Star Trek: Armada ODF Directives

ODF files of the game *Star Trek: Armada* consist of lines of so-called directives, that influence, how an element of the game behaves, e.g. how much health a specific unit does have. Here is a rather lengthy list of directives, with their different uses and effects.

Meta Commands

Some commands/directives are actually meta commands, as they don't have any effect on their own, but only import other files referenced, which in turn are causing the actual effects. They work with any kind of file, but don't always make sense, depending on the type of ODF/object they are used by.

Command/Directive	Effect/Usage
`#include "filename.odf"`	Makes the game include any directives that are not already included in the current file, from another file. Many weapons use this so that they can have a different name and sprite but have the same damage and fire rates etc... Simply put, it behaves as if all the text from the referenced file is placed at this location, instead of the *#include* command. This works cascading. So if an included file uses *#include* as well, these files are also de-referenced during loading. Do not create cyclic references!
`baseName = "my_file_without.sod"`	Specifies which SOD file is used for this unit. This is only useful when the SOD is named differently than the unit's ODF file. When using this directive in an ODF, this ODF inherits all the GUI properties of the ODF designated by the *baseName* directive as well as the SOD (i.e. button, wireframe).
`physicsFile = "my_file.odf"`	Specifies which physics file is used for this ship.

Notation of Names and Values

For reasons of shortening the descriptions below, here is a list of short forms and their explanations used in directives descriptions and depictions:

Short Form	Meaning
y/n value	Variables that are of a binary nature, meaning, the set value means either *true* or *false*, are abbreviated with *y/n value*. The actual, literal values are either `1` (for *yes*, *true*, or *is active*) or `0` (for the opposite).
integer	Integer values are non-fraction numbers. So `-1` is allowed, `0`, too, as well as `50` or `100`. But `2.5` is not.
float	Floating point numbers are fractions of numbers. They come in two basic flavors: with tailing `f` and without it. So `1.5` is OK but means pretty much the same as `1.5f`. It is not entirely clear whether the *f* is actually ever needed. The original files use both types, with and without it. If you somehow notice, that a directive does something else, when adding the `f`, than when omitting it, feel free to point that out to the author. In general you can write floats with a leading `0` also without that *0*, e.g. instead of `0.5` you can write also `.5`. While this is possible, it is discouraged for reasons of readability.

Short Form	Meaning
percentage fraction	This is a *float* whose value range may be between and including *0* and *1*. While `0` or `0.0` depicts *0 %*, `1` or `1.0` depicts *100 %*. So *50 %* is written as `0.5`.
RGB value	These are actually **space separated** triplets of *percentage fractions*, defining a so-called RGB color. RGB colors consist of three values, **R**ed, **G**reen and **B**lue, always in exactly this order. These colors mixed in these amounts result in a specific color in the RGB space. For example a screaming intense red would be depicted as `1.0 0.0 0.0` (100 % red, 0 % green and 0 % blue) while `1.0 0.3 1.0` (100 % red, 30 % green and 100 % blue) on the other hand looks like Magenta color.
string	A string is some sort of sequence of characters, like the name of another file, a classLabel or a free text, such as a tooltip. Strings are enclosed by `""` (two inch signs, no typographic quotation marks).

Type Keys

Key	Meaning
C	directive for commands
M	map object directive
O	other miscellaneous directive
P	physics directive
SH	ship directive
ST	station directive
SW	special weapon directive
W	weapons directive

Ship and Station Directives

Directive	Description	Type
`aiName = "`*`CraftProcess`*`"`	This is an AI parameter that helps the AI evaluate targets. It tells the AI what type of ship it is so it can prioritize the targets and function correctly when told to seek and destroy etc... Allowed string values are: `CraftProcess` `SalvageProcess` `StarbaseProcess`	SH, ST
`ambientSound = "`*`omega`*`.wav"`	This defines the file to be used for the sound a map object makes. The volume changes depending on the proximity to the object.	O
`animation = `*`1`*	Turns the animation in a SOD file on when the unit is constructed. 1 means play once then stop. 2 means play looping infinitely.	SH, ST
`attackPower = `*`0.5`*	This is used by the AI to evaluate how strong the unit is. The higher the number that follows is, the more powerful the AI considers it and the more ships it will send against it when fighting it.	SH, ST
`avoidanceClass = `*`555`*	How much should we try to avoid this ship? The higher the number the more space we assume is around it that we want to avoid.	SH, ST

Directive	Description	Type
boardingPartyStrength = *12.5*	The strength of boarding parties when they transport aboard an enemy ship. The higher the number the more powerful the party is and the more likely they are to overcome the target ships crew. The values of both races, attacker and attacked, will influence how much crew the attacked will lose for each crew beamed over to it. Example: Attacker has 3, attacked has 2. Each beam over of 5 crewmen (standard beam-in party size) costs the attacker 5 crewmen on his own ship, but will kill 7.5 on the attacked ship. So in order to take over a 100 crewmen target ship in this constellation, an investment of 70 crewmen will be required, ending up in 5 own crewmen remaining on the target ship, eventually.	SH
buildAnimation = *1*	y/n value, are there any build functions at this station?	SH, ST
buildHardpoint = "*hp01*"	Defines the hardpoint at which building takes place at this station or unit.	SH, ST
buildItem*0* = "*odf_name*"	This tells the game that whatever ODF name follows this statement can be built from this unit or station, the *0* can change into any number, depending on how many items are on the list.	ST
buildTime = *123*	The amount of time in seconds it takes to build this unit.	SH, ST
canAttack = *0*	This y/n value tells the game whether or not this ship can engage in combat. By default ships like construction ships cannot attack, with a couple exceptions. It determines if the button for an action related to this is present when you click on a unit with this line in it.	SH, ST
crewCost = *50*	The amount of crew required for the ship or station, to be built. The amount is removed from the players stock, the moment the construction starts.	SH, ST
crewHitPercent = *2.5f*	This determines the chances that the crew will be hit by a weapon, once the shields are completely down. Must add up to 100 (%) along with the following other values: *hullHitPercent*, *lifeSupportHitPercent*, *sensorsHitPercent*, *shieldGeneratorHitPercent* and *weaponsHitPercent*.	SH, ST
criticalTargetHardpoints = "hp*12*"	This designates the hardpoint for critical hits. After this directive a series of hardpoint names is given, that are the firing points for weapons. If the critical hit hardpoint has no hitpoints left, the ship or station is destroyed.	SH, ST
damagedScan = *150f*	When the sensors are destroyed/deactivated, how far does the ship scan, or „see“?	

