This document has been transferred to wikidot and will no longer be updated!

Click me to visit the wikidot page!

----TABLE OF CONTENTS

----Introduction

This document is pending a transfer to wikidot.

This command guide is designed to allow modders to better understand the new aiscript commands created by Neiv and iquare. For the stock AI command list, refer to Nekron's document here.

Many of these commands are staples of high-end AI in the modern era. It is my hope that by writing this guide and shedding more light on the use cases of these new commands, more modders will pay closer heed to the designs of their AI and find more ways to make compelling missions in SCBW. If you have any questions, feel free to contact me.

Other resources include the expanded unitdef.txt file (for use with PyAI) and the notepad++ highlighter (automatically associates with .asc3 files, just rename your scripts with that extension).

-Pr0nogo
pronogo@hotmail.com, Pr0nogo#1940 on discord

----AI updates

The aiscript extension plugin, `aise`, features a swath of new commands by both Neiv and iquare. In addition to these new commands, certain core AI behaviors have been optimized and behave differently than in vanilla StarCraft.

The request log

StarCraft's AI can only process one request (build, train, upgrade, tech) at a time. In vanilla SC, an AI will wait to satisfy its current request before proceeding to new ones, which often leads to scripts 'hanging' on a request it cannot fulfill. With aise, AI will immediately discard requests that it cannot fulfill, and re-add them every few seconds. This feature is colloquially referred to as a 'request recycler', and ensures that AI will not freeze on requests it can't satisfy.

Limits and bugs

The following miscellaneous limits and bugs have been resolved in some way by aise:

Thread limit,

previously 100, now at least 8,000 (more may cause issues, and creating more than 100 threads in a single frame will cause issues)

Train bug,

explanation in Nekron's AI document, now considers units in eggs, cocoons, and archon mergings when checking if a request is satisfied

Guard crash,

explanation in Nekron's AI document, fixed completely

Calls,

previously more than 1 used in the same frame could crash, now works like a traditional call stack (can have multiple calls per player running simultaneously, can stack calls within calls)

Script file size limits,

previously 65kb, now aise commands (and goto) are not limited to jumping inside the first 65kb of space; this means that only vanilla commands are limited to 65kb

----attack_to (by Neiv)
attack_to (X1 Y1) (X2 Y2)
"Tells the AI to prepare attacks in %1region, targeting the enemies closest to %2region"

Notes:
1) Using the latest version of SCMDraft allows users to view pathfinder regions.
2) Coordinates always go to the center of the region they belong to.
3) Users can use locations with the syntax Loc.n, where n is the location ID. Location ID is zero-based, and `63` is Anywhere. Enabling the debug view in SCMDraft and deleting or placing a location will display its accurate location ID.
4) iquare has created a version of attack_to called **attack_to_deaths**, which behaves identically to attack_to, but takes unit deaths as an argument for prep and attack regions. This is used in conjunction with plugin data that marks a region with a unit death, allowing more dynamic targeting.

Example use cases:

Regions are taken from Rebel Yell 10 - The Hammer Falls

attack_to (3566 1135) (3312 3535)
the AI will prepare its attack inside the white base and target the region with the player's command center.

attack_to (1677 1102) (3087 3312)
the AI will prepare its attack near an empty expansion and target the region above the player's command center, where they are expected to build defenses.

attack_to Loc.3 Loc.63
the AI will prepare its attack in location 3 (called `Staging 1` in the editor) and targets the region in the center of the map (Anywhere).

deaths 8 Set 4910 0 blank
deaths 8 Set 4719 1 blank
deaths 8 Set 6096 2 blank
deaths 8 Set 4912 3 blank
attack_to_deaths 0 8 1 8 2 8 3 8
the AI will use player 8 deaths of marine and ghost for its prep region, and vulture and goliath base for its attack region. Ergo, the AI will prepare its attack at region 4910 4719 and select a target closest to region 6096 4912.

----attack_timeout (by Neiv)
attack_timeout time
"Sets the maximum attack preparation time to %1time"

`timer` specifies the max prep time in seconds.

Example use cases:

attack_add 6 carrier
attack_prepare
attack_timeout 360
attack_do
attack_clear

the AI will have a higher max prep time than default, increasing the likelihood that it fulfills the attack request.

attack_add 4 marine
attack_prepare
attack_timeout 30
attack_do
attack_clear

the AI will have a shorter max prep time than default, forcing the AI to commit to the attack very soon. No different than using wait 720 and quick_attack.

