The Flatspace IIk Technical Manual v1.01

by Mark Sheeky

General Guidelines

Items in the game universe can be created or modified by experienced users.

Each item should be defined in a plain text file with any filename and a ".txt" extension. Files should be placed in the "Items" folder. If you delete or rename the folder, the default internal items will be used instead. There is no fixed limit to the number of items (although each demands a unique positive ID, so there is a theoretical maximum in the billions region). Each item needs a corresponding name, and usually a description in the text files Items0.txt and Items1.txt in the Data folder (see the "text" parameter for details on that).

Item files are lists of parameters with the format parameter=value on each line. Some parameters are strings, some are integers (whole numbers), some are floating points (fractional numbers), and some are boolean values which should be "true" or "false" or "yes" or "no".

The "Plugin Management" option in the game will list the number of plugins loaded. Any rejected due to bad parameters will be counted too and, in that circumstance, a file called ErrorLog.txt will be saved out indicating the error and the filename that caused the problem.

Most of the parameters are limited but not all, and some are not fully tested in extremes. It's possible that bad values, such as unexpected negative numbers or illegal ID numbers (like ammoid for an item that doesn't exist) could cause a crash or other unexpected fault. However I think this is unlikely, and the Flatspace programming is so beautiful and written with such genius that things like that have generally been checked for anyway.

There now follows a description of each parameter.

genuscode=(integer)

A number 1 to 19, this indicates the type of item. The number that must be one of the following which corresponds to the item type:

1 for GENUS_CARGO, cargo items.
2 for GENUS_PACKAGE, a mission delivery item.
3 for GENUS_ARMOUR, armour/armor (I love armourous armour).
4 for GENUS_WEAPON, main weapons.
5 for GENUS_TURRETWEAPON, weapons for turrets, which are largely the same as main weapons.
6 for GENUS_WEAPONAMMO, ammunition for GENUS_WEAPON or GENUS_TURRETWEAPON items.
7 for GENUS_ECMAMMO, ammunition for GENUS_ECM (only flares use ammo).
8 for GENUS_MISSILE, missiles or other pylon weapons like mines.
9 for GENUS_HYPERDRIVE, hyperdrives.

10 for GENUS_THRUSTER, thrusters.
11 for GENUS_GENERATOR, generators.
12 for GENUS_SHIELD, shields.
13 for GENUS_ECM, electronic counter measures, anti-missile devices.
14 for GENUS_HOOVER, hoover/scoop.
15 for GENUS_SCANNER, scanners.
16 for GENUS_RADAR. radar.
17 for GENUS_ESCAPEPOD, escape pod.
18 for GENUS_ROBOT, robots.
19 for GENUS_USELESS, any other item like pets, or anything that doesn't do anything but you'd like to have in the game, such as a record player or Scarridberry lollipop.

Example

genuscode=1

indicates a cargo item like Meat/Fish or Rock Fragments.

text=(string)

Contains the text [BRACKET] that forms the description of the item in the game. The text should be entered into the Items0.txt and Items1.txt files in the Data folder. Items0.txt is for UK/Aus English, Items1.txt is for US English. The same information should be typed into both files.

For most items the first line following the [BRACKET] heading (which should be unique) is the name of the item, eg. "M1 Machine Gun". The second line should be a short description for the further information. Make sure it's not too long.

For GENUS_CARGO items the first line is the name of the item, and subsequent lines sub-categories, eg. for the "Clothing" cargo there are five sub-categories, so the line is "Clothing", and subsequent lines are "Fashion Goods", "Shoes", "Uniforms", "Workwear", "Protective Suits".

For GENUS_PACKAGE items the first line is the name of the item and the second line is the text for the mission description in the form of "upon delivery of the item". eg. for the Scarlet Falcon F9 sports car package the first line is "Scarlet Falcon F9" and the second line is "upon delivery of a sports car".

Example

text=[ITEM M1MACHINEGUN]

would indicate that the text description for this item in the Items0.txt and Items1.txt files follows the [ITEM M1MACHINEGUN] bracket.

Technical notes: There is no fundamental requirement to use all caps or include the square brackets in these headings but this is highly recommended to avoid confusion and duplication. The brackets are useful because without them "ITEM M1MACHINEGUN" and "ITEM M1MACHINEGUN2" would be confused by the program because the first line terminates before the other.

id=(integer)

Unique ID code for each item. Duplicate ID codes are not permitted so items with duplicate ID codes will be rejected. There is no enforced limit to the ID number, which is an unsigned long integer. The largest ID used in the game is below 5000 so anything over that is new and unused.