Directive	Description	Type
dilithiumCost = *1000*	The amount of dilithium required for the ship or station to be built. The amount is removed from the players stock, the moment the construction starts.	SH, ST
dilithiumTransferRate = *15.0*	Sets the rate at which the resources are transferred between the freighter and the resource station.	SH, ST
distanceBelowGrid = *30*	How far below the grid should we move this station?	ST
enginesCrewLoss = *5.0f*	This is the percentage of how much of the crew dies, the moment the engines sub-system gets destroyed.	SH, ST
enginesHitPoints = *12*	Number of hitpoints the engine sub-system has. Once damage exceeds this number it becomes disabled.	SH, ST
enginesRepairTime = *1.0f*	The amount of time in seconds that it takes to repair one hitpoint of damage to the engines sub-system. This is modified by the crew status. If the crew status is yellow, it usually is slower. In status red it is usually even slower. But it can also be totally different (e.g. Borg always repair at the same rate).	SH, ST
enginesTargetHardpoints = "hp*12*"	This designates the hardpoint for the engine sub-system. After this directive a series of hardpoint names is given, that are targets for weapons.	SH, ST
fireball = "*xfirebal*"	This specifies the explosion fireball that the ship uses when it blows up.	SH, ST
footprintBuffer = *20.0f*	Sets the footprint that the unit leaves when placing it for construction.	ST
freighterName = "*kfreight*"	Defines what freighter is used as a resource gatherer for a refinery. This ship is automatically built along with the mining station.	ST
hullHitPercent = *62.0f*	This determines the chances that the hull will be hit by a weapon, once the shields are completely down. Must add up to to 100 (%) along with the following other values: *crewHitPercent*, *lifeSupportHitPercent*, *sensorsHitPercent*, *shieldGeneratorHitPercent* and *weaponsHitPercent*.	SH, ST
hullTargetHardpoints = "hp*12*"	This designates the hardpoint for the hull sub-system. After this directive a series of hardpoint names is given, that are the firing points for weapons. If the hull hardpoint's hitpoints are depleted, the ship or station gets destroyed.	SH, ST
intrinsicValue = *1.5*	This tells the AI how important of a target this unit is. The higher the number the more likely the AI is to attack it.	SH, ST
is_starbase = *0*	Is it a starbase? y/n value.	ST
isHero = 1	Specifies that this ship is a special hero unit, y/n value.	SH, ST

Directive	Description	Type
isTranswarpGate = *1*	Specifies that this station is a Transwarp Gate, y/n value.	ST
lifeSupportCrewLoss = *3.0f*	This is the percentage of how much of the crew dies, the moment the life support sub-system gets destroyed. The only exception is life support. Life support goes by a percent of the crew that dies for every second that the life support is disabled. See also lifeSupportLoss.	SH, ST
lifeSupportHitPercent = *2.5f*	This determines the chances that the life support sub-system will be hit by a weapon, once the shields are completely down. Must add up to 100 (%) along with the following other values: *crewHitPercent*, *hullHitPercent*, *sensorsHitPercent*, *shieldGeneratorHitPercent* and *weaponsHitPercent*.	SH, ST
lifeSupportHitPoints = *12*	Number of hitpoints this system has, once damage exceeds this number it becomes disabled.	SH, ST
lifeSupportLoss = *2.0f*	This is the percentage of how much of the crew dies every second, while the life support sub-system is inoperable. Important: If the percentage x residual crew amount drops below 1, it is always 1 crewman per second. So a ship without life support can totally „bleed out“.	SH, ST
lifeSupportRepairTime = *0.1f*	The amount of time in seconds that it takes to repair one hitpoint of damage to the life support sub-system. This is modified by the crew status. If the crew status is yellow, it usually is slower. In status red it is usually even slower. But it can also be totally different (e.g. Borg always repair at the same rate).	SH, ST
lifeSupportTargetHardpoints = "hp*12*"	This designates the hardpoint for the life support sub-system. After this directive a series of hardpoint names is given, that are the firing points for weapons. If the life support hardpoint’s hitpoints are depleted, the ship or station will lose crew over time. This progresses faster, the more crew is actually still aboard.	SH, ST
mapIcon = "*map_worm*"	This specifies what icon to use for the minimap to display a planet.	O
maxHealth = *100*	Max Shield Strength	SH ST
maximumUpgrades = *6*	Defines the max number of officer upgrades the station can build.	ST
maxRoll = *12*	This defines the roll of a craft. Roll is side to side tilting when turning, also called „banking“. Low values makes the ship always fly flat and level even when turning.	SH
maxSpecialEnergy = *123*	Specifies the total amount of special energy this unit can have when fully charged.	SH, ST
moveBelowCombatArea = *0*	Should we move this station below the grid. This is a y/n value.	ST

Directive	Description	Type
numberOfWorkerBees = *5*	Defines how many bees are used to build it.	SH
officerCost = *10*	The amount of officers required for the ship or station to be built. The amount is removed from the players stock, the moment the construction starts. Specialty here: If the ship or station gets destroyed, the used up officers will get freed up, too. So it's more like reserving the cost, than subtracting it from the stock.	SH, ST
officerGain = *20*	Tells how many officers are gained when this upgrade is built.	ST
possibleCraftNames = "name_1" "name_2"	Lists all the possible names for the craft or station. and each one is surrounded by quotes and separated with a space	SH, ST
podHardpoints = "*hp01*"	Defines the hardpoint at which pod building takes place at this station or unit.	SH, ST
race = "*borg*"	Names the race that can build the station or ship	SH, ST
rangeScan = *123*	Defines how far this unit can scan (line of sight).	SH, ST
repairFacility = *0*	Specifies whether this station is a repair facility, y/n value.	SH, ST
repairRed = *0.5*	A modifier that controls how fast the ship repairs when the crew status is red.	SH, ST
repairYellow = *0.5*	A modifier that controls how fast the ship repairs when the crew status is yellow.	SH, ST
ScaleSOD = *1.1*	Multiplier, resizes the SOD to a smaller or larger size.	SH, ST
sensorsCrewLoss = *2.0f*	This is the percentage of how much of the crew dies, the moment the sensors sub-system gets destroyed.	SH, ST
sensorsHitPercent = *13.0f*	This determines the chances that the sensors sub-system will be hit by a weapon, once the shields are completely down. Must add up to 100 (%) along with the following other values: *crewHitPercent*, *hullHitPercent*, *lifeSupportHitPercent*, *shield-GeneratorHitPercent* and *weaponsHitPercent*.	SH, ST
sensorsHitPoints = *12*	Number of hitpoints this system has, once damage exceeds this number it becomes disabled.	SH, ST
sensorsRepairTime = *3.0f*	The amount of time in seconds that it takes to repair one hitpoint of damage to the sensors sub-system. This is modified by the crew status. If the crew status is yellow, it usually is slower. In status red it is usually even slower. But it can also be totally different (e.g. Borg always repair at the same rate).	SH, ST

