threshold	u8	ClientThrottleThreshold is the threshold for client throttling. If the number of players exceeds this value, the client will throttle players.
Client Throttle Scalar	lf32	ClientThrottleScalar is the scalar for client throttling. The scalar is the amount of players that are ticked when throttling is enabled.

144 **Packet Player Auth Input** Serverbound

PlayerAuthInput is sent by the client to allow for server authoritative movement. It is used to synchronise the player input with the position server-side. The client sends this packet when the ServerAuthoritativeMovementMode field in the StartGame packet is set to true, instead of the MovePlayer packet. The client will send this packet once every tick.

Field Name	Field Type	Notes
Pitch	lf32	Pitch that the player reports it has.
Yaw	lf32	Yaw that player reports it has.
Position	vec3f	Position holds the position that the player reports it has.
Move Vector	vec2f	MoveVector is a Vec2 that specifies the direction in which the player moved, as a combination of X/Z values which are created using the WASD/controller stick state.
Head Yaw	lf32	HeadYaw is the horizontal rotation of the head that the player reports it has.
Input Data	InputFlag	InputData is a combination of bit flags that together specify the way the player moved last tick. It is a combination of the flags above.
Input Mode	varint enum	

Field Name	Field Type		Notes
	0	Unknown	InputMode specifies the way that the client inputs data to the screen. It is one of the constants that may be found above.
	1	Mouse	
	2	Touch	
	3	Game Pad	
	4	Motion Controller	
Play Mode	varint enum		
	0	Normal	PlayMode specifies the way that the player is playing. The values it holds, which are rather random, may be found above.
	1	Teaser	
	2	Screen	
	3	Viewer	
	4	Reality	
	5	Placement	
	6	Living Room	
	7	Exit Level	
	8	Exit Level Living Room	
	9	Num Modes	
Interaction Model	zigzag32 enum		
	0	Touch	InteractionModel is a constant representing the interaction model the player is using.
	1	Crosshair	
	2	Classic	
Interact Rotation	vec2f		interact_rotation is the rotation the player is looking that they intend to use for interactions. This is only different to Pitch and Yaw in cases such as VR or when custom cameras being used.

Field Name	Field Type	Notes
Tick	varint64	Tick is the server tick at which the packet was sent. It is used in relation to CorrectPlayerMovePrediction.
Delta	vec3f	Delta was the delta between the old and the new position. There isn't any practical use for this field as it can be calculated by the server itself.
Transaction *if Input Data.Item Interact*	*Is True*	**Legacy** — TransactionLegacy **Actions** — TransactionActions **Data** — TransactionUseItem
Item Stack Request *if Input Data.Item Stack Request*	*Is True*	ItemStackRequest
🔁 *if Input Data.Client Predicted Vehicle*	*Is True*	**Vehicle Rotation** — vec2f **Predicted Vehicle** — zigzag64
Block Action *if Input Data.Block Action*	*Is True* **length**	zigzag32
	Is True array	**Action** — Action 🔁 *if Action*: *Is Start Break Or Abort Break Or Crack Break Or Predict Break Or Continue Break* **Position** — vec3i BlockPosition is the position of the target block, if the action with the ActionType set concerned a block. If that is not the case, the block position will be zero.

Field Name	Field Type	Notes				
				Face	zigzag32	BlockFace is the face of the target block that was touched. If the action with the ActionType set concerned a block. If not, the face is always 0.
Analogue Move Vector	vec2f	AnalogueMoveVector is a Vec2 that specifies the direction in which the player moved, as a combination of X/Z values which are created using an analogue input.				
Camera Orientation	vec3f	CameraOrientation is the vector that represents the camera's forward direction which can be used to transform movement to be camera relative.				
Raw Move Vector	vec2f	RawMoveVector is the value of MoveVector before it is affected by input permissions, sneaking/fly speeds and isn't normalised for analogue inputs.				

145

Packet Creative Content

Clientbound

CreativeContent is a packet sent by the server to set the creative inventory's content for a player. Introduced in 1.16, this packet replaces the previous method - sending an InventoryContent packet with creative inventory window ID. As of v1.16.100, this packet must be sent during the login sequence. Not sending it will stop the client from joining the server.

Field Name	Field Type	Notes		
Groups length	varint	The groups that are displayed within the creative inventory menu.		
Groups array	**Category**	li32 enum		
		0	All	Where the group shows up in the creative inventory (e.g. Construction, Items, etc)

Field Name	Field Type	Notes
		1 Construction 2 Nature 3 Equipment 4 Items 5 Item Command Only
	Name	string — The name of the group (e.g. Decorative stone, Wool, etc.)
	Icon Item	ItemLegacy — The item whose icon is used to label the group (icon you click on to open/close the group)
Items length	varint	Individual items that are displayed within the creative inventory menu, grouped by their category.
Items array	**Entry Id**	varint — The index of the item in the creative menu.
	Item	ItemLegacy
	Group Index	varint — The group index of the item - which group in the groups array this item belongs to.

146

Packet Player Enchant Options

Clientbound

PlayerEnchantOptions is sent by the server to update the enchantment options displayed when the user opens the enchantment table and puts an item in. This packet was added in 1.16 and allows the server to decide on the enchantments that can be selected by the player. The PlayerEnchantOptions packet should be sent once for every slot update of the enchantment table. The vanilla server sends an empty PlayerEnchantOptions packet when the player opens the enchantment table (air is present in the enchantment table slot) and sends the packet with actual enchantments in it when items are put in that can have enchantments.

Field Name	Field Type	Notes
Options length	varint	Options is a list of possible enchantment options for the item that was put into the enchantment

Field Name	Field Type	Notes
Options array	EnchantOption	table.

147 Serverbound

Packet Item Stack Request

ItemStackRequest is sent by the client to change item stacks in an inventory. It is essentially a replacement of the InventoryTransaction packet added in 1.16 for inventory specific actions, such as moving items around or crafting. The InventoryTransaction packet is still used for actions such as placing blocks and interacting with entities.

Field Name	Field Type	Notes
Requests length	varint	
Requests array	ItemStackRequest	

148 Clientbound

Packet Item Stack Response

ItemStackResponse is sent by the server in response to an ItemStackRequest packet from the client. This packet is used to either approve or reject ItemStackRequests from the client. If a request is approved, the client will simply continue as normal. If rejected, the client will undo the actions so that the inventory should be in sync with the server again.

