Code Builder for Minecraft: Education Edition

API Documentation

Contents

Commands for Agent

Return	Command	Description
[bool success]	move [string direction]	Attempts to move Agent in specified direction. /move?direction=forward
[bool success]	turn [string direction]	Attempts to rotate Agent 90 degrees. /turn?direction=left
[bool success]	place [int slotNum] [string direction]	Places a block in the selected inventory slot in direction specified. /place?slotNum=1&direction=left
[bool success]	till [string direction]	Tills the soil with diamond hoe in direction specified. /till?direction=left
[bool success]	attack [string direction]	Attacks in direction specified with diamond sword strength. /attack?direction=left
[bool success]	destroy [string direction]	Destroys any types of block by single hit in direction specified. /destroy?direction=left
[bool success]	collect [string item]	Attempts to collect items. /collect?item=all
[bool success]	drop [int slotNum] [int quantity] [string direction]	Drops specified Inventory Slot Numbered Items in the specified quantity in the specified direction. /drop?slotNum=1&quantity=10&direction=right
[bool success]	dropall [string direction]	Drops all Inventory Items in the specified direction. /dropall?direction=right
[bool result]	detect [string direction]	Detects if there is a destructible Block in direction specified. /detect?direction=right
[string blockName]	inspect [string direction]	Returns the name of the block in the specified direction. /inspect?direction=right
[int data]	inspectdata [string direction]	Returns the data value of the block in the specified direction. /inspectdata?direction=right
[bool result]	detectredstone [string direction]	Detects Redstone signal in specified direction. /detectredstone?direction=forward
[string itemName]	getitemdetail [int slotNum]	Returns the item name in the specified slot. /getitemdetail?slotNum=1
[int spaceCount]	getitemspace [int slotNum]	Returns the number of spaces remaining in the specified slot. /getitemspace?slotNum=1
[int stackCount]	getitemcount [int slotNum]	Returns the number of items in the specified slot. /getitemcount?slotNum=1
[bool success]	transfer [int srcslotNum] [int quantity] [int dstslotNum]	Transfers specified quantity of items from the selected slot to another specified slot of Agent's Inventory. /transfer?srcslotNum=1&quantity=64&dstslotNum=2

None	tptoplayer	Teleports Agent to player. /tptoplayer

Commands for world **(* indicates optional parameters)**

Return	Command	Description
[int count]	clone [blockpos begin] [blockpos end] [blockpos destination] *[string maskMode] *[string cloneMode] *[string tileName] *[int tileData]	Copies blocks from one place to another.
None	executeasother [target origin] [blockpos position] [string command]	Executes another command as the specified target. Will fail if they don't have permission for the command.
None	executedetect [target origin] [blockpos position] [string detect] [blockpos detectPos] [string detectBlock] [int detectData] [string command]	Executes another command if the block at the specified position matches the specified block type and data.
[int fillCount] [string blockName]	fill [blockpos from] [blockpos to] [string tileName] *[int tileData] *[string oldBlockHandling] *[string replaceTileName] *[int replaceDataValue]	Fills the region with the specific block and data. What happens to existing blocks at the given locations is determined by oldBlockHandling.
[string itemName] [int itemAmount] [string playerName]	give [target player] [string itemName] *[int amount] *[int data]	Gives an item to a player.
None	kill *[target target]	Kills entities (players, mobs, items, etc.). Default kills local player.
None	setblock [blockpos position] [string tileName] *[int tileData] *[string oldBlockHandling]	Changes a block to another block.
[bool wasSpawned]	summon [string entityType] [blockpos spawnPos]	Tries to summon an entity at specified position.
[bool matches]	testforblock [blockpos position] [string tileName]	Tests whether the specified block is at the specified location.