Directive	Description	Type
sensorsTargetHardpoints = "hp*12*"	This designates the hardpoint for the sensor sub-system. After this directive a series of hardpoint names is given, that are the firing points for weapons. If the sensor hardpoint's hitpoints are depleted, the ship or station loses it's longer range scanning ability (line of sight), so only the very-short range ling of sight remains.	SH, ST
shieldDelay = *3.0f*	Delay before shields begin recharging once it has reached zero in seconds	SH, ST
shieldGeneratorCrewLoss = *2.0f*	This is the percentage of how much of the crew dies, the moment the shield generator sub-system gets destroyed.	SH, ST
shieldGeneratorHitPercent = *8.0f*	This determines the chances that the shield generator sub-system will be hit by a weapon, once the shields are completely down. Must add up to 100 (%) along with the following other values: *crewHitPercent*, *hullHitPercent*, *lifeSupportHitPercent*, *sensorsHitPercent*, *shieldGeneratorHitPercent* and *weaponsHitPercent*.	SH, ST
shieldGeneratorHitPoints = *12*	Number of hitpoints this system has, once damage exceeds this number it becomes disabled.	SH, ST
shieldGeneratorRepairTime = *2.0f*	The amount of time in seconds that it takes to repair one hitpoint of damage to the shield generator sub-system. This is modified by the crew status. If the crew status is yellow, it usually is slower. In status red it is usually even slower. But it can also be totally different (e.g. Borg always repair at the same rate).	SH, ST
shieldGeneratorTargetHardpoints = "hp*12*"	This designates the hardpoint for the shield generator sub-system. After this directive a series of hardpoint names is given, that are the firing points for weapons. If the shield generator hardpoint's hitpoints are depleted, the ship or station loses it's ability to replenish its shield energy. This does not mean, it cannot have any shields any more (some special weapons only replenish shield energy, while the generator may very well be still offline).	SH, ST
shieldRate = *3.5*	Number of points regained by the shields roughly, per second.	SH, ST
specialEnergyRate = *12*	Specifies the recharge rate of the special energy in units per second.	SH, ST
tooltip = "*You text goes here*"	Specifies the short tooltip that appears when the mouse is passed over the unit.	SH, ST, SW, W
unitName = "*myunit*"	Specifies the unit name.	SH, ST

Directive	Description	Type
verboseTooltip = "*Some longer text*"	The extended tooltip that shows up when the unit is selected or the mouse is held over the unit for an extended time.	SH, ST, W, SW
weapon*1* = "*name_without_odf*"	Specifies the name of the weapon ODF file the number 1 shows that it is the first weapon equipped on this ship, that number changes with each additional weapon added to the ship.	SH, ST
weaponHardpoints*1* = "hp*1*" "hp*3*"	Designates the hardpoints that the weapon uses number 1 shows that it is the first weapon equipped on this ship, that number changes with each additional weapon added to the ship. After this directive is a series of hardpoints that are the firing points for the weapon. They are noted in format hp*xx* where *xx* is a number. Each one is surrounded by "" quotes and separated with a space.	SH, ST
weaponRed = *0.5*	A multiplier that controls how fast the ship fires when the crew status is red.	SH, ST
weaponsCrewLoss = *2.0f*	This is the percentage of how much of the crew dies, the moment the weapons sub-system gets destroyed.	SH, ST
weaponsHitPercent = *2.0f*	This determines the chances that the weapons sub-system will be hit by a weapon, once the shields are completely down. Must add up to 100 (%) along with the following other values: *crewHitPercent*, *hullHitPercent*, *lifeSupportHitPercent*, *sensorsHitPercent* and *shieldGeneratorHitPercent*.	SH, ST
weaponsHitPoints = *12*	Number of hitpoints this system has, once damage exceeds this number it becomes disabled.	SH, ST
weaponsRepairTime = *2.0f*	The amount of time in seconds that it takes to repair one hitpoint of damage to the weapons sub-system. This is modified by the crew status. If the crew status is yellow, it usually is slower. In status red it is usually even slower. But it can also be totally different (e.g. Borg always repair at the same rate).	SH, ST
weaponsTargetHardpoints = "hp*12*"	This designates the hardpoints for the weapons sub-system. After this directive a series of hardpoint names is given, that are the firing points for weapons. If the weapon hardpoint's hitpoints are depleted, the ship or station loses it's ability to fire any weapon, including special weapons.	SH, ST
weaponYellow = *0.5*	A modifier that controls how fast the ship fires when the crew status is yellow.	SH, ST
weldingBeamSprite = "*weldbeam*"	The sprite used for the workers when this ship is built.	SH, ST
weldingBeamWidth = *12*	Sets the width of the beam that the worker bees use when constructing this vessel	SH, ST

Directive	Description	Type
workerBeeHardpoints = "*hp01*"	Defines the hardpoint at which worker bee launching takes place at this station or unit.	SH, ST
workerBeeName = "*my_name*"	Specifies the name of the small worker bee SOD that is used during construction of this unit.	SH, ST

Weapons Directives

Directive	Description
blastRadius = *150.0f*	This is the radius of the explosion that the weapon creates. This is the effect radius not the size of the sprite.
color = *1.0 0.3 1.0*	This RGB value modifies the color of the texture definition.
explosionSprite = "*sparkball1*"	This specifies the explosion that is used for this ordinance when it impacts the target.
explosionSpriteRadius = *50.f*	This is the radius of the sprite for the above explosion. It doesn't have to be the same size as the blast radius.
explSound = "*nel1*.wav"	The name of the sound played when the ordinance explodes.
fadeFactor = *0.9*	A percentage fraction that controls how long before the shockwave or other graphical effect starts to fade out. If this is too high the shockwave will be too big and will appear to be much larger than it's actual effect radius.
fireSound = "*my_file*.wav"	The sound that is played when the weapon fires.
hitChance = *1.0* "*shipname*.odf" *0.75*	This percentage fraction is the chance that the weapon will hit the target. This can be applied to specific ships by adding a line under it in this format: "*shipname.odf*" *x* where *x* is the number between 0 and 1
hullCrewModifier = *0.1*	This percentage fraction is a modifier that controls what percent of the damage which hits the hull while exceeding the damage threshold is used to calculate the percentage of the crew, that dies with each hit. This only applies to damage that is in excess of the damage threshold. This can be applied to specific ships by adding a line under it in this format: "*shipname*.odf" *x* where *x* is the number between 0 and 1.
Length = *10.0*	This is the length of the beam used for this weapon. It is usually used for Pulse Phasers along with the *radius* directive to designate the width and length of the pulse.
lightFalloffRange = *5.0*	This controls the way the color faded in and out by setting the range of the „falloff".
lightFalloffStart = *50.0*	This controls control the way the color faded in and out by setting the start of the „falloff".
lightColor = *0.0 1.0 0.3*	This RGB value controls the way the weapon lighting works. This controls the actual color of a light.
maxShots = *3*	How many shots should we save up? This is the number of shots that will be released the first time the weapon fires. **Note**: This only works on units with *saveFire* set to *yes*.
minimumShotInterval = *0.2*	This is the least amount of time between the saved up shots. This only works the first time the weapon is fired. After that the weapon fires normally until it has a period of time not firing saving up shots again. This only works with units who have *saveFire* set to *yes*.