Field Name	Field Type	Notes
Responses	ItemStackResponses	Responses is a list of responses to ItemStackRequests sent by the client before. Responses either approve or reject a request from the client. Vanilla limits the size of this slice to 4096.

149 Clientbound

Packet Player Armor Damage

PlayerArmourDamage is sent by the server to damage the armour of a player. It is a very efficient packet, but generally it's much easier to just send a slot update for the damaged armour.

Field Name	Field Type	Notes
Entries length	varint	
Entries array	ArmorDamageEntry	

Type **ArmorDamageEntry** Datatype

Field Name	Field Type	Notes
Armor Slot	u8 enum	
	0 Helmet	
	1 Chestplate	
	2 Leggings	
	3 Boots	
	4 Body	
Damage	li16	

150 **Packet Code Builder** Clientbound

CodeBuilder is an Education Edition packet sent by the server to the client to open the URL to a Code Builder (websocket) server.

Field Name	Field Type	Notes
Url	string	URL is the url to the Code Builder (websocket) server.
Should Open Code Builder	bool	ShouldOpenCodeBuilder specifies if the client should automatically open the Code Builder app. If set to true, the client will attempt to use the Code Builder app to connect to and interface with the server running at the URL above.

151 **Packet Update Player Game Type** Clientbound

UpdatePlayerGameType is sent by the server to change the game mode of a player. It is functionally identical to the SetPlayerGameType packet.

Field Name	Field Type	Notes
Gamemode	GameMode	GameType is the new game type of the player. It is one of the constants that can be found in set*player*game_type.go. Some of these game types require additional flags to be set in an AdventureSettings packet for the game mode to obtain its full functionality.
Player Unique Id	zigzag64	PlayerUniqueID is the entity unique ID of the player that should have its game mode updated. If this packet is sent to other clients with the player unique ID of another player, nothing happens.
Tick	varint64	

152

Packet Emote List

Serverbound

EmoteList is sent by the client every time it joins the server and when it equips new emotes. It may be used by the server to find out which emotes the client has available. If the player has no emotes equipped, this packet is not sent. Under certain circumstances, this packet is also sent from the server to the client, but I was unable to find when this is done.

Field Name	Field Type	Notes
Player Id	varint64	PlayerRuntimeID is the runtime ID of the player that owns the emote pieces below. If sent by the client, this player runtime ID is always that of the player itself.
Emote Pieces length	varint	EmotePieces is a list of emote pieces that the player with the runtime ID above has.
Emote Pieces array	uuid	

154

Serverbound

Packet Position Tracking Db Request

PositionTrackingDBClientRequest is a packet sent by the client to request the position and dimension of a 'tracking ID'. These IDs are tracked in a database by the server. In 1.16, this is used for lodestones. The client will send this request to find the position a lodestone compass needs to point to. If found, it will point to the lodestone. If not, it will start spinning around. A PositionTrackingDBServerBroadcast packet should be sent in response to this packet.

Field Name	Field Type		Notes
Action	u8 enum		
	0	Query	RequestAction is the action that should be performed upon the receiving of the packet. It is one of the constants found above.
Tracking Id	zigzag32		TrackingID is a unique ID used to identify the request. The server responds with a PositionTrackingDBServerBroadcast packet holding the same ID, so that the client can find out what that packet was in response to.

153

Clientbound

Packet Position Tracking Db Broadcast

PositionTrackingDBServerBroadcast is sent by the server in response to the PositionTrackingDBClientRequest packet. This packet is, as of 1.16, currently only used for lodestones. The server maintains a database with tracking IDs and their position and dimension. The client will request these tracking IDs, (NBT tag set on the lodestone compass with the tracking ID?) and the server will respond with the status of those tracking IDs. What is actually done with the data sent depends on what the client chooses to do with it. For the lodestone compass, it is used to make the compass point towards lodestones and to make it spin if the lodestone at a position is no longer there.

Field Name	Field Type		Notes
Broadcast Action	u8 enum		
	0	Update	BroadcastAction specifies the status of the position tracking DB response. It is one of the constants above, specifying the result of the request with the ID below. The Update action is sent for setting the position of a lodestone compass, the Destroy and NotFound to indicate that there is not (no longer) a lodestone at that position.
	1	Destory	

Field Name	Field Type		Notes
	2	Not Found	
Tracking Id	zigzag32		TrackingID is the ID of the PositionTrackingDBClientRequest packet that this packet was in response to. The tracking ID is also present as the 'id' field in the SerialisedData field.
Nbt	nbt		

155 Packet Debug Info

Clientbound

DebugInfo is a packet sent by the server to the client. It does not seem to do anything when sent to the normal client in 1.16.

Field Name	Field Type	Notes
Player Unique Id	zigzag64	PlayerUniqueID is the unique ID of the player that the packet is sent to.
Data	ByteArray	Data is the debug data.

156 Packet Packet Violation Warning

Serverbound

PacketViolationWarning is sent by the client when it receives an invalid packet from the server. It holds some information on the error that occurred.

Field Name	Field Type		Notes
Violation Type	zigzag32 enum		
	0		Malformed
Severity	zigzag32 enum		
	0	Warning	Severity specifies the severity of the packet violation. The action the client takes after this violation depends on the severity sent.

Field Name	Field Type	Notes
	1 Final Warning 2 Terminating	
Packet Id	zigzag32	PacketID is the ID of the invalid packet that was received.
Reason	string	ViolationContext holds a description on the violation of the packet.

157 Clientbound

Packet Motion Prediction Hints

MotionPredictionHints is sent by the server to the client. There is a predictive movement component for entities. This packet fills the "history" of that component and entity movement is computed based on the points. Vanilla sends this packet instead of the SetActorMotion packet when 'spatial optimisations' are enabled.

Field Name	Field Type	Notes
Entity Runtime Id	varint64	EntityRuntimeID is the runtime ID of the entity whose velocity is sent to the client.
Velocity	vec3f	Velocity is the server-calculated velocity of the entity at the point of sending the packet.
On Ground	bool	OnGround specifies if the server currently thinks the entity is on the ground.

158 Clientbound

Packet Animate Entity

AnimateEntity is sent by the server to animate an entity client-side. It may be used to play a single animation, or to activate a controller which can start a sequence of animations based on different conditions specified in an animation controller. Much of the documentation of this packet can be found at https://learn.microsoft.com/minecraft/creator/reference/content/animationsreference.