Example

id=1

Technical notes: These ID codes are used when assigning default equipment to the player when the game begins. This, and the "refinedid" and "ammoid" parameters are the primary uses of this parameter. As a result, be careful of modifying existing ID codes; if you change the ID code for the basic M2 Machine Gun for example the character classes that normally start with that weapon will start the game unarmed. You could use this fact to start the player with something new, but it would have to be legal in mass and ship compatibility.

onsaleinbaseprobability=(integer)

This indicates the percentage probability of the item being shown for sale in a base. Range 0 to 100.

Example

onsaleinbaseprobability=10

indicates a 10% chance of the item appearing on sale.

onsaleinpolicebaseprobability=(integer)

This indicates the percentage probability of the item being shown for sale in a police base. Range 0 to 100.

Example

onsaleinpolicebaseprobability=50

indicates a 50% chance of the item appearing on sale in a police base.

onsaleinpiratebaseprobability=(integer)

This indicates the percentage probability of the item being shown for sale in a pirate/independent base. Range 0 to 100.

Example

onsaleinpiratebaseprobability=0

indicates that that item would never appear on sale in a pirate base.

boolintegral=(boolean)

If true the item is an integral part of a vessel, something built in to the ship design that can't be removed. This is for fixed items like specialised thrusters, generators, or other components tied to a particular ship design. These items are never on sale and can't be removed when intalled, so have no use as a new object at the moment. For the time being set this to false.

Example

boolintegral=false

indicates this is a normal item that can be fitted and removed, bought and sold.

boolmineral=(boolean)

If true this is a mineral item and might be inside an asteroid and carried sometimes by miner type AI craft. Currenty used only for Crystals, Rock Fragments and Ores.

Example

boolmineral=false

for most items.

Technical notes: Crystals and Metals are permanent parts of the universe that are not present as plug-ins because they are needed to generate the specialised crystal and metal asteroids.

boolgenerateforai=(boolean)

If true then this item could be chosen for AI craft. This should be true for almost all items including hyperdrives, thrusters, generators, radars, weapons, shields, ecm, etc. as most AI ships can carry the same equipment as the player. Set to false for GENUS_CARGO and GENUS_PACKAGE items because those shouldn't be fitted to ships as equipment. GENUS_USELESS, GENUS_AMMO and GENUS_ECMAMMO items are currently unused by the AI craft so you can set to false for those too, although nothing bad would (probably) happen if those items were fitted.

Example

boolgenerateforai=true

for most items.

Technical notes: The AI uses radar to detect other craft, and robots to repair and

maintain like the player. The same goes for scanners, hoovers and most items, which are utilised by the artificial intelligence as needed.

boolgenerateforplayer=(boolean)

If true then these items could be chosen to add to initial player craft at random. Used specifically for the ship basics; generators (GENUS_GENERATOR), thrusters (GENUS_THRUSTER), and hyperdrives (GENUS_HYPERDRIVE). Set to false for other pieces of equipment because those are chosen from a list based on the player profession. This is used to stop guild equipment or other expensive upgrades from being given as starting equipment.

Example

boolgenerateforplayer=true

would allow this generator, thruster or hyperdrive into the initial player craft. For other types of item this should be false.

boolispet=(boolean)

If true then this item is a pet. Pets should be GENUS_USELESS items. This is solely used for the "menagerie" secret. Pets or other useless items have no function other than to absorb damage, protecting other vital ship systems.

Example

boolispet=true

for a Korean talking llama. Set to false for a radar.

mass=(integer)

Mass of the item. Mass must be positive, although zero is permitted.

Example

mass=1

indicates an items with a mass of 1 unit.

cost=(integer)

Cost of the item when purchased. Cost must be positive, although zero is permitted.

Example

cost=100

indicates that this item costs 100 credits.

guildrank=(integer)

The rank needed to purchase this item, if guild restricted. For normal items this is zero but for restricted guild items this should be a number from 1 to 5 to indicate the rank needed before this item can be purchased.

Example

guildrank=1

indicates that this item requires a rank of 1 before the item can be purchased.

guildtype=(integer)

For restricted items this indicates the guild needed to purchase this item. For normal items this should be set to zero. This is ignored if guildrank is zero. It should be one of the following numbers:

0, GUILD_MERCHANT, merchant guild.
1, GUILD_COURIER, courier guild.
2, GUILD_OUTLAW, outlaw's guild.
3, GUILD_CRIMEFIGHTER, crime fighter's guild.
4, GUILD_ALIENFIGHTER, alien fighter's guild.

Example

guildtype=1

indicates that this item can only be purchased by members of the courier guild.

maxammo=(integer)

The maximum ammo capacity of the item if this is a weapon (GENUS_WEAPON or GENUS_TURRETWEAPON), or flare launching ecm device, GENUS_ECM. It's ignored for other types of item and should be set to zero. If zero for weapons or ecm it indicates infinite ammo.