Directive	Description
needTarget = *1*	This y/n value defines whether the weapon does need manual targeting. This is mostly used for special weapons but is also used for artillery Photons. For the latter do not use it for Armada 1. This will have strange side-effects in the form of „sticking“ photons on the hull of the target, instead of going off.
objectName = "*muball*"	This designates an another ODF that has the specifications for another effect that this weapon has. Could be an artillery wave or an object like an ion storm nebula or any of several other effect types.
omegaTurn = *3.0*	This controls the turning radius of the ordinance. The lower the number the faster the weapon turns to chase the target.
ordName = "*odf_name*"	This specifies the file that is used for the ordinance for this weapon. For example, a gun is a weapon. However a bullet is the ordinance that guns may use. Without some ordinance to do the actual damage the weapon is useless.
Radius = *18*	This is the radius or „thickness“ of the sprite. It is used for Phasers, Photons and various other weapons to control the size of the sprite.
radiusFactor = *10.0*	A multiplier, that is a multiple of the ship’s radius and is used to get the radius of the shockwave.
range = *400.0f*	The maximum distance at which this weapon fires. This weapon will fire when an enemy comes within this range of the ship with this weapon.
rate = *150*	This controls how fast a wave expands. The higher the number the faster the wave moves and grows.
rays = *16*	The number of „rays“ or sections on a shockwave. The higher it is the more detailed the wave will look but also the more heavy the load on the game engine will be.
restrictFireArc = *1*	This tells the game to restricts the arc of fire of the weapon. This is a yes/no question, 0 = *no*, 1 = *yes*.
saveFire = *1*	Should we save up our fire so we can fire several rapid bursts once we get a target? y/n value. **Note**: This only works on units with the classLabel *cannon*.
seekTime = *0.25*	This is the amount of time in seconds the ordinance obeys the omega turn limit before it starts to head for the target. It is mostly used for Photons, since they are fired from a location on the ship and then start to head toward the target after they clear the immediate area of the ship.
shotAccel = *100.0*	This is the speed at which the ordinance accelerates once fired. The higher the number the faster it accelerates.
shotColor = *227*	This is a useless directive that seems to do nothing at all. I believe it was originally intended to change the color of the Phaser beams so that they could use one sprite for the Phasers and change the color through their ODFs. But it seems to never have been implemented. Do not remove it but also don’t bother changing it.
shotDelay = *2*	This is the delay between shots in seconds. It controls the firing rate of the weapon. It is a number and can be virtually any number. The higher the number the slower the firing rate.
spriteDuration = *1.0*	This is the length of time it takes to play through one cycle of the sprite’s animation. Then it loops back to the beginning.

Directive	Description
stopToHitModifier = *0.15*	This multiplier modifies the hitChance when the firing ship is stationary. This can be a positive or a negative number. You can add a - in front to represent a negative percentage which would decrease the chance of hitting.
targetLocation = *1*	This y/n value allows the weapon to target a location on the map instead of having to target a specific unit or station. This is mostly used for special weapons but is can also used for artillery Photons.
texture = "*wselfdes*"	The name of the texture used for this item, usually used for shockwaves.
toHitStopModifier = *0.5*	This multiplier modifies the *hitChance* when the target is stationary. This can be a positive or a negative number. You can add a - in front to represent a negative percentage which would decrease the chance of hitting.
wpnName = "*Weapon's Name*"	This is the weapon's name to be displayed.

Ordinance Directives

Directive	Description
damageBase = *24*	This is the base amount of damage that the weapon does with each hit. This can be applied to specific ships by adding a line under it in this format: "*shipname*.odf" *x* where *x* is a replacement number. This directive belongs in ordinance files.
damageThreshold = *50*	This is the threshold or amount of damage that must be exceeded to cause crew casualties prior to shield exhaustion. This can be applied to specific ships by adding a line under it in this format: "*shipname*.odf" *x* where *x* is a number. This directive belongs in ordinance files.
damageVariance = *2*	This is the amount of variance to the base damage. If this is not 0, not every hit will do the exact same damage. If the damageBase = 4 and damageVariance = 2, then each hit of the weapon can do damage between 2 and 6 (4 ± 2). This can be applied to specific ships by adding a line under it in this format: "*shipname*.odf" *x* where *x* is a number. This directive belongs in ordinance files.
Length = *10.0*	This is the length of the beam used for this weapon. It is usually used for Pulse Phasers along with the *radius* directive to designate the width and length of the pulse.
lifeSpan = *5*	This defines how long the graphic or „sprite" will stay on screen without hitting something before expiring. This must be long enough for the weapon to hit the target from it's maximum range but should not be excessively longer than that. Setting this too high is the reason some mods having Phasers going all the way to the opposite side of the map when missing the target. This directive belongs in ordinance files.
shieldCrewModifier = *0.2*	This percentage fraction is a modifier that controls what percent of the damage dealt to the shields while exceeding the damage threshold, is used to calculate the percentage of the crew that dies with each hit. This only applies to damage that is in excess of the damage threshold. If the actual damage dealt by one shot is below that value, no crew casualties will be applied. This can be applied to specific ships by adding a line under it in this format: "*shipname*.odf" *x* where *x* is the number between 0 and 1. This directive belongs in ordinance files.

Directive	Description
`shieldDuration = 1.0`	This specifies how long the graphical effect of the weapon hitting the shield lasts, the higher the number the longer the shield hit is, this should not be much longer than the time that the weapon is actually making contact with the shield. This directive belongs in ordinance files.
`shotSpeed = 1e6`	This is the speed at which the shot travels. In this example I kept the speed value *1e6*. It stands for 10^6 = 1 000 000. It is used for standard Phasers and represents the speed of light/instant hits. As soon as the Phaser fires it makes contact with the target (if it actually hits the target). This can be virtually any number. The higher the number the faster the speed. This directive belongs in ordinance files.
`Sprite = "sprite_name"`	This specifies the sprite or graphical effect that the weapon displays when fired. These sprites are defined in the sprite files. This directive belongs into ordinance files.

Special Weapons Directives

Directive	Description
`AItargetLocation = 1`	This tells the game that the AI should try to target a location when using this weapon, not target a ship or station. This is a y/n value.
`amplitude = 10`	The height of the Subspace Shockwave.
`attackRange = 400`	The weapon will automatically toggle off once the distance between target and attacker has grown bigger than this value. This can be bigger than the *range* value, meaning, the initial attack range can be less than the eventual range, when the target escapes this weapons reach.
`beneficial = 1`	This y/n value indicates that the weapon is not harmful to the target. It is used for „weapons“ like shield projector, repair teams etc…, so things that are actually beneficial for the target. It is a yes/no question.
`boardingDelay = 1`	The delay in seconds before the boarding party actually starts with the first beam-in aboard the target ship.
`boardingInterval = 1`	Time in seconds between waves of boarding parties.
`boardingSize = 20`	Size of the boarding parties. It is a number that represents the number of crew in each party.
`buttonBorder = 1`	The type of border or frame that the button appears in inside the game. `0` = no border, `1` = offensive border and `2` = defensive border. This has strictly optical implications. It does not change any actual effects a buttons has.
`buttonPriority = 5`	This sets the priority of buttons. So if 2 buttons occupy the same space (e.g. two ships with the same button slot used), the one with the higher button priority will be displayed instead of the one with the lower priority. Hidden buttons are still accessible for the AI.