Field Name	Field Type	Notes
Animation	string	Animation is the name of a single animation to start playing.
Next State	string	NextState is the first state to start with. These states are declared in animation controllers (which, in themselves, are animations too). These states in turn may have animations and transitions to move to a next state.
Stop Condition	string	StopCondition is a MoLang expression that specifies when the animation should be stopped.
Stop Condition Version	li32	StopConditionVersion is the MoLang stop condition version.
Controller	string	Controller is the animation controller that is used to manage animations. These controllers decide when to play which animation.
Blend Out Time	lf32	How long to move from the previous animation to the next.
Runtime Entity Ids length	varint	EntityRuntimeIDs is list of runtime IDs of entities that the animation should be applied to.
Runtime Entity Ids array	varint64	

159 ## Packet Camera Shake

Clientbound

CameraShake is sent by the server to make the camera shake client-side. This feature was added for map- making partners.

Field Name	Field Type	Notes
Intensity	lf32	Intensity is the intensity of the shaking. The client limits this value to 4, so anything higher may not work.
Duration	lf32	Duration is the number of seconds the camera will shake for.

Field Name	Field Type		Notes
Type	u8		Type is the type of shake, and is one of the constants listed above. The different type affects how the shake looks in game.
Action	u8 enum		
	0	Add	Action is the action to be performed, and is one of the constants listed above. Currently the different actions will either add or stop shaking the client.
	1	Stop	

160

Packet Player Fog

Clientbound

PlayerFog is sent by the server to render the different fogs in the Stack. The types of fog are controlled by resource packs to change how they are rendered, and the ability to create custom fog.

Field Name	Field Type	Notes
Stack length	varint	Stack is a list of fog identifiers to be sent to the client. Examples of fog identifiers are "minecraft:fog*ocean" and "minecraft:fog*hell".
Stack array	string	

161

Packet Correct Player Move Prediction

Clientbound

CorrectPlayerMovePrediction is sent by the server if and only if StartGame.ServerAuthoritativeMovementMode is set to AuthoritativeMovementModeServerWithRewind. The packet is used to correct movement at a specific point in time.

Field Name	Field Type	Notes
Prediction Type	u8 enum	
	0	Player

Field Name	Field Type	Notes
	1	Vehicle
Position	vec3f	Position is the position that the player is supposed to be at at the tick written in the field below. The client will change its current position based on movement after that tick starting from the Position.
Delta	vec3f	Delta is the change in position compared to what the client sent as its position at that specific tick.
Rotation	vec2f	
Angular Velocity **Optional**	lf32	
On Ground	bool	OnGround specifies if the player was on the ground at the time of the tick below.
Tick	varint64	Tick is the tick of the movement which was corrected by this packet.

162 Packet Item Registry

Clientbound

ItemRegistryPacket is used to declare what items the server makes available, and components of custom items. After 1.21.60, this packet replaces the functionality of the "itemstates" field in the StartGamePacket In pre-1.21.60 versions, this was the ItemComponentPacket, and was sent by the server to attach client-side components to custom items.

Field Name	Field Type	Notes
Itemstates	Itemstates	items holds a list of all items.

163 Packet Filter Text Packet

Bidirectional

FilterText is sent by the both the client and the server. The client sends the packet to the server to allow the server to filter the text server-side. The server then responds with the same packet and the safer version of the text.

Field Name	Field Type	Notes
Text	string	Text is either the text from the client or the safer version of the text sent by the server.
From Server	bool	FromServer indicates if the packet was sent by the server or not.

164

Packet Debug Renderer

Clientbound

ClientBoundDebugRenderer is sent by the server to spawn an outlined cube on client-side.

Field Name	Field Type		Notes
Type	li32 enum		
	1	Clear	Type is the type of action. It is one of the constants above.
	2	Add Cube	
🔁 *if Type*	*Is Clear*	void	
	Is Add Cube		
	Text	string	Text is the text that is displayed above the debug.
	Position	vec3f	Position is the position to spawn the debug on.
	Red	lf32	Red is the red value from the RGBA colour rendered on the debug.
	Green	lf32	Green is the green value from the RGBA colour rendered on the debug.
	Blue	lf32	Blue is the blue value from the RGBA colour rendered on the debug.
	Alpha	lf32	Alpha is the alpha value from the RGBA colour rendered on the debug.

Field Name	Field Type			Notes
		Duration	li64	Duration is how long the debug will last in the world for. It is measured in milliseconds.

165 ## Packet Sync Entity Property

Clientbound

Sent by the server to synchronize/update entity properties as NBT, an alternative to Set Entity Data.

Field Name	Field Type	Notes
Nbt	nbt	

166 ## Packet Add Volume Entity

Clientbound

AddVolumeEntity sends a volume entity's definition and components from server to client.

Field Name	Field Type		Notes
Runtime Id	varint64		EntityRuntimeID is the runtime ID of the entity. The runtime ID is unique for each world session, and entities are generally identified in packets using this runtime ID.
Nbt	nbt		EntityMetadata is a map of entity metadata, which includes flags and data properties that alter in particular the way the entity looks.
Encoding Identifier	string		
Instance Name	string		
Bounds	**Min**		BlockCoordinates
	Max		BlockCoordinates
Dimension	zigzag32		
Engine Version	string		

167

Packet Remove Volume Entity

Clientbound

RemoveVolumeEntity indicates a volume entity to be removed from server to client.

Field Name	Field Type	Notes
Entity Id	varint64	The Runtime Entity ID

168

Packet Simulation Type

Datatype

SimulationType is an in-progress packet. We currently do not know the use case.

Field Name	Field Type		Notes
Type	u8 enum		
	0	Game	SimulationType is the simulation type selected
	1	Editor	
	2	Test	
	3	Invalid	

169

Packet Npc Dialogue

Clientbound

NPCDialogue is a packet that allows the client to display dialog boxes for interacting with NPCs.

Field Name	Field Type		Notes
Entity Id	lu64		ActorUniqueID is the ID of the NPC being requested.
Action Type	varint enum		
	0	Open	ActionType is the type of action for the packet.
	1	Close	

Field Name	Field Type	Notes
Dialogue	string	Dialogue is the text that the client should see.
Screen Name	string	SceneName is the scene the data was pulled from for the client.
Npc Name	string	NPCName is the name of the NPC to be displayed to the client.
Action Json	string	ActionJSON is the JSON string of the buttons/actions the server can perform.