----issue_order (by Neiv)
issue_order orderId quantity unitId (src_x src_y ~ src_r) (tgt_x tgt_y ~ tgt_r) tgtUnitId flags
"Issues %1orderId to %2quantity %3unitId in %4srcArea, targeting %6tgtUnitId in %5tgtArea, using %7flags for additional instructions"

`order` specifies the order that will be given to the unit.
`count` specifies how many units will at most be ordered. Use `9999` for all units.
`unit_id` specifies which unit type will be ordered. `229` has special meaning of "any unit".
`src_x`, `src_y` specify the point at which units to be ordered are searched, and `src_r` specifies the search radius. The search area is square shaped, like locations.
`tgt_x`, `tgt_y` specify the target point. If targeting units instead of terrain, `tgt_r` will specify the search radius for matching target units. See note 1 for more.
`tgt_unit_id` specifies the acceptable unit ID for targets, when targeting units. Use `Any` for "any unit".
`flags` specifies some extra settings. They can be combined with '|'. See note 2 for more.

-Enemies: Target enemy units
-Own: Target own units
-Allied: Target allied units (neutral units are always considered allied, even if they were owned by enemy and then given to a neutral player)
-SingleUnit: Target only a single unit, even if there are multiple matches
-EachAtMostOnce: Target each target at most once. If there are more units in the source area than there are targets, the remaining source units will not be issued an order.

Additionally, the following unit building orders are supported, and they take the unit type to be built in `tgt_unit_id`. See note 3 for more.

-25: Drone build
-30: SCV build
-31: Probe build
-36: Place addon
-42: Unit morph
-43: Building morph

Notes:
1) Users can use locations with the syntax Loc.n, where `n` is the location ID. Location ID is zero-based, and `63` is Anywhere. Enabling the debug view in SCMDraft and deleting or placing a location will display its accurate location ID. If using a location, `src_r` can still be used as a square radius around the location.
2) Enabling any of the first 3 flags makes the ordered units to target units instead of ground. If no valid targets can be found in the area, then the command will not do anything.
3) When using any order that places buildings, such as 'SCV Build' or 'Land', the coordinates will not automatically be snapped to the building tile grid. The correct coordinates for placement are in the middle of unit's placement box. For example, building a command center at the top-left corner of the map would use coordinates 64, 48, as the placement box is 4x3 tiles (128x96 pixels). Locations appear to work correctly, as long as they've been snapped to the tile grid and have not moved through triggers.

Example use cases:

issue_order BuildingMorph 4 creep_colony Loc.39 Loc.39 sunken_colony 0

4 creep colonies at Location 39 will morph to sunken colonies if the player has enough resources.

issue_order Land 1 command_center Loc.5 Loc.2 Any 0

1 command center at Location 5 will land at Location 2 if there is sufficient space to land.

issue_order MoveToInfest 2 queen Loc.9 Loc.63 command_center Enemies|SingleUnit

2 queens at Location 9 will infest a command center at Anywhere if it is infestable.

----idle_orders (by Neiv)
idle_orders order rate maxPerTarget userUnitId radius tgtUnitId priority flags
"Adds %1orderId order to %3maxPerTarget %4userUnitId every %2rate frames at %7priority, targeting %6tgtUnitId unit types within %5radius from the caster, using %8flags for additional instructions"

`order` specifies the order that will be given to the unit.
`user_unit_id` specifies the unit that will receive the order.
`target_unit_id` specifies the unit that will be targeted by the user unit. Can also be `Any`, `Group_Men`, `Group_Buildings`, or `Group_Factories`. See note 3 for more.
`rate` specifies how often the AI will (try to) send a unit to use the order. Specified in frames.
`max_per_target` specifies how many units may simultaneously target a same unit with a idle order.
`radius` is the maximum distance in pixels for which the AI will send its units from their idle position.
`priority` specifies the priority. Higher priority is more important. Does not compete with stock AI commands such as `build` and `train`.
`flags` specifies various targeting flags. They can be combined with '|'. Underlined flags are 1.16.1 only.