Directive	Description
buttonSlot = *3*	The slot that the button is placed at on the speed bar of the graphical interface. It can hold values between 1 and 6, while 6 is not actually used in A1. 5 is the special slot, that is set apart from the other four slots. Be aware: Placing multiple buttons on the same slot, makes them invisible and unusable for human players. The AI on the other hand can use all of them, regardless.
chainDamageModifier = *25*	This is a modifier that adds more damage to a chain weapon with each target it hits. Each target impact increases the damage done to the next target in the chain.
chainRadius = *300.0f*	The distance at which a chain weapon will fire and go from target to target. This is different than the standard range directive because it is the **total** distance that the ordinance will travel, including the added up distances from one target to the next one.
cloakEnable = *1*	Can this weapon be fired while the ship is cloaked (not uncloaking in the process)? This is a y/n value.
crewAffectsWeaponDelay = *0*	This specifies that the weapon is not affected by the crew status. Even when the crew status is yellow or red, this weapon will continue firing at the same rate. This is a yes/no question.
DamageModifier = *1.0*	Multiplier that affects the weapons sub-system, changing its damage per hit for classLabel *deathchant*.
damageValue = *70*	This is a number that sets how much damage per second the object does to the unit inside.
detectClass = *3*	The class of the detection weapon. I am not sure what the different classes are for but all the race specific detection weapons use class 3. So I would recommend using class 3 also. Used are however: 1, 2 and 3.
detectRadius = *250.0f*	This is the range at which the weapon ordinance can acquire a target. It is used for tracking weapons like mines that follow a target once it enters their detection range.
detectTimer = *10.0f*	The amount of time the cloaked ship must be in range of the detecting weapon before it is seen.
disableTime = *10*	The amount of time the target sub-system is disabled by this weapon.
drainRate = *50*	This is the rate at which the hull is drained in points per second and converted to bio matter for the 8472 resource extraction beam. Also it is the amount of special weapons energy drained by using the Transwarp Gate and the rate at which the special weapon drains shields from the target (classLabel *shieldinv*). Do not confuse this with *DrainRate*.
duration = *1.0*	The amount of time that the effect lasts in seconds. This is used for several different effects and varies, depending on the context it is used in.
effectTexture = "*Nanites*"	The texture used for the nanites effect.
effectTimer = *4.0f*	How long in seconds does it take for this weapon to be completely in effect? This is used mostly in cloaking devices. This is the time it takes to cloak and uncloak.

Directive	Description
expansionTime = *1.5*	The amount of time in seconds that it takes to go from it's initial size to it's max size.
fadeFactor = *0.9*	This percentage fraction determines how quickly the effect starts to fade out. It is also mentioned in the Weapons Directives section.
fallDuration = *5*	Once the shot geometry starts falling away this specifies how long it stays around while falling, before it disappears.
fallRate = *15*	The speed at which a shot geometry falls off of its target, once it hit the target.
FireModifier = *1.5*	Multiplier that affects the weapons sub-system, changing its firing rate for classLabel *deathchant*.
frequency = *10*	The number of waves that fit in a certain distance for the Subspace Shockwave.
hitCondition = *0*	Sets the hit conditions for targets. Used mostly with weapons that have wave effects. Options are: 0: hit everyone, 1: hit allies and 2: hit enemies.
hitFriends = *0*	Do we want to hit (and damage) friendly ships with this weapon? This is a yes/no question.
HitModifier = *1*	Unknown effect for classLabel deathchant. What it is **not** (derived from testing): Changing the probability to actually hit a target. Firing rate and damage per hit are managed by other parameters. Also the probability to be hit seems to be unaffected by it.
hitSound = "*my_sound*.wav"	The sound of the ordinance hitting the target.
ignoreShield = *1*	This defines whether the ordinance can pass through the shields of the target. This is a y/n value.
initialDistance = *20*	For Transwarp Drive this is the distance between visual and ship.
initialScale = *3.0*	This is the size that the object starts out as. It is a multiplier of the SOD's default size.
lengthScale = *5*	It is used for scaling the length of an SOD that is used for a weapon effect.
lifetime = *20*	The duration in seconds that this particular weapon's effect lasts.
maxHits = *6*	The maximum number of targets a chain weapon will hit. It may hit less targets depending on the distance between targets and the setting in the *chainRadius* directive.
maxScale = *2*	This is the size that the object ends up as. It is a fraction or multiple of the SOD's default size.
maxTargets = *4*	The maximum number of targets that this weapon can hit at once. This directive is used for the *areaCannon* and *areaMissile* classLabels.
maxTime = *5.0*	Sets the maximum time for the movement of a panel in the same direction, while the nanites are effective.
minTime = *2.0*	Sets the minimum time for the movement of a panel in the same direction, while the nanites are effective.

Directive	Description
movementSpeed = *2.0*	How fast will the panels move when infected with Borg nanites? This is a number and the number **must** be an **even number**.
objectName = "*muball*"	This designates another ODF that has the specifications for another effect that this weapon has. Could be an artillery wave or an object like an ion storm nebula or any of several other effect types. This is also mentioned in the standard weapons section but I felt that this one was important enough to repeat here in the special weapons section.
offSound = "*clkdisen*.wav"	Specifies the sound used when this weapon is toggled off.
omegaType = *1*	Can this tractor beam target the *Omega Particle*? This is a y/n value.
ownerSystemDuration = *3*	Duration of the visuals of the computer override weapon (classLabel *override*), specifically the glitter effect applied to the attacker, right after successfully taking over the victim.
particleArt = "wmicremit"	Assigns the sprite that is used for the particle trail that follows this weapon's shot geometry.
partySize = *40*	The amount of crew, per shot.
percentage = *0.02*	This percentage fraction defines the amount of crew removed when the crew of a unit goes insane from the Psychonic Blast.
radius = *20.0f*	The radius for the self destruct blast and other effects.
radiusScale = *2.5*	This is similar to the radius directive for Pulse Phasers. But it is used for scaling the radius of an SOD that is used for a weapon effect.
rays = *9*	The number of vertices a wave weapon shows.
reflectBeam = *1*	Specifies, that beam weapons (such as Phasers) will be reflected, while active. It is a y/n value. **Note**: Pulses cannot be reflected.
reflectMissile = *1*	Specifies, that bullet/missile weapons (such as Photons) will be reflected, while active. It is a y/n value. **Note**: Pulses cannot be reflected.
regenModifier = *30*	This sets the multiplier controlling the speed at which repairs are proceeding.
regularWeaponDelay = *1*	Sets the time delay in seconds before the unit can fire its regular weapons after using this weapon.
RepairModifier = *0.5*	Multiplier that affects the repair speed, changing its effectiveness for classLabel *deathchant*.
repulsionForce = *50.0f*	This is the amount of repulsion force put on the target vessel. It determines how far the ship is pushed back when the repulsion wave hits it.
retractionTime = *2.1*	The amount of time in seconds it takes to go from it's maximum size back to it's initial size.
rollRate = *12*	The speed at which the weapon effect or SOD spins or „rolls".
rotationRate = *0.00015*	Speed of the rotation of the shot geometry.
safe = *0*	Specifies whether this weapon is „safe" or not. If it is not safe it will damage friendly units as well as enemies. This is a y/n value.
scaleCount = *5*	Effect unclear, used for classLabel *utribeam*.
scaleSize = *1.0*	Relative scale to the size of the SOD.
scaleValue = *0.30*	Scale value applied to the object.