	*[int dataValue]	
[int compareCount] [bool matches]	testforblocks [blockpos begin] [blockpos end] [blockpos destination] *[string mode]	Tests whether the blocks in two regions match.
None	timesetbyname [string time]	Sets in game time.
None	timesetbynumber [int time]	Sets in game time to ‘day’ or ‘night’.
None	tptargettotarget [target victim] [target destination] *[int y-rot] *[int x-rot]	Teleports target victim to target destination.
None	tptargettopos [target victim] [blockpos destination] *[int y-rot] *[int x-rot]	Teleports target victim to destination position.
None	weather [string type] [int duration]	Sets the weather for the specified duration.

For more detail on world commands and their usage, go to:

http://minecraft.gamepedia.com/Commands

Using this API

The Code Builder experience involves three apps: Minecraft: Education Edition (MC), Code Connection (CC), and an editor, presumably in a browser. The editor sends command requests in the form of REST queries to localhost on port 8080. CC picks these up, converts them into MC's JSON format, and sends them over WebSockets to MC. MC executes the command, sends output back to CC, which parses it and responds to the REST query.

This means that to use CC it must be open and connected to MC. Once CC is on the editor selection screen it is ready to receive REST queries. There are two ways to accomplish this.

- Run command /code in MC. This will launch CC and automatically connect.
- Launch CC, then run command /connect localhost:19131 in MC.

The Scratch implementation is available as an example here:

https://github.com/Mojang/mojang.github.io

REST Queries and Responses

The app's rest server is listening on port 8080 and is used with queries of type "GET" specifying the command name followed by the arguments. Here is an example of a full (unencoded) URL:

http://localhost:8080/drop?slotnum=1&quantity=1&direction=forward

The app will always respond with JSON of some form. This will either be the expected return value, or an error code and message. Below is an example of possible return values of the "summon" command.

```
{
  wasSpawned: true
}
{
  errorCode: 2,
  errorMessage: "No WebSocket connection"
}
```

Format of "blockpos"

All blockpos parameters will be passed as strings as if they were typed in Minecraft. Below is an example (unencoded) of an absolute position followed by a relative one. Relative positions are relative to the player, not the Agent. For more information visit: http://minecraft.gamepedia.com/Commands/

/exampleCommand?absolute=1 2 3

/exampleCommand?relative=~1 ~2 ~3

Format of “target”

Similar to blockpos, targets are also specified as they would be in a Minecraft command. Below are two (unencoded) examples of targets. For a more in-depth description of target syntax, visit: http://minecraft.gamepedia.com/Commands/

/exampleCommand?targetparam=Steve

/exampleCommand?targetparam=@e[type=pig,x=1]

URL Encoding

Depending on the environment the REST queries are being made in it may be necessary to encode unsafe characters in the URLs for the commands. The app properly deals with encoded characters. Characters of concern are spaces, and characters from targets like ‘@’ ‘[‘ ‘]’ and ‘=’. Below is an example of this encoding.

http://localhost:8080/example?target=@e[type=pig,x=1]
Encoded to:
http://localhost:8080/example?target=%40e%5Btype%3Dpig%2Cx%3D1%5D

Queueing Commands

The extension that is performing the REST queries is responsible for queueing commands in such a way that there is only ever one pending command in this app. If multiple commands are initiated concurrently, all but the last will fail with error code 6 and only the last one will be executed.

Command Permissions

All Agent commands can be executed by any player. However, world commands require a higher permission level (mostly op) and will fail with error code 5 if the local player does not have the required permission level.

Error Codes

Value	Description
0	Missing parameters in REST query.
1	App failed to parse parameter type, meaning it is either in an invalid format or not supported.
2	App is not connected to Minecraft and cannot run commands.
3	App encountered a WebSocket error and was unable to send the command.
4	The command was executed, but the app was unable to parse Minecraft’s response.
5	The command failed on the Minecraft side, either because of invalid parameters or because the command hit a fail case. A common example would be specifying a target that doesn’t get any results. The message from Minecraft will be appended onto the “errorMessage” field.
6	The pending command was cancelled because the app received a new REST query before the pending command was complete.
7	Invalid REST endpoint.

move [string direction]