Example

maxammo=500

indicates that this item holds 500 shots when full.

tradeclass=(integer)

GENUS_CARGO items only. This indicates the trade class of the item, the class that

different bases import and export. There are five classes of item, and should be a number 1 to 5 as follows:

1, TRADECLASS_MINERAL, minerals.
2, TRADECLASS_AGRICULTURAL, agricultural items.
3, TRADECLASS_LIGHTINDUSTRIAL, light industrial items.
4, TRADECLASS_HEAVYINDUSTRIAL, heavy industrial items.
5, TRADECLASS_TECHNOLOGICAL, technological items.

Example

tradeclass=2

indicates that this cargo item is agricultural.

refineditemid=(integer)

GENUS_CARGO items only. This is the ID number of the item that this cargo is refined into using a refining scoop/hoover. For example Staple Foods are refined into Biological Derivatives by the scoop, so the Staple Foods refineditemid is set to 4008, the ID of Biological Derivatives. The ID should be a cargo item of the same mass, however the game should be able to cope with other masses and even other equipment types, although items such as weapons that are mutually exclusive will probably be rejected upon scooping if you already have something like that fitted. Set to zero for items that do not refine further.

Example

refineditemid=4002

indicates that this cargo would refine into crystals when scooped.

Technical notes: The ID for Crystals is 4002. The ID for Metals is 4004. The ID for Meat/Fish is 4008. Those items are global internal items.

numsubcategories=(integer)

GENUS_CARGO items only. A number 0 to 10 for the sub-category of cargo in the description; for example the current Rock Fragments item can be further subdivided into Carbon and Basalt, shown in brackets when purchased. Up to 10 further sub-descriptions should be in the Items0.txt and Items1.txt text files beneath the main item name. Some cargo like Water has no subcategories, which is fine too.

Example

numsubcategories=3

indicates that this cargo could be of three further subtypes.

stockvolitility=(integer)

GENUS_CARGO items only. A number 0 to 10 which indicates how volatile the stock prices are, effectively how often the prices change. They change more often with a lower number, so a number of zero will change the prices each time the stock market is processed. That process effectively makes cargo prices with a lower stockvolitility more stable due to the quantity of fluctuations. The most volatile existing stock items have a value of 6.

Example

stockvolitility=1

indicates that this cargo might change price every other loop of the stockmarket.

Technical notes: The price direction will change from moving up, moving down or stable, when a random number between zero and stockvolitility equals zero. Stock prices are updated for one item each frame, 1/50 of a second by default.

armourpoints=(integer)

GENUS_ARMOUR items only. Number of armour points of protection this item provides. This must range 0 to 10000.

Example

armourpoints=10

is the default protection that plasteel armour provides.

energydrainperuse=(integer)

GENUS_WEAPON, GENUS_TURRETWEAPON, GENUS_MISSILE, GENUS_HYPERDRIVE, GENUS_THRUSTER, GENUS_SHIELD, GENUS_ECM, GENUS_HOOVER and GENUS_SCANNER only. The amount of energy used for each use of the item. For weapons this is per shot. This is typically zero for weapons like the machine gun or missiles. For hoovers/scoops or thrusters this is per-pulse, typically 1 for all thrusters and zero for hoovers. For scanners, hyperdrives this is per use, but is currently zero for all hyperdrives. For ecm units this is very high for EMP devices, several thousand units, but less for reprogrammers and zero for flare launchers. Shield units drain energy only as they recharge and this number is 1 for almost all shield units. This number must be positive.

Example

energydrainperuse=1

is a typical value for a thruster or shield. A scanner might use 10 or 20, and weapons much more.

lifetime=(integer)

GENUS_WEAPON, GENUS_TURRETWEAPON and GENUS_MISSILE only. This is the lifetime in frames of the weapon shot before it vanishes, effectively setting the range of the weapon. There are 50 frames per second by default. Most weapons should vanish a little off the screen, although missiles should persist for longer as they chase down an enemy. A typical lifetime for a weapon is under 100, with 1000 being more typical for a missile.

Example

lifetime=50

would set a lifetime for a shot of about one second, which is the default for the Cannon.

damage=(integer)

GENUS_WEAPON, GENUS_TURRETWEAPON and GENUS_MISSILE only. Damage of the shot.

Example

damage=5

would set the damage to 5, the default damage for the M1 Machine Gun.

stundamage=(boolean)

GENUS_WEAPON, GENUS_TURRETWEAPON and GENUS_MISSILE only. If true then this is a stun weapon. Normal damage will be done to shields and armour, but when penetrated, no further damage will be done and the generator will be disabled.