170

Packet Edu Uri Resource Packet

Datatype

Field Name	Field Type	Notes
Resource	EducationSharedResourceURI	

171

Packet Create Photo

Clientbound

CreatePhoto is a packet that allows players to export photos from their portfolios into items in their inventory. This packet only works on the Education Edition version of Minecraft.

Field Name	Field Type	Notes
Entity Unique Id	li64	EntityUniqueID is the unique ID of the entity.
Photo Name	string	PhotoName is the name of the photo.
Item Name	string	ItemName is the name of the photo as an item.

172

Packet Update Subchunk Blocks

Clientbound

UpdateSubChunkBlocks is essentially just UpdateBlock packet, however for a set of blocks in a sub chunk.

Field Name	Field Type	Notes
X	zigzag32	SubChunkX, SubChunkY, and SubChunkZ help identify the sub chunk.
Y	zigzag32	
Z	zigzag32	
Blocks length	varint	Blocks contains each updated block change entry.
Blocks array	BlockUpdate	
Extra length	varint	Extra contains each updated block change entry for the second layer, usually for waterlogged blocks.
Extra array	BlockUpdate	

173 Packet Photo Info Request

Serverbound

Field Name	Field Type	Notes
Photo Id	zigzag64	

Type HeightMapDataType

Datatype

Field Name	Field Type	Notes
HeightMapDataType	u8 enum	
	0	No Data
	1	Has Data
	2	Too High
	3	Too Low
	4	All Copied

Type **SubChunkEntryWithoutCaching** Datatype

Field Name	Field Type		Notes
SubChunkEntryWithoutCaching length	lu32		
SubChunkEntryWithoutCaching array	**Dx**	i8	
	Dy	i8	
	Dz	i8	
	Result	u8 enum	
		0 Undefined	
		1 Success	
		2 Chunk Not Found	
		3 Invalid Dimension	
		4 Player Not Found	
		5 Y Index Out Of Bounds	
		6 Success All Air	
	Payload	ByteArray	Payload has the terrain data, if the chunk isn't empty and caching is disabled
	Heightmap Type	HeightMapDataType	
	Heightmap		
	if Heightmap Type	*Is Has Data*	["buffer", { "count": 256 }]
	Render Heightmap	HeightMapDataType	

Field Name	Field Type		Notes
	Type		
	Render Heightmap *if Render Heightmap Type*		
		Is Has Data	["buffer", { "count": 256 }]

Type **SubChunkEntryWithCaching** Datatype

Field Name	Field Type		Notes
SubChunkEntryWithCaching length	lu32		
SubChunkEntryWithCaching array	**Dx**	i8	
	Dy	i8	
	Dz	i8	
	Result	u8 enum	
		0	Undefined
		1	Success
		2	Chunk Not Found
		3	Invalid Dimension
		4	Player Not Found
		5	Y Index Out Of Bounds
		6	Success All Air

Field Name	Field Type			Notes
	Payload *if Result*	*Is Success All Air*	void	Payload has the terrain data, if the chunk isn't empty and caching is disabled
		Default	ByteArray	
	Heightmap Type	HeightMapDataType		
	Heightmap *if Heightmap Type*	*Is Has Data*	["buffer", { "count": 256 }]	
	Render Heightmap Type	HeightMapDataType		
	Render Heightmap *if Render Heightmap Type*	*Is Has Data*	["buffer", { "count": 256 }]	
	Blob Id	lu64		

174

Packet Subchunk

Clientbound

SubChunk sends data about multiple sub-chunks around a center point.

Field Name	Field Type	Notes
Cache Enabled	bool	
Dimension	zigzag32	

Field Name	Field Type	Notes
Origin	vec3i	Origin point
Entries *if Cache Enabled*	*Is True*	SubChunkEntryWithCaching
	Is False	SubChunkEntryWithoutCaching

175 Serverbound

Packet Subchunk Request

Field Name	Field Type	Notes
Dimension	zigzag32	
Origin	vec3i	Origin point
Requests length	lu32	
Requests array	**Dx**	i8
	Dy	i8
	Dz	i8

176 Serverbound

Packet Client Start Item Cooldown

ClientStartItemCooldown is sent by the client to the server to initiate a cooldown on an item. The purpose of this packet isn't entirely clear.

Field Name	Field Type	Notes
Category	string	
Duration	zigzag32	Duration is the duration of ticks the cooldown should last.

177 Serverbound

Packet Script Message

ScriptMessage is used to communicate custom messages from the client to the server, or from the server to the client. While the name may suggest this packet is used for the discontinued scripting API, it is likely instead for the GameTest framework.

Field Name	Field Type	Notes
Message Id	string	Message ID is the identifier of the message, used by either party to identify the message data sent.
Data	string	Data contains the data of the message.

178 Serverbound

Packet Code Builder Source

CodeBuilderSource is an Education Edition packet sent by the client to the server to run an operation with a

Field Name	Field Type		Notes
Operation	u8 enum		
	0	None	Operation is used to distinguish the operation performed. It is always one of the constants listed above.
	1	Get	
	2	Set	
	3	Reset	
Category	u8 enum		
	0	None	Category is used to distinguish the category of the operation performed. It is always one of the constants
	1	Code Status	
	2	Instantiation	
Code Status	u8 enum		
	0	None	
	1	Not Started	

Field Name	Field Type	Notes
	2	In Progress
	3	Paused
	4	Error
	5	Succeeded

179 Packet Ticking Areas Load Status

Clientbound

TickingAreasLoadStatus is sent by the server to the client to notify the client of a ticking area's loading status.

Field Name	Field Type	Notes
Preload	bool	Preload is true if the server is waiting for the area's preload.

180 Packet Dimension Data

Clientbound

DimensionData is a packet sent from the server to the client containing information about data-driven dimensions that the server may have registered. This packet does not seem to be sent by default, rather only being sent when any data-driven dimensions are registered.

Field Name	Field Type		Notes
Definitions length	varint		
Definitions array	**Id**	string	
	Max Height	zigzag32	
	Min Height	zigzag32	
	Generator	zigzag32 enum	
		0 Legacy	
		1 Overworld	
		2 Flat	
		3 Nether	

Field Name	Field Type		Notes
		4	End
		5	Void

Clientbound

181

Packet Agent Action

AgentAction is an Education Edition packet sent from the server to the client to return a response to a previously requested action.