-NotEnemies: Will not accept enemies as targets. Targeting enemies is enabled unless disabled with this.
-Own: Will accept own units as targets
-Allied: Will accept own units as targets
-Unseen: Will accept unseen targets (i.e. uses maphacks instead of targeting only things that are visible to the player)
-Invisible: Will accept undetected targets, and use the order on the ground near target.
-Remove: Instead of adding an idle order, removes one that matches the current declaration completely (excluding the `Remove` flag itself). Shows a warning if unsuccessful.
-RemoveSilentFail: Like Remove, but won't show any warnings if nothing gets removed.
-SpellEffects(effects): Target must have all of selected spell effects:
`ensnare`, `plague`, `lockdown`, `irradiate`, `parasite`, `blind`, `matrix`, `maelstrom`
-WithoutSpellEffects(effects): Target may not have any of selected spell effects.
-InCombat: Will accept targets that are targeting an enemy with an attacking order.
-Hp: Will compare hit point values for acceptable targets.
-Shields: Will compare shield values for acceptable targets.
-Health: Will compare the sum of hit point and shield values for acceptable targets.
-Energy: Will compare energy values for acceptable targets.
-Deathrattle: Will only receive the order when brought to 25% of max health (hp + shields) or 50% of health at the time the script executes the command, whichever value is smaller. Does not use the `count` or `rate` fields. Only executes if the unit was amidst executing a non-Deathrattle idle order. See note 5 for more.
-Self: Will compare following flags with its own state.
-Order: Target must have the selected order.
-Targeting: Target must be targeting one of the selected classes:
`Own`, `Enemy`, `Ally`, `Nothing` -- use `OtherUnit` for the user of the idle_order
-UnitFlags: Target must have all of the selected unit flags:

`Building`, `Addon`, `Air`, `Worker`, `Turret`, `FlyingBuilding`, `Hero`, `Regeneration`, `Animated Overlay`, `Cloakable`, `DualBirth`, `Powerup`, `ResourceDepot`, `ResourceContainer`, `Robotic`, `Detector`, `Organic`, `RequiresCreep`, `Unk40000`, `RequiresPsi`, `Burrower`, `Spellcaster`, `PermanentlyCloaked`, `Carryable`, `IgnoreSupplyCheck`, `MediumOverlays`, `LargeOverlays`, `CanMove`, `FullAutoAttack`, `Invincible`, `Mechanical`, `UnkProducesUnits`

-WithoutUnitFlags: Target must not have any of the selected unit flags.

-RandomRate: Replaces `rate` with a random number between a specified min and max value. Syntax is *RandomRate(min max)*

-Hangar: Will compare subunit/ammo count for acceptable targets (Carrier, Reaver, Vulture, Nuke Silo).

-Count: Target must be adjacent to the specified number of units. Uses AtLeast/AtMost/Exactly. Syntax is *Count(operator quantity range playerId)*

-Cargo: Will compare cargo amount for acceptable targets, e.g. *Cargo(GreaterThan 3)*

-TileFlags: Target must be on the following kinds of tile:

`Walkable`, `Unbuildable`, `Creep`, `Ramp`

-WithoutTileFlags: Target must not be on the selected kinds of tile.

Notes:

1) The target picking isn't perfect. It prefers the closest unit it can find, and the logic works fine when the all order users/targets are roughly in same area, but if there are two or more separate groups of both, it may ignore one of the groups completely.

2) To disable the default AI behavior for the player, you can use DisableBuiltIn in `order` parameter (And leave everything else 0)

3) If `target_unit_id` is set to `Any`, the units just try to target anything they can even if its impossible, like trying to cast hallucination on buildings.

4) Comparisons for Hp, Shields, Health, and Energy can be `LessThan`, `GreaterThan`, `LessThanPercent`, or `GreaterThanPercent`. Comparing shields will never be true if the unit has no shields.

5) Deathrattle piggybacks off of previously-executed idle_orders commands and does nothing by itself. Think of it as a built-in retaliatory system for idle_orders spellcasts.