Directive	Description
ScalingCount = *17.0f*	This sets the scaling for objects like the Ion Storm nebula and Ultritium Burst. It affects the eventual size of the object.
ScalingStepSize = *0.20f*	This sets the scaling for objects like the Ion Storm nebula and Ultritium Burst. It affects the speed of the growth of the object.
segments = *30*	The number of vertices the Subspace Shockwave is long.
shieldsReturned = *0.5*	The upper percentage fraction of the shields that is returned to the target ship when this weapon hit it. If the ship is already at that level of above, no changes to the shields will occur.
shotGeometry = "*wmyotron*.sod"	This designates what the SOD file used for the ordinance is called. Some weapons do not use 2-dimensional sprites as their graphic. Instead they use a 3D SOD model (usually that's some sort of projectile).
shotInfo = "*NaniteHitInfo*"	This is the reference to game events that triggers an audio warning about the nanites affecting your systems.
slowPercent = *70*	The percentage by which this weapon slows down it's target's warp drive. This is used for gravity mines. It will never slow down a ship entirely, as it has no effect on the impulse speed portion of the speed.
smokeFrames = *16*	Affects the texture and trail patterns for a smoke trail of a weapon. Most weapons do not work with a smoke trail but for example chain weapons do. Guess: The number of images taken from the texture file, to form a progressing smoke effect.
smokeInherit = *0.5*	Affects the texture and trail patterns for a smoke trail of a weapon. Most weapons do not work with a smoke trail but for example chain weapons do.
smokeLifespan = *0.6*	Affects the texture and trail patterns for a smoke trail of a weapon. Most weapons do not work with a smoke trail but for example chain weapons do.
smokePause = *0.01*	Affects the texture and trail patterns for a smoke trail of a weapon. Most weapons do not work with a smoke trail but for example chain weapons do.
smokeRadius = *2.0*	Affects the texture and trail patterns for a smoke trail of a weapon. Most weapons do not work with a smoke trail but for example chain weapons do.
smokeTexture = "*Xsmoke*"	Affects the texture and trail patterns for a smoke trail of a weapon. Most weapons do not work with a smoke trail but for example chain weapons do. This variable sets the texture to be used for this smoke pattern.
smokeVariance = *1.0*	Affects the texture and trail patterns for a smoke trail of a weapon. Most weapons do not work with a smoke trail but for example chain weapons do.
special = *1*	This marks the weapon as a special weapon as opposed to a standard one. It is a y/n value.

Directive	Description
specialEnergyCost = *50*	This specifies the amount of special energy the weapon uses with each shot. For special weapons it can be the cost of one shot or the cost per second (if the weapon is a toggle on/off type weapon). This is not usually used for non special weapons but it can be used for non specials too, if desired. This directive goes in the ordinance file.
specialValue = *20*	This is for setting the value of a specific special weapon. For different weapons it does different things. Sometimes it is the amount of time a weapon is in effect. Other times it is the amount of hit points restored to a friendly ship. It strictly depends on the weapon it is used with.
speedModifier = *1.2*	Multiplier that affects the engine sub-system, resp. impulse speed. It is used with the *speed_boost* classLabel, as well as with *deathchant*.
spreadSpeed = *15*	The speed at which the Subspace Shockwave spreads.
startReturnTime = *2.0*	This sets the amount of time after the nanites effect ends and the panels begin to return to their normal positions.
stayDuration = *5*	How long in seconds that a shot geometry stays around after hitting the target.
stepSize = *0.0004f*	Set the step size of radians.
switchToAttack = *0*	After firing this weapon should we switch to attack mode and attack with other weapons we have? This is a y/n value.
targetAlliedBuildings = *1*	Allows for this weapon to be used on friendly stations. It is a y/n value.
targetAlliedCraft = *1*	Allows for this weapon to be used on friendly ships. It is a y/n value.
targetEnemyBuildings = *1*	Allows for this weapon to be used on enemy stations. It is a y/n value.
targetEnemyCraft = *1*	Allows for this weapon to be used on enemy ships. It is a y/n value.
targetLocation = *1*	Allows for this weapon to be used on a location (no target unit required). It is a y/n value.
targetRange = *100.f*	Effect is unclear.
targetSystemDuration = *3*	Duration of the visuals of the computer override weapon (classLabel *override*) in the third phase, when both, the victim and the attacker glitter. After this, only the victim glitters, until the effect wears off.
timeBetweenSystems = *2*	Duration of the visuals of the computer override weapon (classLabel *override*) between first and third phase, where none are glittering.
timer = *5*	The amount of time in seconds before this weapon detonates. Used mostly for self destructs.
timeAtMaxSize = *2.5*	The amount of time in seconds, that it stays at it's maximum size.
timeBetween = *2*	This sets the time between the pulses or explosions when using *wave_Effect*.
toggle = *1*	Sets the weapon as a toggle weapon, even though the weapon doesn't use the *toggleWeapon* classLabel. This is a yes/no question.

Directive	Description
`transparency = 0.0`	This percentage fraction sets the amount of transparency when fully cloaked. This is in addition to the standard additive blending.
`TurnModifier = 1.0`	Multiplier that affects the turning speed, changing its effectiveness for classLabel *deathchant*.
`unsafeCloakDistance = 400.0f`	Sets the distance from enemies that you can cloak a vessel.
`visualFactor = 1.0`	This is the factor to scale the art by (not the range of effects).
`waveArc = 90`	The arc of the Subspace Shockwave in degrees.
`waveEffect = 1`	Does this weapon have a wave effect because it affects multiple ships in the surrounding area, instead of a standard sprite? This is a yes/no question. Do not confuse this with *wave_Effect*.
`waveLength = 110`	The length of the Subspace Shockwave.
`waveName = "Wholowav"`	The name of the ODF file to refer to for the wave. It is just the name of the file. It does not require the *.odf* extension to be added here. Same as *wave_name*.
`waveSpeed = 10`	The speed of the Subspace Shockwave cycling up and down.
`wpnCategory = "ARCA"`	This sets the category for the weapon. It seems to be only multi-target and toggle weapons. I am not sure what purpose it actually serves. Options are: TOWE, ARCA, AREA, ARMI and CANN.
`wpnPriority = 0`	Unsure what this is for. But it appears in most toggle weapon files. It could have something to do with the AI and how aggressively they try to attack the ship that has this weapon. Funny: Even in the originally delivered ODFs from Activision/Mad Doc Software, the lines with this definition have a comment *UNKNOWN*. So apparently even they didn't know.
`wpnReticle = "gblast"`	Defines the cursor that is used to target this weapon. As *gblast* does not exist, it's basically no cursor at all. This is used for area of effect weapons emanating from the casting ship.
`wormholeID = "wborgate"`	Specifies the ODF name of the wormhole to be used. It does not require the *.odf* file name extension.
`xplGround = "xmslgnd"`	Sound for impacts on the ground (not used in Armada).
`xplVehicle = "xmslcar"`	Sound for impacts on units (ships).
`xplBuilding = "xmslbld"`	Sound for impacts on buildings (stations).