Field Name	Field Type	Notes
Request Id	string	

Field Name	Field Type	Notes
Action Type	zigzag32 enum	
	0 None	
	1 Attack	
	2 Collect	
	3 Destroy	
	4 Detect Redstone	
	5 Detect Obstacle	
	6 Drop	
	7 Drop All	
	8 Inspect	
	9 Inspect Data	
	10 Inspect Item Count	
	11 Inspect Item Detail	
	12 Inspect Item Space	
	13 Interact	
	14 Move	
	15 Place Block	
	16 Till	
	17 Transfer Item To	
	18 Turn	
Body	string	

182 **Packet Change Mob Property** Clientbound

ChangeMobProperty is a packet sent from the server to the client to change one of the properties of a mob client-side.

Field Name	Field Type	Notes
Entity Unique Id	zigzag64	EntityUniqueID is the unique ID of the entity whose property is being changed.
Property	string	Property is the name of the property being updated.
Bool Value	bool	BoolValue is set if the property value is a bool type. If the type is not a bool, this field is ignored.
String Value	string	StringValue is set if the property value is a string type. If the type is not a string, this field is ignored.
Int Value	zigzag32	IntValue is set if the property value is an int type. If the type is not an int, this field is ignored.
Float Value	lf32	FloatValue is set if the property value is a float type. If the type is not a float, this field is ignored.

183

Packet Lesson Progress

Clientbound

LessonProgress is a packet sent by the server to the client to inform the client of updated progress on a lesson. This packet only functions on the Minecraft: Education Edition version of the game.

Field Name	Field Type	Notes
Action	u8	Action is the action the client should perform to show progress. This is one of the constants defined above.
Score	zigzag32	Score is the score the client should use when displaying the progress.
Identifier	string	Identifier is the identifier of the lesson that is being progressed.

184

Packet Request Ability

Serverbound

RequestAbility is a packet sent by the client to the server to request permission for a specific ability from the server.

Field Name	Field Type		Notes
Ability	zigzag32 enum		
	0	Build	Ability is the ability that the client is requesting. This is one of the constants defined above.
	1	Mine	
	2	Doors And Switches	
	3	Open Containers	
	4	Attack Players	
	5	Attack Mobs	
	6	Operator Commands	
	7	Teleport	
	8	Invulnerable	
	9	Flying	
	10	May Fly	
	11	Instant Build	
	12	Lightning	
	13	Fly Speed	
	14	Walk Speed	
	15	Muted	
	16	World Builder	
	17	No Clip	
	18	Ability Count	
Value Type	u8 enum		
	1	Bool	Value type decides which of the fields you should read/write from
	2	Float	

Field Name	Field Type	Notes
Bool Value	bool	If value type is bool, use this value
Float Val	lf32	If value type is float, use this value

185 Serverbound

Packet Request Permissions

RequestPermissions is a packet sent from the client to the server to request permissions that the client does not currently have. It can only be sent by operators and host in vanilla Minecraft.

Field Name	Field Type	Notes
Entity Unique Id	li64	EntityUniqueID is the unique ID of the player. The unique ID is unique for the entire world and is often used in packets. Most servers send an EntityUniqueID equal to the EntityRuntimeID.
Permission Level	PermissionLevel	PermissionLevel is the current permission level of the player. Same as constants in AdventureSettings packet.
Requested Permissions	RequestPermissions	RequestedPermissions contains the requested permission flags.

186 Clientbound

Packet Toast Request

ToastRequest is a packet sent from the server to the client to display a toast to the top of the screen. These toasts are the same as the ones seen when, for example, loading a new resource pack or obtaining an achievement.

Field Name	Field Type	Notes
Title	string	Title is the title of the toast.
Message	string	Message is the message that the toast may contain alongside the title.

187

Packet Update Abilities

Clientbound

UpdateAbilities is a packet sent from the server to the client to update the abilities of the player. It, along with the UpdateAdventureSettings packet, are replacements of the AdventureSettings packet since v1.19.10.

Field Name	Field Type	Notes
Entity Unique Id	li64	EntityUniqueID is the unique ID of the player. The unique ID is a value that remains consistent across different sessions of the same world, but most servers simply fill the runtime ID of the entity out for this field.
Permission Level	PermissionLevel	PlayerPermissions is the permission level of the player. It is a value from 0-3, with 0 being visitor, 1 being member, 2 being operator and 3 being custom.
Command Permission	CommandPermissionLevel	CommandPermissions is a permission level that specifies the kind of commands that the player is allowed to use. It is one of the CommandPermissionLevel constants in the AdventureSettings packet.
Abilities length	u8	Layers contains all ability layers and their potential values. This should at least have one entry, being the base layer.
Abilities array	AbilityLayers	

188

Packet Update Adventure Settings

Clientbound

UpdateAdventureSettings is a packet sent from the server to the client to update the adventure settings of the player. It, along with the UpdateAbilities packet, are replacements of the AdventureSettings packet since v1.19.10.

Field Name	Field Type	Notes
No Pvm	bool	NoPvM is a boolean indicating whether the player is allowed to fight mobs or not.
No Mvp	bool	NoMvP is a boolean indicating whether mobs are allowed to fight the player or not. It is unclear why this is sent to the client.
Immutable World	bool	ImmutableWorld is a boolean indicating whether the player is allowed to modify the world or not.
Show Name Tags	bool	ShowNameTags is a boolean indicating whether player name tags are shown or not.
Auto Jump	bool	AutoJump is a boolean indicating whether the player is allowed to jump automatically or not.

189

Packet Death Info

Clientbound

DeathInfo is a packet sent from the server to the client expected to be sent when a player dies. It contains messages related to the player's death, which are shown on the death screen as of v1.19.10.

Field Name	Field Type	Notes
Cause	string	Cause is the cause of the player's death, such as "suffocation" or "suicide".
Messages length	varint	Messages is a list of death messages to be shown on the death screen.
Messages array	string	

190

Packet Editor Network

Clientbound

EditorNetwork is a packet sent from the server to the client and vise-versa to communicate editor-mode related information. It carries a single compound tag containing the relevant information.

Field Name	Field Type	Notes
Route To Manager	bool	
Payload	nbt	Payload is a network little endian compound tag holding data relevant to the editor.

191 Clientbound

Packet Feature Registry

FeatureRegistry is a packet used to notify the client about the world generation features the server is currently using. This is used in combination with the client-side world generation system introduced in v1.19.20, allowing the client to completely generate the chunks of the world without having to rely on the server.