Example use cases:

idle_orders dark_swarm 64 1 defiler 1200 zergling 60 Own|Allied|NotEnemies

every 64 frames, 1 idle defiler will attempt to dark swarm own or allied zerglings within 1200 pixels of the idle defiler.

idle_orders attack 64 3 infested_terran 1200 Group_Buildings 60 SpellEffects(Plague)

every 64 frames, up to 3 idle infested terrans will attempt to attack any enemy unit afflicted with plague within 1200 pixels of the idle infested terran.

idle_orders MoveToInfest 64 1 h_queen 2400 command_center 99 Hp(LessThanPercent 49)

every 64 frames, 1 matriarch will attempt to infest enemy command centers that have less than 49% health within 2400 pixels of the idle matriarch.

idle_orders PsiStorm 0 0 high_templar 5000 Group_Men, 99 Deathrattle

any high templar executing an idle order will attempt to retaliate if brought below the hp threshold by casting psi storm on enemy men within 5000 pixels.

idle_orders Feedback 480 2 dark_archon 1280 high_templar 70 Order(PsiStorm)

every 480 frames, up to 2 idle dark archons will attempt to feedback hostile high templar that have been ordered to cast psi storm within 1280 pixels.

idle_orders Consume 500 1 defiler 240 zergling Own|NotEnemies|Hp(LessThanPercent 25)|Self(Energy(LessThanPercent 50))

every 500 frames, 1 idle defiler will attempt to consume own zerglings with less than 25% life in 240 pixels if the caster has less than 50% energy.

idle_orders DisableBuiltin 0 0 0 0 0 0 0

all standard idle (not retaliatory) spellcasts are disabled

----deaths (by Neiv)
deaths playerId operator quantity unit block
"%2operator (read/write) %3quantity %1playerId's %4unitId deaths, jumping to %5block if true"

`playerId` can be 0-21. See note 1 for more.
`operator` can be AtLeast, AtMost, or Exactly for reading; Add, Subtract, Set, or Randomize for writing. Read comparisons can have the suffix _Call to call the specified block if true, or the suffix _Wait to wait for the comparison to be true before continuing script execution (does not jump or call).
`quantity` specifies the number of kills to compare or write.
`unit` specifies the units to use when comparing or writing. Accepts multiple entries with `|`.
`block` specifies the block in the aiscript to jump to if the comparison is true. Not used if writing data.

Notes:
1) All `player` fields are zero-based and accept the following values:
0-11 - Players 1-12
13 - Current player
14 - Foes
15 - Allies
17 - All players
18-21 - Force 1-4

Example use cases:
deaths 0 Exactly 1 probe phaseTwo
if player 1 has suffered exactly 1 death of probe, jump to block phaseTwo

deaths 13 Set 119 arbiter phaseOne
set current player's deaths of arbiter to 119 (and do not jump)

deaths 4 Randomize 3 fleet_beacon blank
randomize player 5's fleet beacon deaths within a range of 0-2

----bring_jump (by iquare)
bring_jump playerId operator quantity unit area block
"%2operator (read/write) %1playerId's %3quantity of %4unitIds in %5area, jumping to %6block if true"

`playerId` can be 0-21.
`operator` can be AtLeast, AtMost, or Exactly (includes _Call and _Wait).
`quantity` specifies the number of units to compare for.
`unit` specifics which unit to compare for. Can be combined with "|" where a matching quantity of any combination of the specified units will be a true comparison.
`region` specifies the region (or location with `Loc.n`) to search for the units.
`block` specifies which block in the aiscript to jump to if the comparison is true.

Example use cases:

bring_jump 4 Exactly 0 infested_command_center (367 4334) requestHatchery

if player 5 brings exactly 0 infested command centers to the specified region, jump to block.

bring_jump 13 AtLeast 1 Group_Factory Loc.63 defendFactory

if current player brings at least 1 factory-class structure to anywhere, jump to block.

bring_jump 18 AtLeast 30 goliath|siege_tank|science_vessel Loc.63 adaptMech

if force 1 brings at least 30 of the selected units to anywhere, jump to block.

----create_script (by Neiv)
create_script block playerId area town resarea
"Runs %1block for %2playerId in %3area, attaching to %5resarea
%4town: 0 (no town) or 255 (current town)"

`block` is the thread that will run in the newly-created thread
`playerId` specifies the player that owns the thread. Only accepts 0-7, and 255 for current player.
`area` specifies the precise coordinates the thread will be run at. Radius is in pixels. Can also accept locations through Loc.n~R. The center of this value acts as the town center if a town is created in the block.
`town` specifies whether to use no town (0) or the current town (255). Any other value does nothing. Using *town 255* in a thread that has no attached town is identical to using *town 0*.
`resarea` specifies which resarea the thread will attach to. A value of 0 tells the script to not attach to any specific resarea, but will still mine from nearby resources if allowed.