Physics Directives

Directive	Description
`backwardAccel = 1`	Deceleration; this is a multiplier of *forwardAccel*. This parameter scales with impulse speed.
`combatSpeed = 10`	This is the speed setting for the ship while it is in combat.
`forwardAccel = 3`	Acceleration in forward direction, normal is 1. This parameter scales with impulse speed.

Directive	Description
forwardControlDistance = *200*	This is the angle in degrees to make it stop dampening its turning.
frontBackInertia = *50*	The bigger this number, the slower acceleration into intended movement direction is. Bigger ships should have larger inertia, as this helps give the illusion of them having a bigger mass.
impulseSpeed = *10*	This is the speed setting for the ship while it is not in combat and not at warp, e.g. traveling in the gravity well of a planet. Many settings scale with impulse speed, so faster ships do most things faster.
leftRightInertia = *7.5*	This defines the effect of inertia while changing direction in plain (turning left or right). Tight turns make the ships skid a bit, depending on this value. It should be a lot smaller than *frontBackInertia*.
maxPitch = *45*	This is the maximum climb or dive angle, in degrees. This **must** be less than *90*, so 89 is OK.
maxReverseSpeed = *4.0*	This is as a fraction of impulse speed.
maxRoll = *40*	Roll is moving one wing up and the other down. The max roll determines how much the ship banks when turning.
maxRotationSpeed = *0.75*	This is the speed that Borg ships (like Cubes or Spheres) rotate or spin on their vertical axis.
pathLeadDistance = *100*	When moving to a target location, the ship will follow a point on its path that's this far in front of it. This is usually 150.
physics = "*smooth*"	This is the physics type of this file. There are three allowed values for this parameter: smooth is the standard physics, for moving like most races ships do in the game. borg that makes ships move like Borg ships do: Very little banking and rolling, and some additional rotation for some Borg ship types. hover is not used in Armada.
pitchAlpha = *1*	Maximum pitch acceleration/deceleration, normalized to 1, scales with *pitchOmega*, so also with impulse speed.
pitchAlphaFractionAtRest = *0.0*	Defines how much the ship accelerates its nose tilting while standing still. 0.0 means can't turn while not moving, 1.0 means can turn fully, even when still. This does **not** scale with impulse speed.
pitchDefault = *0*	The angel in degrees to the plain to which the ship will tilt its nose by default (when not affected by objects). Usually that's 0.
pitchDefaultSpeed = *1*	The speed of the change of the angel to which the ship will tilt its nose by default (when not affected by objects). Zero or negative speed means to stay at the last pitch.
pitchOmega = *1*	Maximum pitch rate, normalized to 1. Pitch is the angle forwards and backwards, meaning the tilt of the nose of the ship. This parameter scales with impulse speed.
pitchOmegaFractionAtRest = *0.0*	Defines how fast the ship tilts its nose at max while standing still. 0.0 means can't turn while not moving, 1.0 means can turn fully, even when still. This does not scale with impulse speed.

Directive	Description
pitchRotationSpeed = *0.0667*	Pitch is moving the nose up/down. The bigger this value, the faster can a ship enter/leave a dive or climb.
rollCoupling = *0*	This defines, how much a ship rolls to the sides when it turns, normalized to 1. Roll is the tilting of a ship's side when it turns. Set this to 0 to eliminate roll entirely. This does **not** scale with impulse speed.
rollRotationSpeed = *0.5*	Roll is moving one wing up and the other down. The bigger this value, the faster can a ship enter/leave a roll.
rotationAcceleration = *2.0*	The spin acceleration of a Borg ship.
rotationStiffness = *1.0*	?
tooCloseToTurn = *50*	When attacking a target while being closer than the given value, the ship will not try and turn around to face the target, even when facing another direction. This is usually 50 (which is rather close).
tooSlowToTurn = *5*	A parameter similar to *tooCloseToTurn* but meant for the rotation of Borg ships. When a Borg ship is moving slower than that, it will stop turning around its vertical axis.
turnAlpha = *1*	Maximum turn acceleration/deceleration, normalized to 1. This scales with *turnOmega*, so also with impulse speed.
turnAlphaFractionAtRest = *0.0*	Defines how much the ship accelerates its turning while standing still. 0.0 means can't turn while not moving, 1.0 means can turn fully, even when still. This does **not** scale with impulse speed.
turnControlSquared = *1*	Determines how to dampen (decelerate) the turning rate once it's close to it's target. 1 means to square it, to make it even more dampened.
turnOmega = *1*	Maximum turn rate, normalized to 1. This parameter scales with impulse speed.
turnOmegaFractionAtRest = *0.0*	Defines how fast the ship turns while standing still. 0.0 means can't turn while not moving, 1.0 means can turn fully, even when still. This does **not** scale with impulse speed.
turnRadius = *30*	The turn radius in meters. Whenever a ship is turning, it is turning around a centered point, which lies in this distance.
upDownInertia = *7.5*	This defines the effect of inertia while changing direction out of plain (going up or down). Tight turns make the ships skid a bit, depending on this value. It should be a lot smaller than *frontBackInertia*.
velocCoeffStill = *0.5*	This defines the fraction of max turning speed while standing still. 0.0 means none at all, 1.0 means 100 %.
velocWhenCoeffHalf = *0.7*	This defines the fraction of max turning speed while moving with at least half of the maximum speed.
warpSpeed = *101*	Warp speed is the speed the ship travels when moving long distances or moving to an area designated by a right click on the mini map. Warp is much faster than impulse or combat speeds. However most weapons will not fire at warp until you „drop out of warp“ (slowdown). If you want the ship to not have warp drive at all use a negative number like −1.

Depending Physics Directives

These here are set automatically from the above parameters. If you change the other parameters only by a little, you probably won't need to adjust these here. But if you make more drastic changes, you'll probably need to. They affect the control loops. This means things like how far ahead to start coming out of a turn or when to start braking. If these are set all wrong, you'll get poor performance from the ships. You may not even notice. For example, the ship might start breaking long before it needs to and get slower and slower as it approaches the destination, while it could as well start slowing down a lot later.