Field Name	Field Type	Notes	
Features length	varint	Features is a slice of all registered world generation features.	
Features array	**Name**		string
	Options		string

192 Clientbound

Packet Server Stats

ServerStats is a packet sent from the server to the client to update the client on server statistics. It is purely used for telemetry.

Field Name	Field Type	Notes
Server Time	lf32	
Network Time	lf32	

193 Serverbound

Packet Request Network Settings

Field Name	Field Type	Notes
Client Protocol	i32	

194 **Packet Game Test Request** Serverbound

Field Name	Field Type		Notes
Max Tests Per Batch	varint		MaxTestsPerBatch …
Repetitions	varint		Repetitions represents the amount of times the test will be run.
Rotation	u8 enum		
	0	0deg	Rotation represents the rotation of the test. It is one of the constants above.
	1	90deg	
	2	180deg	
	3	270deg	
	4	360deg	
Stop On Error	bool		StopOnError indicates whether the test should immediately stop when an error is encountered.
Position	BlockCoordinates		Position is the position at which the test will be performed.
Tests Per Row	varint		TestsPerRow …
Name	string		Name represents the name of the test.

195 **Packet Game Test Results** Clientbound

GameTestResults is a packet sent in response to the GameTestRequest packet, with a boolean indicating whether the test was successful or not, and an error string if the test failed.

Field Name	Field Type	Notes
Succeeded	bool	Succeeded indicates whether the test succeeded or not.
Error	string	Error is the error that occurred. If Succeeded is true, this field is empty.
Name	string	Name represents the name of the test.

196 ## Packet Update Client Input Locks

Clientbound

Field Name	Field Type	Notes
Locks	InputLockFlags	Locks is an encoded bitset of all locks that are currently active. The locks are defined in the constants above.
Position	vec3f	Position is the server's position of the client at the time the packet was sent. It is unclear what the exact purpose of this field is.

197 ## Packet Client Cheat Ability

Clientbound

Deprecated: ClientCheatAbility is deprecated as of 1.20.10.

Field Name	Field Type	Notes
Entity Unique Id	li64	EntityUniqueID is the unique ID of the player. The unique ID is a value that remains consistent across different sessions of the same world, but most servers simply fill the runtime ID of the entity out for this field.
Permission Level	PermissionLevel	PlayerPermissions is the permission level of the player. It is a value from 0-3, with 0 being visitor, 1 being member, 2 being operator and 3 being custom.

Field Name	Field Type	Notes
Command Permission	CommandPermissionLevel	CommandPermissions is a permission level that specifies the kind of commands that the player is allowed to use. It is one of the CommandPermissionLevel constants in the AdventureSettings packet.
Abilities length	u8	Layers contains all ability layers and their potential values. This should at least have one entry, being the base layer.
Abilities array	AbilityLayers	

198 Clientbound

Packet Camera Presets

camera_presets gives the client a list of custom camera presets.

Field Name	Field Type	Notes
Presets length	varint	
Presets array	CameraPresets	

199 Clientbound

Packet Unlocked Recipes

unlocked_recipes gives the client a list of recipes that have been unlocked, restricting the recipes that appear in the recipe book.

Field Name	Field Type		Notes
Unlock Type	lu32 enum		
	0	Empty	new_unlocks determines if new recipes have been unlocked since the packet was last sent.
	1	Initially Unlocked	
	2	Newly Unlocked	

Field Name	Field Type	Notes
	3 Remove Unlocked 4 Remove All Unlocked	
Recipes length	varint	Recipes is a list of recipe names that have been unlocked.
Recipes array	string	

300 **Packet Camera Instruction** Clientbound

camera_instruction gives a custom camera specific instructions to operate.

Field Name	Field Type			Notes
Instruction Set Optional	**Runtime Id**	li32		Instruction Set is a camera instruction that sets the camera to a specified preset. Preset is the index of the preset in the CameraPresets packet sent to the player.
	Ease Data Optional	**Type**	EaseType	Ease represents the easing function that is used by the instruction.
		Duration	lf32	
	Position **Optional**	vec3f		
	Rotation **Optional**	vec2f		
	Facing **Optional**	vec3f		

Field Name	Field Type	Notes
	Offset **Optional**	vec2f	TODO: Investigate when this was added ViewOffset is only used in a follow_orbit camera and controls an offset based on a pivot point to the player, causing it to be shifted in a certain direction.
	Entity Offset **Optional**	vec3f	
	Default **Optional**	bool	
	Remove Ignore Starting Values	bool	
Clear **Optional**	bool	Clear can be set to true to clear all the current camera instructions.
Fade Optional	**Fade In Duration**	lf32	Fade is a camera instruction that fades the screen to a specified colour.
	Wait Duration	lf32	
	Fade Out Duration	lf32	
	Color Rgb	vec3f	
Target Optional	**Offset** **Optional**	vec3f	Target is a camera instruction that targets a specific entity.
	Entity Unique Id	li64	
Remove Target **Optional**	bool	RemoveTarget can be set to true to remove the current aim assist target.
Fov Optional	**Field Of View**	lf32
	Ease Time	lf32
	Ease Type	EaseType
	Clear	bool

301 Clientbound

Packet Compressed Biome Definitions

Removed in 1.21.80

Field Name	Field Type	Notes
Raw Payload	ByteArray	via PMMP: This packet is only sent by the server when client-side chunk generation is enabled in vanilla. It contains NBT data for biomes, similar to the BiomeDefinitionListPacket, but with a large amount of extra data for client-side chunk generation. The data is compressed with a cursed home-brewed compression format, and it's a miracle it even works.

302 Clientbound

Packet Trim Data

Field Name	Field Type	Notes
Patterns length	varint	
Patterns array	**Item Name**	string
	Pattern	string
Materials length	varint	
Materials array	**Material**	string
	Color	string
	Item Name	string

303 Clientbound

Packet Open Sign

Field Name	Field Type	Notes
Position	BlockCoordinates	
Is Front	bool	

304 Packet Agent Animation

Clientbound

agent_animation is an Education Edition packet sent from the server to the client to make an agent perform an animation.

Field Name	Field Type		Notes
Animation	u8 enum		
	0	Arm Swing	animation is the ID of the animation that the agent should perform. As of its implementation, there are no IDs that can be used in the regular client.
	1	Shrug	
Entity Runtime Id	varint64		entity*runtime*id is the runtime ID of the entity. The runtime ID is unique for each world session, and entities are generally identified in packets using this runtime ID.