Notes:
1) To inherit values from the current thread, use the following values (this is identical to multirun):
create_script block 255 (65534 65534) 255 0

Example use cases:

create_script expand3 255 (1251 3894~24) 0 14
create a thread for current player at the specified coordinates; use code in expand3, don't use current town, and attach to resarea 14.

create_script cannonRush 4 Loc.24 255 0
create a thread for player 5 using the bounds of location 24; use code in cannonRush, use current town, and don't attach to any resarea.

----player_jump (by iquare -- 1.16.1 only)
player_jump string block
"Compares player names in-game to %1string, jumping to %2block if reading a true comparison"

`username` is the text to match. Must be a complete match, not case sensitive.
`block` specifies which block in the aiscript to jump to if the comparison is true.

Example use cases:

player_jump nekron killnekron

if the player's username matches `nekron`, jump to block.

----**kills** (by iquare -- 1.16.1 only, see note 2)
kills player1 player2 operator quantity unit block
"%3operator (read/write) %1player1's %4quantity kills of %2player2's %5unitId, jumping to %6block if true"

`player1` specifies the player to compare kill counts with.
`player2` specifies the player to compare kill counts against.
`operator` can be AtLeast, AtMost, or Exactly for reading (includes _Call and _Wait); Add, Subtract, Set, or Randomize for writing.
`quantity` specifies the number of kills to compare or write.
`unit` specifies the units to use when comparing or writing. Accepts multiple entries with `|`.
`block` specifies which block in the aiscript to jump to if the comparison is true.

Notes:
1) As the kills table is several times larger than the deaths table, it can be used instead of deaths for variables.
2) Though the bulk of this command's functionality is only usable in 1.16.1, you can still read values you've written elsewhere when writing scripts for 1.2. You will not be able to track kills in-game.

Example use cases:

kills 1 17 AtLeast 10 mutalisk|devourer|guardian buildAir

if player 2 kills at least 10 of the specified units owned by any player, jump to block.

kills 18 19 AtLeast 2 Group_Buildings warnPlayer

if force 1 kills at least 2 buildings owned by force 2, jump to block.

----supply (by iquare)
supply playerId operator quantity supplyType unitId race block
"%2operator (read/write) %1playerId's %3quantity %4supplyType of %6race, jumping to %7block if true
%4supplyType: Provided, Used, Max, InUnits
%5unitId: 0, unless %4supplyType set to InUnits"

`operator` can be AtLeast, AtMost, Exactly (includes _Call and _Wait), Set, Add, Subtract, or Randomize.
`supplyType` can be Provided, Used, Max, or InUnits. When writing supply, you must use Max.
`units` is used only if `supplyType` is set to InUnits. `0` otherwise. Compares supply of the specified units.
`race` can be Terran, Zerg, Protoss, Any_Total, or Any_Max. When writing, you cannot use the last two.

Example use cases:

supply 18 AtLeast_Call 54 InUnits battlecruiser Terran adaptBattlecruiser

if force 1 has at least 54 terran supply used by battlecruisers, call the block.

supply 13 Set 250 Max 0 Terran blank

sets current player's max terran supply to 250.

----time (by iquare)
time operator quantity timeType block
"Using %1operator, compare game time to %2quantity %3timeType, jumping to %4block if true
%3timeType: Frames, Minutes"

`operator` can be AtLeast, AtMost, or Exactly (includes _Call and _Wait).
`timeType` can be Frames or Minutes. Minutes are normal-speed game minutes, as with time_jump.

Example use cases:

time AtLeast 7200 Frames startScript

if at least 2.5 real minutes have passed, jump to block.

----resources (by iquare)
resources playerId operator resourceType quantity block
"%2operator (read/write) %1playerId's %4quantity %3resourceType, jumping to %5block if true
%3resourceType: Ore, Gas, Any"

`resourceType` can be Ore, Gas, or Any. When reading, `Any` requires both Ore and Gas to be true. When writing, `Any` writes to both resource types.
`operator` can be AtLeast, AtMost, Exactly (includes _Call and _Wait), Set, Add, Subtract, or Randomize.

Example use cases:

resources 14 AtMost_Call Ore 350 startHarass

if Foes have 0-350 minerals, call the block.

resources 0 Add Any 200 blank

add 200 ore and gas to player 1's resources.