Generic Control Parameter

- `braccelControlStiffness = 4.0`

ServoYaw Parameters

- `sy_yawAngleCoeff = 1.497`
- `sy_yawVelocCoeff = 0.2`

ServoPitch Parameters

- `sp_pitchAngleCoeff = -1.197`
- `sp_velocAngleCoeff = 0.060`

Race Directives

Directive	Description
`affectedByPsionicInsanity = 0`	This sets whether or not this race is affected by Psionic Insanity weapons. This is a yes/no value.
`boardingStrength = 1.75f`	This is how strong the race is when fighting as or against boarding parties. Higher is stronger and 1.0 is an average strength. The ratio of two races defines how effective boarding is. E.g. attacker having 1.75 while the victim has 1.0 means, for each 5 crew beamed over (removed from the attacker's unit), 8.75 crew is removed from the victim unit.
`canBeCommandeered = 0`	This parameter set whether the race's vessels can be captured by enemies. This is a y/n value.
`canCommandeer = 0`	This parameter set whether the race can capture enemy vessels. This is a y/n value.
`crewAccumulationRate = 4.0`	This is the number of crew accumulated per second and Starbase. This is the base number before any modifiers are applied.

Directive	Description
crewRedStatus = *0.25f*	This is percentage fraction at which a unit enters red status. If the crew is at this percentage or below, the unit is in red state. See also *crewYellowStatus*.
crewRetreatRatio = *0.2f*	This defines when the race will retreat from battle if controlled by the AI. This is the percentage fraction of the crew that is still remaining on the ship. When the ship is below that percentage they will retreat. In the example shown the race would not retreat until the crew is down to 20 %.
crewYellowStatus = *0.5f*	This is the percentage fraction of crew at which yellow state will be entered. If the crew amount is bigger, it is in green state. See also *crewRedStatus*.
displayName = "*Klingon*" | *The display value representing the race.* | | *displayKey* = "GUI_RD_BORG" | *This sets the name of the file that is used to define the GUI layout for the race.* | | *instantActionSlot* = 1 | *This is the slot index the race uses on the drop-down selection menu of the Instant action game setup screen. The first slot is number 0.* | | *insufficientOfficersEvent* = "InsufficientRomOfficers"	The name of the event to refer to when this race runs out of officers (or the race's equivalent of them).
interfaceConfiguration = "*gui_bor*.cfg"	This specifies the file used to create the race's GUI, specifically, the placement of the interface parts and which parts are present or missing for the interface.
interfaceSprites = "*gui_borg*.spr"	This specifies the files used to create the race's GUI, specifically what graphics files to use and what these parts are, and where they are positioned in the actual TGA files that contain the interface graphics.
minimalUnits*1* = "*rconst*.odf"	These are the units you get if you start a game with minimal units. This is basically just the name of the race's construction vessel. But more units may be defined. The continuous index starts at 1.
name = "borg"	The race's name. This is the name that the game uses in scripts and interfaces to select the appropriate files for things like AI files.
officerRes = "*GUI_CP_BORG_OFF_RES*"	This defines the names for the officer equivalents for the race. For example the Borg use *Power Nodes* instead of *Officers*.
officerTooltip = "*GUI_RD_BORG_OFF_TOOLTIP*"	This defines the tooltip for the officer equivalent for the race.
officerUpgradeODF = "*borgoff*"	This specifies which file is used for officer upgrades for this race.
officerVerboseTooltip = "*GUI_RD_BORG_OFF_VTOOLTIP*"	This defines the verbose tooltip for the officer equivalent for the race.

Directive	Description
planetCrewAccumulationModifier = *1.5*	This is a modifier that adjusts the rate of crew accumulation for starbases near a planet.
repairRed = *0.6*	This is a multiplier defining the relative repair rate of the unit while in red state. In the example it is only 60 % of normal repair speed.
repairStrength = *0.8f*	This defines how fast the race repairs, the larger the number the slower they repair.
repairYellow = *0.8f*	This is a multiplier defining the relative repair rate of the unit while in yellow state. In the example it is only 80 % of normal repair speed.
retreatStrength = *0.1f*	This defines when the race will retreat from battle if controlled by the AI. This is the percentage of shields remaining. When the ship is below that percentage they will retreat. In the example, 10 % or less of shields remaining will make the AI retreat this ship.
standardUnits*1* = "*8472_mothership*.odf"	These are the units you get when starting a game with normal units. Usually this is a couple construction ships, a base and a scout. The continuous index starts at 1.
superUnits*1* = "*bconst*.odf"	These are the units that you get when starting a Super Side match. These usually include a construction ship from each race. The continuous index starts at 1.
track*0* = "*audiofile*.wav"	Defines the audio file, that is playing as music during campaign and multi-player. The index starts with 0.
transportSprite = "*btransport*"	This specifies the sprite name for the transporter effect for this race.
transportToEnemyOnHard = *0*	This setting for the AI determines whether to transport crew to enemy ships to try and capture them. This depends also on the difficulty level of the game. It is done by the AI **only** if the game is on hard. This is a yes/no value.
transportToFriendsOnNotEasy = *0*	This setting for the AI determines whether to transport crew to friendly vessels for support. This depends also on the difficulty level of the game. It is done by the AI **only** if the game is not on easy. This is a yes/no question.
weaponRed = *5.0f*	This is the multiplier defining the relative firing times while at red state. In the example this is 500 % more **time** during two shots, reducing the firing **rate** by 80 %.
weaponYellow = *1.5f*	This is the multiplier defining the relative firing times while at yellow state. In the example this is 50 % more **time** during two shots, reducing the firing **rate** by 33 %.

In the race's ODF there is also a list of sound directives that specify which sounds are played for various events, like beginning construction or your base being attacked. These are very self explanatory but apparently not used (the corresponding audio files are simply not present, so theses settings cannot be the ones taking effect). For completeness reasons, this is the list of these

directives:

- *computerEnginesOffline1*,
- *computerEnginesOffline2*,
- *computerFacilityConstructionBegin1*,
- *computerFacilityConstructionBegin2*,
- *computerFacilityConstructionComplete1*,
- *computerLifeSupportOffline1*,
- *computerSensorsOffline1*,
- *computerShieldGeneratorOffline1*,
- *computerShipConstructionBegin1*,
- *computerShipConstructionBegin2*,
- *computerShipConstructionComplete1*,
- *computerUnderAttack1* and
- *computerWeaponsOffline1*.

Directives for Other Miscellaneous Things

Directive	Description

[**Modding**] [Tools] [ODF Files] [ODF Directives] [Class Labels] [Tech Tree Files] [SOD Files] [Buttons] [Wire Frames] [Sprites] [AI Scripts] [Model Hierarchy] [Node Names] [Emitter Names] [Texture Animation Names] [Sprite Names]

[Back to Modding]