/move?direction=forward	
Attempts to move Agent in specified direction.	
Parameters	**[string direction]** forward, back, left, right, up, down
Return	**[bool success]** Returns if Agent moved in the direction specified. Would be false if he was blocked by a solid obstacle.

turn [string direction]

/turn?direction=left	
Attempts to rotate Agent 90 degrees.	
Parameters	**[string direction]** Left, Right
Return	**[bool success]** Always returns true as the Agent cannot fail to turn.

place [int slotNum] [string direction]

/place?slotNum=1&direction=left	
Right-clicks on the block in the specified direction. This basically places a block (ex: Place Stone Block) or a place-able item (ex: Place Oak Fence) from the selected Inventory Slot in the specified direction. However, this doesn't interact with things like doors, levers, and containers. Additionally, when player specifies a slot which contains the following non-place-able items, this behaves as follows: Get water/lava - Bucket Till - hoe Make a path - shovel Spawn - Spawn eggs Dye - dyePowder Fire - flint and steel - fire charge (I believe this is a fireball) Just drop - Boat - Minecart - Bottle (try to make a bottle whatever you use it on, and be dropped)	
Parameters	**[int slotNum]** Inventory Slot Number (1-27) **[string direction]** forward, back, left, right, up, down
Return	**[bool success]** Returns if a block in the specified inventory slot was placed.

attack [string direction]

/attack?direction=left	
Attacks in direction specified (diamond sword strength)	
Parameters	**[string direction]** forward, back, left, right, up, down
Return	**[bool success]** Returns if the Agent damaged an entity.

destroy [string direction]

/destroy?direction=left	
Destroys breakable Block or Item in direction specified. Any breakable blocks can be broken in one hit.	
Parameters	**[string direction]** Forward, Back, Left, Right, Up, Down
Return	**[bool success]** Returns if the Agent destroyed a block. This would be false if the block was unbreakable or wasn't solid, such as air or water.

till [string direction]

/till?direction=left	
Tills the soil by hoe in direction specified.	
Parameters	**[string direction]** Forward, Back, Left, Right, Up, Down
Return	**[bool success]** Returns if the Agent could till, meaning it's a dirt-like material that can be turned into farmland.

collect [string item]

/collect?item=all	
Attempts to collect all items within a one block from Agent in three dimensions.	
Parameters	**[string item]** all, Item name (stone, lapis_ore, etc.) Full list here: http://minecraft.gamepedia.com/Data_values/Block_IDs Filters by given type
Return	**[bool success]** Returns if any items were collected.

drop [int slotNum] [int quantity] [string direction]

/drop?slotNum=1&quantity=10&direction=right	
Drops the given number of items from the specified inventory slot onto the ground one block in the specified direction.	
Parameters	**[int slotNum]** Inventory Slot Number (1-27) **[int quantity]** Quantity (1-64) **[string direction]** forward, back, left, right
Return	**[bool success]** Returns if any items were dropped.

dropall [string direction]

/dropall?direction=right	
Drops all items from all slots onto the ground one block in the specified direction.	
Parameters	**[string direction]** forward, back, left, right
Return	**[bool success]** Returns if any items were dropped.

detect [string direction]

/detect?direction=right	
Detects if there is a collideable block in specified direction. Does not detect mobs.	
Parameters	**[string direction]** forward, back, left, right, up, down
Return	**[bool result]** Returns if there was a collideable block in the specified direction.

inspect [string direction]

/inspect?direction=right	
Returns the name of the block in the specified direction.	
Parameters	**[string direction]** forward, back, left, right, up, down
Return	**[string itemName]** Of the form “coal_ore”. Full list here: http://minecraft.gamepedia.com/Data_values/Block_IDs

inspectdata [string direction]

/inspectdata?direction=right	
Returns the data value of the block in the specified direction.	
Parameters	**[string direction]** forward, back, left, right, up, down
Return	**[int data]** Data value of the block in the specified direction, 0 for air.

detectredstone [string direction]