----remove_[request] (by iquare)
remove_[requestType] quantity unitId townId
"removes %1quantity of %2unitId from %3townId's request log"

`requestType` can be build. More types are not yet implemented.
`requestedUnit` is the unit requested by the [requestType] command, e.g. `build 1 command_center`.
`townId` can be 0-254, as set by set_id, or 255 to use the current town. See misc commands for more info. The town id must be set even when using a value of 255.

Example use cases:

remove_build 2 nuclear_silo 0

remove 2 nuclear silo requests from town 0.

----guard (by iquare)
guard unitId area quantity retrainingCount priority
"Requests %3quantity %1unitId guards at %3area and %5priority, retraining them %*retrainingCount* times
%4deaths - 0 for infinite"

`location` can be (PosX PosY) or Loc.n.
`deaths` can be 0-255, with 255 being the max value, 1 being never remake, and 0 being remake infinitely.

Example use cases:

guard goliath (1353 4966) 1 3 40

requests 1 goliath at the specified coordinates at priority 40 and remakes the guard 3 times

----base_layout (by iquare -- 1.16.1 only)
base_layout structureId operator area quantity townId priority
"%2operator (set/remove) %3area for %4quantity %1structureId in %5townId at %6priority"

`operator` can be Set or Remove.
`area` can be (PosX PosY)~R or Loc.n~R. If all locations are occupied, the AI will build normally.
`townId` can be 0-254, as set by set_id, or 255 to use the current town. The town id must be set even when using a value of 255.
`priority` only competes with base_layout slots fo the same unitId.

Example use cases:

base_layout supply_depot Set (48 3040) 1 1 50
base_layout supply_depot Set (80 2880) 1 1 50
base_layout supply_depot Set (112 2944) 1 1 50

the AI will prioritize the three specified locations for depots in town 1

base_layout command_center Remove (384 2608) 1 13 50

the AI will no longer prioritize the specified location for command centers in town 13

----load_bunkers (by iquare)
load_bunkers area unitId unitQuantity bunkerUnitId bunkerId priority
"Requests %3unitQuantity %2unitId for %5bunkerId %4bunkerUnitIds in %1area at %6priority"

`unitId` and `unitQuantity` specify what number of which units will be staffed in the bunker.
`bunkerUnitId` specifies which bunker-type unit or structure will accept the specified units.
`bunkerId` specifies which bunker specifically will accept the specified units. e.g. the first bunker a player uses should use a `bunkerId` of 1. This value is player-specific.

Notes:
1) This command does not request units; it only orders eligible units to enter the bunker.
2) Units are eligible if they are not in a transport, bunker, attack wave, or defense force. Units that are assigned to other AI tasks or killed prior to reaching their bunker are removed from the bunker's list.
3) The `bunkerUnitId` field accepts any unitId for the purpose of mods that add bunker functionality to other unitIds.

Example use cases:
load_bunkers (1234 5678) h_marine 3 bunker 1 50
bunker at the specified point will order up to 3 eligible hero marines at priority 50

----reveal_area (by iquare -- 1.16.1 only)
reveal_area playerId area time flags
"Reveals %2area for %1playerId, lasting %3time frames and using %4flags for reveal type"

`area` can be Loc.n, coordinate, or coordinate~radius.
`time` is in frames. A value of 0 is permanent.
`flag` can be RevealFog. A proper reveal flag is being investigated but may not be possible.

Example use cases:

reveal_area 1 Loc.3 0 RevealFog

permanently reveals fog in location 3 for player 2

reveal_area 4 (1234 4321)~32 360 RevealFog

reveals fog for 15 seconds in a 32-pixel radius around the specified point for player 5

----bank_data (by iquare)
bank_data operator quantity category key block
"%1operator (read only) %2quantity of %3category's %4key, jumping to %5block if true
Use save_bank(string) and load_bank(string) to save or load external files"

`operator` is a comparison, like other jump commands (deaths, kills, bring_jump, etc).
`category` and `key` are string labels for variables.
`block` only jumps if reading a true comparison.

Notes:
1) Use *save_bank string* and *load_bank string* to save and load external files.
2) External files are stored as binary data. On 1.16.1, they are stored in a folder called `save`, in the mod's directory. On 1.2+, they are stored in Documents\Starcraft.