305 Packet Refresh Entitlements

Serverbound

RefreshEntitlements is sent by the client to the server to refresh the entitlements of the player.

Field Name	Field Type	Notes

306 Packet Toggle Crafter Slot Request

Datatype

Field Name	Field Type	Notes
Position	vec3li	
Slot	u8	
Disabled	bool	

Packet Set Player Inventory Options

Field Name	Field Type	Notes
Left Tab	zigzag32 enum	
	0 None	
	1 Construction	
	2 Equipment	
	3 Items	
	4 Nature	
	5 Search	
	6 Survival	
Right Tab	zigzag32 enum	
	0 None	
	1 Fullscreen	
	2 Crafting	
	3 Armor	
Filtering	bool	
Layout	zigzag32 enum	
	0 None	
	1 Survival	
	2 Recipe Book	
	3 Creative	
Crafting Layout	zigzag32 enum	
	0 None	
	1 Survival	
	2 Recipe Book	
	3 Creative	

Datatype

308 Packet Set Hud

SetHud is sent by the server to set the visibility of individual HUD elements on the client. It is important to note that the client does not reset the state of the HUD elements after it leaves a server, meaning they can leak into sessions on different servers. To be safe, you should reset the visibility of all HUD elements when a player connects.

Field Name	Field Type		Notes
Elements length	varint		Elements is a list of HUD elements that are being modified. The values can be any of the HudElement constants above.
Elements array	Element		
Visibility	varint enum		
	0	Hide	Visibility represents the new visibility of the specified Elements. It can be any of the HudVisibility constants above.
	1	Reset	

Datatype

Type Element

Field Name	Field Type	
Element	varint enum	
	0	PaperDoll
	1	Armour
	2	ToolTips
	3	TouchControls
	4	Crosshair
	5	HotBar
	6	Health
	7	ProgressBar
	8	Hunger

Field Name	Field Type	Notes
	9	AirBubbles
	10	VehicleHealth
	11	EffectsBar
	12	ItemTextPopup

309

Packet Award Achievement

Clientbound

Field Name	Field Type	Notes
Achievement Id	li32	

16

Packet Server Post Move

Datatype

Field Name	Field Type	Notes
Position	vec3f	

310

Packet Clientbound Close Form

Clientbound

clientbound*close*form is sent by the server to clear the entire form stack of the client. This means that all forms that are currently open will be closed. This does not affect inventories and other containers.

Field Name	Field Type	Notes

312

Packet Serverbound Loading Screen

Serverbound

skip 0x137 ServerBoundLoadingScreen is sent by the client to tell the server about the state of the loading screen that the client is currently displaying.

Field Name	Field Type	Notes
Type	zigzag32	The type of the loading screen event.
Loading Screen Id **Optional**	lu32	

313 Clientbound

Packet Jigsaw Structure Data

JigsawStructureData is sent by the server to let the client know all the rules for jigsaw structures.

Field Name	Field Type	Notes
Structure Data	nbt	StructureData is a network NBT serialised compound of all the jigsaw structure rules defined on the server.

314 Clientbound

Packet Current Structure Feature

CurrentStructureFeature is sent by the server to let the client know the name of the structure feature that the player is currently occupying.

Field Name	Field Type	Notes
Current Feature	string	CurrentFeature is the identifier of the structure feature that the player is currently occupying. If the player is not occupying any structure feature, this field is empty.

315 Serverbound

Packet Serverbound Diagnostics

ServerBoundDiagnostics is sent by the client to tell the server about the performance diagnostics of the client. It is sent by the client roughly every 500ms or 10 in-game ticks when the "Creator > Enable Client Diagnostics" setting is enabled.

Field Name	Field Type	Notes
Average Frames Per Second	lf32	
Average Server Sim Tick Time	lf32	

Field Name	Field Type	Notes
Average Client Sim Tick Time	lf32	
Average Begin Frame Time	lf32	
Average Input Time	lf32	
Average Render Time	lf32	
Average End Frame Time	lf32	
Average Remainder Time Percent	lf32	
Average Unaccounted Time Percent	lf32	

316

Packet Camera Aim Assist

Serverbound

CameraAimAssist is sent by the server to the client to set up aim assist for the client's camera.

Field Name	Field Type		Notes
Preset Id	string		Preset is the ID of the preset that has previously been defined in the CameraAimAssistPresets packet.
View Angle	vec2f		CameraAimAssistTargetModeAngle.
Distance	lf32		Distance is the distance that the camera should keep from the target, if TargetMode is set to CameraAimAssistTargetModeDistance.
Target Mode	u8 enum		
	0	Angle	TargetMode is the mode that the camera should use to aim at the target. This is one of the constants below.
	1	Distance	
Action	u8 enum		
	0	Set	Action is the action that should be performed with the aim assist. This is one of the constants above.
	1	Clear	
Show Debug	bool		

Field Name	Field Type	Notes
Render		

317 Clientbound

Packet Container Registry Cleanup

ContainerRegistryCleanup is sent by the server to trigger a client-side cleanup of the dynamic container registry.

Field Name	Field Type	Notes
Removed Containers length	varint	RemovedContainers is a list of protocol.FullContainerName's that should be removed from the client-side container registry.
Removed Containers array	FullContainerName	

318 Clientbound

Packet Movement Effect

movement_effect is sent by the server to the client to update specific movement effects to allow the client to predict its movement. For example, fireworks used during gliding will send this packet to tell the client the exact duration of the boost.

Field Name	Field Type	Notes
Runtime Id	varint64	runtime_id is the runtime ID of the entity. The runtime ID is unique for each world session, and entities are generally identified in packets using this runtime ID.
Effect Type	MovementEffectType	effect_type is the type of movement effect being updated.
Effect Duration	varint	effect_duration is the duration of the effect, measured in ticks.
Tick	varint64	tick is the server tick at which the packet was sent. It is used in relation to CorrectPlayerMovePrediction.

319 Clientbound

Packet Set Movement Authority

set*movement*authority is sent by the server to the client to change its movement mode. This packet has been deprecated and as of protocol 818 (1.21.90) has been removed from the game.