/detectredstone?direction=forward	
Detects Redstone signal in specified direction.	
Parameters	**[string direction]** forward, back, left, right, up, down
Return	**[bool result]** Returns if the object in the specified direction is receiving Redstone power at that moment.

getitemdetail [int slotNum]

/getitemdetail?slotNum=1	
Returns the name of the item in the specified slot.	
Parameters	**[int slotNum]** Inventory Slot Number
Return	**[string itemName]** Returns item name of the form "coal_ore". Full list here: http://minecraft.gamepedia.com/Data_values/Block_IDs

getitemspace [int slotNum]

/getitemspace?slotNum=1	
Returns the number of spaces remaining in the specified slot, as in the number of items you could add before it would be full.	
Parameters	**[int slotNum]** Inventory Slot Number
Return	**[int stackCount]**

getitemcount [int slotNum]

/getitemcount?slotNum=1	
Returns the number of items in the specified slot, as in how many you could use before it would be empty.	
Parameters	**[int slotNum]** Inventory Slot Number
Return	**[int stackCount]** Returns the number of items in the specified slot.

transfer [int srcslotNum] [int quantity] [int dstslotNum]

<table>
<tr><td colspan="2">/transfer?srcslotNum=1&quantity=64&dstslotNum=2</td></tr>
<tr><td colspan="2">Transfers specified quantity of items from the selected slot to another specified slot of Agent's Inventory.</td></tr>
<tr><td>Parameters</td><td>[int srcslotNum]
Source Inventory Slot Number

[int quantity]
Quantity (1-64)

[int dstslotNum]
Destination Inventory Slot Number</td></tr>
<tr><td>Return</td><td>[bool success]
If the destination slot already has items of a different type, it will return false (does not try to fill the next slot). If there are fewer than the specified quantity of items in the selected slot or only room for fewer items in the destination slot, it will transfer as many as possible and return true. If none can be transferred, it returns false.</td></tr>
</table>

tptoplayer

<table>
<tr><td colspan="2">/tptoplayer</td></tr>
<tr><td colspan="2">Teleports the Agent to the player's feet.</td></tr>
<tr><td>Parameters</td><td>None</td></tr>
<tr><td>Return</td><td>None</td></tr>
</table>

Item list (1.1)

Blocks:

acacia_stairs,allow,bedrock,birch_stairs,border_block,brick_block,brick_stairs,clay,coal_ore,cobblestone,dark_oak_stairs,deny,diamond_ore,dirt,emerald_ore,end_bricks,end_stone,gold_ore,grass,gravel,hardened_clay,ice,iron_ore,jungle_stairs,lapis_ore,log,log2,mossy_cobblestone,mycelium,nether_brick,nether_brick_stairs,netherbrick,netherrack,oak_stairs,obsidian,packed_ice,planks,podzol,prismarine,purpur_block,purpur_stairs,quartz_block,quartz_ore,quartz_stairs,red_sandstone,red_sandstone_stairs,redstone_ore,sand,sandstone,sandstone_stairs,snow,soul_sand,spruce_stairs,stained_hardened_clay,stone,stone_brick_stairs,stone_slab,stone_slab2,stone_stairs,stonebrick,wooden_slab

Decorations:

acacia_door,acacia_fence_gate,anvil,beacon,bed,birch_door,birch_fence_gate,black_glazed_terracotta,blue_glazed_terracotta,board,bookshelf,brewing_stand,brown_glazed_terracotta,brown_mushroom,brown_mushroom_block,cactus,cake,carpet,cauldron,chest,coal_block,cobblestone_wall,concrete,concretepowder,crafting_table,cyan_glazed_terracotta,dark_oak_door,dark_oak_fence_gate,deadbush,diamond_block,double_plant,dragon_egg,emerald_block,enchanting_table,end_crystal,end_portal_frame,end_rod,ender_chest,fence,fence_gate,flower_pot,frame,furnace,glass,glass_pane,glowstone,gold_block,gray_glazed_terracotta,green_glazed_terracotta,hay_block,iron_bars,iron_block,iron_door,iron_trapdoor,jungle_door,jungle_fence_gate,ladder,lapis_block,leaves,leaves2,light_blue_glazed_terracotta,lime_glazed_terracotta,lit_pumpkin,magenta_glazed_terracotta,melon_block,mob_spawner,monster_egg,nether_brick_fence,noteblock,orange_glazed_terracotta,painting,pink_glazed_terracotta,pumpkin,purple_glazed_terracotta,red_flower,red_glazed_terracotta,red_mushroom,red_mushroom_block,redstone_block,sapling,sealantern,shulker_box,sign,silver_glazed_terracotta,skull,slime,snow_layer,sponge,spruce_door,spruce_fence_gate,stonecutter,tallgrass,trapdoor,trapped_chest,vine,waterlily,web,white_glazed_terracotta,wooden_door,wool,yellow_flower,yellow_glazed_terracotta

Miscellaneous:

apple,appleenchanted,arrow,baked_potato,beef,beetroot,beetroot_seeds,beetroot_soup,blaze_powder,blaze_rod,bone,book,bowl,bread,brick,carrot,carrotonastick,chicken,chorus_flower,chorus_fruit,chorus_fruit_popped,chorus_plant,clay_ball,clownfish,coal,cooked_beef,cooked_chicken,cooked_fish,cooked_porkchop,cooked_rabbit,cooked_salmon,cookie,diamond,dragon_breath,dye,egg,emerald,emptymap,enchanted_book,experience_bottle,feather,fermented_spider_eye,fish,flint,ghast_tear,glass_bottle,glowstone_dust,gold_ingot,gold_nugget,golden_apple,golden_carrot,gunpowder,iron_ingot,iron_nugget,leather,lingering_potion,magma_cream,melon,melon_seeds,mushroom_stew,muttoncooked,muttonraw,nether_wart,netherstar,paper,poisonous_potato,porkchop,potato,potion,prismarine_crystals,prismarine_shard,pufferfish,pumpkin_pie,pumpkin_seeds,quartz,rabbit,rabbit_foot,rabbit_hide,rabbit_stew,reeds,rotten_flesh,salmon,shulker_shell,slime_ball,speckled_melon,spider_eye,splash_potion,stick,string,sugar,wheat,wheat_seeds

Tools:

activator_rail,boat,bow,bucket,camera,chain_command_block,chainmail_boots,chainmail_chestplate,chainmail_helmet,chainmail_leggings,chest_minecart,clock,command_block,command_block_minecart,comparator,compass,daylight_detector,detector_rail,diamond_axe,diamond_boots,diamond_chestplate,diamond_helmet,diamond_hoe,diamond_leggings,diamond_pickaxe,diamond_shovel,diamond_sword,dispenser,dropper,elytra,ender_eye,ender_pearl,fireball,fishing_rod,flint_and_steel,golden_axe,golden_boots,golden_chestplate,golden_helmet,golden_hoe,golden_leggings,golden_pickaxe,golden_rail,golden_shovel,golden_sword,heavy_weighted_pressure_plate,hopper,hopper_minecart,horsearmordiamond,horsearmorgold,horsearmoriron,horsearmorleather,iron_axe,iron_boots,iron_chestplate,iron_helmet,iron_hoe,iron_leggings,iron_pickaxe,iron_shovel,iron_sword,lead,leather_boots,leather_chestplate,leather_helmet,leather_leggings,lever,light_weighted_pressure_plate,minecart,nametag,observer,piston,portfolio,rail,redstone,redstone_lamp,redstone_torch,repeater,repeating_command_block,saddle,shears,snowball,spawn_egg,sticky_piston,stone_axe,stone_button,stone_hoe,stone_pickaxe,stone_pressure_plate,stone_shovel,stone_sword,structure_block,tnt,tnt_minecart,torch,totem,tripwire_hook,wooden_axe,wooden_button,wooden_hoe,wooden_pickaxe,wooden_pressure_plate,wooden_shovel,wooden_sword