Example use cases:

load_bank survivorCount
bank_data AtLeast 1 map1 marineCount createMarine

load data, and if key `marineCount` in category `map1` is at least 1, jump to block.

bank_data Add 1 map1 marineCount createMarine
save_bank survivorCount

add 1 to key `marineCount` in category `map1` and save the data.

---lift_land (by iquare -- 1.16.1 only)
lift_land unitId quantity sourceArea targetArea sourceTown targetTown hpValue
"Orders %2quantity %1unitIds at %3sourceArea in %5sourceTown to lift off and land at %4targetArea, transferring to %6targetTown once landing is complete; retreats back to %3sourceArea if dropped below %7hpValue before the landing completes"

`sourceTown` and `targetTown` read values set by set_id.
`hpValue` orders the unit to retreat to its source location if its hp drops below the specified percentage.

Notes:
1) Retreating structures will embark again once repaired.
2) If destroyed while in transit, assuming the structure is rebuilt in the sourceArea, the replacement structure will also attempt to land at the targetArea.
3) Town ownership does not change until the structure lands at the targetArea.
4) Ensure you use a large enough sourceArea to pick up the structures you want, or the AI will be told to transfer nothing.

Example use cases:

lift_land command_center 1 Loc.4 (1234 4321) 1 3 50

orders one command center owned by town 1 at location 4 to move to the specified coordinates and change ownership to town 3, retreating if brought under 50% life.

----queue (by iquare)
queue quantity unit_id factory_id townId modifier area priority
"Queues %1quantity %2unitId from available %3factoryId in %4townId in %6area at %7priority."

`modifier` can be `Local` or `Global`. Local requests are town-specific.

Notes:
1) The command rechecks every 30 frames until enough slots become available, queuing as many as possible until the specified amount have been queued.
2) `townId` is only used if `modifier` is set to `Local`.
3) If `modifier` is set to `Local` and `townId` specifies a nonexistent town, the command will not fire and an error message will print.

Example use cases:

queue 3 h_marine barracks 2 local Loc.63 80

queues 3 hero marines using only town 2's barracks

----unit_name (by iquare -- 1.16.1 only)
unit_name playerId unitId area string operator
"%5operator (write only) %1playerId's names of %2unitIds in %3area to %4string"

string is a text string that replaces the stat_txt.tbl entry; use <32> for a space.
operator can be Set or Remove. Using Remove will reset the string ID used by the specified units.

Example use cases:

unit_name 0 command_center Loc.63 Dominion<32>Command<32>Center Set

sets the name of all player 1 command centers at anywhere to the specified string

----defense (by iquare)
defense quantity unitId defenseType defenseDirection1 defenseDirection2
"Adds %1quantity %2unitId to the AI's %3defenseType list when defending %4defenseDirection1 units from %5defenseDirection2 attackers"

`defenseType` can be `Build` or `Use`. Build counts as a request.
`defenseDirection` parameters can be `Ground` or `Air`.

Example use cases:

defense 1 marine build ground air

adds 1 marine to the list of units the AI builds when its ground units are attacked by air units

----replace_unit (by iquare -- 1.16.1 only)
replace_unit unitId1 unitId2
"Permanently replaces most requests of %1unitId1 with requests of %2unitId2"

Notes:
1) This command only replaces requests for commands hooked in aise. Currently, defenseuse, defensebuild, wait_train, and wait_force are not supported by this command.

Example use cases:

replace_unit goliath marine

permanently replaces all supported goliath requests with marine requests

----container_jump (by iquare)
container_jump resourceType operator quantity area block
"Using %2operator, compares %1resourceType of resource containers in %4area, jumping to %5block if reading a true comparison"

Example use cases:

container_jump Ore AtLeast_Wait 2000 Loc.4 blank
pause script execution until at least 2000 minerals are present in location 4

----misc commands
unstart_campaign (by Neiv)
"Takes no value"
Sets the script class to melee regardless of game type

under_attack (by Neiv)
under_attack value
"Sets AI defense behavior to %1value"

`value` has three valid parameters.
-0: AI will never take units from attack forces to defend.
-1: AI will take units from attack forces to defend. Default behavior.
-2: AI will almost never attack and keep all of its active units available for defense.

max_workers (by Neiv)
max_workers value
"Allows AI to have %1value workers"