Field Name	Field Type		Notes
Movement Authority	u8 enum		
	0	Client	movement_authority specifies the way the server handles player movement. Available options are PlayerMovementModeClient, PlayerMovementModeServer and PlayerMovementModeServerWithRewind, where the server authoritative types result in the client sending PlayerAuthInput packets instead of MovePlayer packets and the rewind mode requires sending the tick of movement and several actions.
	1	Server	
	2	Server With Rewind	

320 Clientbound

Packet Camera Aim Assist Presets

CameraAimAssistPresets is sent by the server to the client to provide a list of categories and presets that can be used when sending a CameraAimAssist packet or a CameraInstruction including aim assist.

Field Name	Field Type		Notes
Categories length	varint		CategoryGroups is a list of groups of categories which can be referenced by one of the Presets.
Categories array	**Name**	string	Identifier is the unique identifier of the group.
	Entity Priorities length	varint	Priorities represents the block and entity specific priorities for targetting. The aim assist will select the block or entity with

Field Name	Field Type	Notes
		the highest priority within the specified thresholds.
	Entity Priorities array	**Id**: string **Priority**: li32
	Block Priorities length: varint	
	Block Priorities array	**Id**: string **Priority**: li32
	Entity Default Optional: li32	
	Block Default Optional: li32	
Presets length: varint		
Presets array	**Id**: string	
	Exclude Blocks length: varint	
	Exclude Blocks array: string	
	Target Liquids length: varint	
	Target Liquids array: string	
	Item Settings length: varint	
	Item Settings array	**Id**: string — Identifier of the item to apply the settings to.

Field Name	Field Type	Notes		
		Category	string	Category is the identifier of a category to use which has been defined by a CameraAimAssistCategory.
	Default Item Settings **Optional**	string	DefaultItemSettings is the identifier of a category to use when the player is not holding an item listed in ItemSettings. This must be the identifier of a category within the above categories field.	
	Hand Settings **Optional**	string	HandSettings is the identifier of a category to use when the player is not holding an item. This must be the identifier of a category within the above categories field.	
Operation	u8 enum			
	0	Set		
	1	Add To Existing		

321

Packet Client Camera Aim Assist

Datatype

Field Name	Field Type		Notes
Preset Id	string		
Action	u8 enum		
	0	Set From Camera Preset	Sets aim-assist to use the settings from a CameraPresets aim_assist field.
	1	Clear	Clears aim-assist settings.
Allow Aim Assist	bool		

322

Packet Client Movement Prediction Sync

Datatype

Field Name	Field Type	Notes
Data Flags	varint128	
Bounding Box	**Scale**	lf32
	Width	lf32
	Height	lf32
Movement Speed	lf32	
Underwater Movement Speed	lf32	
Lava Movement Speed	lf32	
Jump Strength	lf32	
Health	lf32	
Hunger	lf32	
Entity Runtime Id	varint64	
Is Flying	bool	

323

Packet Update Client Options

Bidirectional

Field Name	Field Type	Notes
Graphics Mode **Optional**	u8 enum	
	0	Simple
	1	Fancy
	2	Advanced
	3	Ray Traced

324

Packet Player Video Capture

Clientbound

PlayerVideoCapturePacket is sent by the server to start or stop video recording for a player. This packet only works on development builds and has no effect on retail builds. When recording, the client will save individual frames to '/LocalCache/minecraftpe' in the format specified below.

Field Name	Field Type			Notes
Action	u8 enum			
	0	Stop		action is the action that the client should perform. This is one of the constants defined above.
	1	Start		
🔁 *if Action*	*Is Start*	**Frame Rate**	li32	frame_rate is the frame rate at which the video should be recorded. It is only used when Action is PlayerVideoCaptureActionStart. A higher frame rate will cause more frames to be recorded, but also a noticeable increase in lag.
		File Prefix	string	

Clientbound

Packet Player Update Entity Overrides

PlayerUpdateEntityOverrides is sent by the server to modify an entity's properties individually.

Field Name	Field Type		Notes
Runtime Id	varint64		EntityRuntimeID is the runtime ID of the entity. The runtime ID is unique for each world session, and entities are generally identified in packets using this runtime ID.
Property Index	varint		PropertyIndex is the index of the property to modify. The index is unique for each property of an entity.
Type	u8 enum		
	0	Clear All	Type is the type of action to perform with the property. It is one of the constants above.
	1	Remove	
	2	Set Int	
	3	Set Float	

Field Name	Field Type		Notes
Value *if Type*	*Is Set Int*	li32	IntValue is the new integer value of the property. It is only used when Type is set to PlayerUpdateEntityOverridesTypeInt.
	Is Set Float	lf32	FloatValue is the new float value of the property. It is only used when Type is set to PlayerUpdateEntityOverridesTypeFloat.

326

Packet Player Location

Clientbound

Field Name	Field Type		Notes
Type	li32 enum		
	0	Coordinates	Type is the action that is being performed. It is one of the constants above.
	1	Type Hide	
Entity Unique Id	varint64		
Position *if Type*	*Is Coordinates*	vec3f	

327

Clientbound

Packet Clientbound Controls Scheme

Field Name	Field Type		Notes
Scheme	u8 enum		
	0	Locked Player Relative Strafe	Scheme is the scheme that the client should use. It is one of the constants above.
	1	Camera Relative	

Field Name	Field Type		Notes
	2	Camera Relative Strafe	
	3	Player Relative	
	4	Player Relative Strafe	

Packet Server Script Debug Drawer

Field Name	Field Type	Notes
Shapes length	varint	
Shapes array	**Network Id**	varint64
	Shape Type **Optional**	u8 enum 0 Line 1 Box 2 Sphere 3 Circle 4 Text 5 Arrow
	Location **Optional**	vec3f
	Scale **Optional**	lf32
	Rotation **Optional**	vec3f
	Time Left **Optional**	lf32
	Color	li32

Field Name	Field Type	Notes
	Optional	
	Text **Optional**	string
	Box Bound **Optional**	vec3f
	Line End Location **Optional**	vec3f
	Arrow Head Length **Optional**	lf32
	Arrow Head Radius **Optional**	lf32
	Segment Count **Optional**	u8

Packet Serverbound Pack Setting Change

ServerBoundPackSettingChange

Field Name	Field Type		Notes
Pack Id	uuid		
Pack Setting	**Name**	string	
	Type	u8 enum	
		0 Float	
		1 Bool	
		2 String	
	Value	*Is Float*	lf32

Field Name	Field Type		Notes
	if Type	*Is Bool*	bool
		Is String	string