`value` can go up to 255. Does not request workers, only allows/disallows them. Can be used with a value of `0` to stop area towns from requesting workers. A value of `0` will not prevent an AI from building gas structures on nearby geysers. This command can be used to supersede the global worker limit of 30, but the worker limit is reduced every time the AI expands, so it is advisable to write your desired max_workers value at the beginning of every expansion thread.

aicontrol (by Neiv; some opcodes by iquare)
aicontrol value
"Sets general AI behavior to %1value"

`value` can be one of the following:
-dont_wait_request_resources: the AI will not wait for resources to fulfill a request, recycling them as it would any other request if it doesn't have the resources to satisfy it. Turn off with `wait_request_resources`.
-dont_build_gas: the AI will not build gas extraction structures automatically. Accepts requests. Turn off with `build_gas`.
-no_retaliation: the AI's spellcasters will not retaliate with spells when attacked. Also disables aggressive casts when spellcasters are added to attacks. Turn off with `retaliation`. **1.16.1 only.**
-dont_focus_disabled: the AI's units will not focus enemy units disabled by maelstrom or lockdown. **1.16.1 only.**
-focus_disabled: inverse of above. **1.16.1 only.**

wait_rand (by iquare)
wait_rand minFrame maxFrame
"Pauses script execution for a random amount of frames between %1minFrame and %2maxFrame"

`value1` is the minimum number of frames for which the script will pause, while `value2` is the maximum.

random_call (by iquare)
random_call chance block
"Calls %2block %1chance/256 times"

`chance` is out of 256, so 128 would be 50%

upgrade_jump (by iquare)
upgrade_jump playerId operator upgradeId upgradeQuantity block
"%2operator (read/write) %1playerId's %3upgradeId %4upgradeQuantity, jumping to %5block if true"

Can also write upgrade data with Set, Add, Subtract, and Randomize operators. Can use _Call and _Wait for reading.

tech_jump (by iquare)
tech_jump playerId operator techId techLevel block
"%2operator (read only) %1playerId's %3techId %4techLevel, jumping to %5block if true"

Can use _Call and _Wait for reading. 0 for available, 1 for researched.

attack_rand (by iquare)
attack_rand minQuantity maxQuantity unitId
"Adds a random amount of %3unitId between %1minQuantity and %2maxQuantity"

set_id (by iquare)
set_id variable
"Assigns %1variable to the current town"

`variable` can be 0-255. Must be unique across all players. Commands interacting with town ids can be used in scripts owned by players that don't own the specified town, e.g. a script owned by player 1 can modify the requests of a script owned by player 2 with remove_[request].

print (by iquare)
print string
"Prints %1string for all players, also playing transmission.wav"

attacking (by iquare)
attacking operator block
"%1operator (read only) current AI's attacking state, jumping to %2block if true"

`operator` can be True, False, True_Call, False_Call, True_Wait, and False_Wait.

if_attacking (by Neiv)
if_attacking block
Jump to the desired block if the AI is preparing an attack
Outmoded by `attacking`.

ping (by iquare -- 1.16.1 only)
ping PosX PosY color
"Pings %1posX %2posY on the minimap for all players, using %3color"

Unlike other commands, ping has two separate location fields, for x and y coordinates.
`color` is numerical, where 0 = red, 1 = blue, etc. Currently, colors past 15 (cyan by default) do not work, and custom colors via a modified tminimap.pcx are not supported.

unit_avail (by iquare)
unit_avail playerId operator availability unitId block
"%2operator (read/write) %1playerId's %3availability of %4unitId, jumping to %5block if true
%3availability: Enabled, Disabled"

`operator` can be Exactly, Set, or Randomize.
`availability` can be Enabled or Disabled.

Example use cases:
unit_avail 0 Set Enabled siege_tank blank
sets player 1's siege tank availability to enabled

tech_avail (by iquare)
tech_avail playerId operator, techId, availability, block
"%2operator (read/write) %1playerId's %3techId %4availability, jumping to %5block if true"

`operator` can be Exactly, Set, or Randomize.
`level` can be 0 for available or 1 for researched.

remove_creep (by iquare -- 1.16.1 only)
remove_creep area
"Instantly removes creep in %1area"

`area` can be Loc.n or PosX PosY~R.

max_build (by iquare -- 1.16.1 only)
max_build townId unitId quantity
"Sets the maximum %2unitId that can be requested by %1townId"

Used to limit AI to a max number of structures, particularly useful when requesting supply structures in expansions as these requests are cloned in the main town.