Example

stundamage=true

would indicate that this is a stun weapon.

firingrate=(integer)

GENUS_WEAPON and GENUS_TURRETWEAPON only. The firing rate in frames, so lower is faster. The number must range from 1 to 1000. A value of 1000 is about one shot every 20 seconds. A value of 1 is typically used for beam/flamer type weapons.

Example

firingrate=25

would indicate a maximum firing rate of two shots per second.

totalrange=(float)

GENUS_WEAPON and GENUS_TURRETWEAPON only. This is used by the AI craft to determine the range of a weapon and should be be set to shotvel*lifetime for almost all weapons. The only exception is for beam/flamer type weapons when it is normally set to shotvel*lifetime*0.75 so that the AI craft will move in a bit closer before they attack. This is a floating point value that must be positive.

Example

totalrange=250.0

would be the correct value if shotvel=5.0 and lifetime is 50.

Technical notes: Miner AI craft will move within totalrange*0.75 of an asteroid before firing. So now you know.

muzzlefairy=(integer)

GENUS_WEAPON, GENUS_TURRETWEAPON and GENUS_MISSILE only. This is the sprite number for the graphical effect that flashes on the muzzle of the weapon when it is fired. As with any fairy code, set to -1 for none. The range is zero to 73 or 328 to 339. The best way to use this is to copy it from an existing weapon with the effect you want.

Example

muzzlefairy=17

is the flash graphic for the M1 Machine Gun.

muzzleemitter=(integer)

GENUS_WEAPON, GENUS_TURRETWEAPON and GENUS_MISSILE only. Emitters are the particle effects used in the game. This parameter is a code for the particle effect to use on the muzzle of the weapon when it's fired, the puff of smoke for the machine gun for example. Set to -1 for none. The best way to use this is to copy it from an existing weapon with the effect you want. The maximum number is 34.

Example

muzzleemitter=1

is the puff of smoke used by the M1 Machine Gun.

muzzlealphavel=(float)

GENUS_WEAPON, GENUS_TURRETWEAPON and GENUS_MISSILE only. Muzzle alpha velocity. This is the fade speed of the sprite that appears when weapons are fired. It's a negative number ranged from -1 to 0 where -1 is the fastest fade, like a fast flash "off",

and -0.1 would take ten frames to fade out. If you set it to zero the muzzle sprite will stay there forever so only set it to zero if muzzlefairy=-1 because you're not having a muzzle graphic. Most missiles have a muzzlealphavel of -0.01.

Example

muzzleemitter=-0.25

will fade the muzzle flash in four frames, the speed of the M1 Machine Gun effect.

shotfairy=(integer)

GENUS_WEAPON, GENUS_TURRETWEAPON and GENUS_MISSILE only. The main sprite code for the bullet/missile/shot. As with any fairy code, set to -1 for none. The range is zero to 73 or 328 to 339. The best way to use this is to copy it from an existing weapon with the effect you want. Note that machine gun and minigun bullets are present as graphics in the game, they are just invisible.

Example

shotfairy=19

is the invisible machine gun bullet.

shotemitter=(integer)

GENUS_WEAPON, GENUS_TURRETWEAPON and GENUS_MISSILE only. This is the particle effect code for any particles that should be attached to the shot or bullet or missile. Set to -1 for none/off. Missiles generally have this as -1 for off, with the later glowemitter parameter used for that effect. The best way to use this is to copy it from an existing weapon with the effect you want. The maximum number is 34.

Example

shotemitter=9

is the particle effect for the plasma weapon.

glowfairy=(integer)

GENUS_WEAPON, GENUS_TURRETWEAPON and GENUS_MISSILE only. This is a second sprite code for the bullet/missile/shot. Two are overlayed for special visual effects reasons, notably here the glowing star and trail for missiles and rockets. As with any fairy code, set to -1 for none. The range is zero to 73 or 328 to 339. The best way to use this is to copy it from an existing weapon with the effect you want.

Example

glowfairy=49

is the glow effect for Renton Rockets.

glowemitter=(integer)

GENUS_WEAPON, GENUS_TURRETWEAPON and GENUS_MISSILE only. This is the particle effect code for a second particle effect attached to a weapon bullet or missile. Set to -1 for none/off. Missiles tend to use this to produce the long streak effect in different colours. The best way to use this is to copy it from an existing weapon with the effect you want. The maximum number is 34.

Example

glowemitter=21

is a missile streak effect, with 22 or 23 as alternatives for missiles.

shotvel=(float)

GENUS_WEAPON, GENUS_TURRETWEAPON and GENUS_MISSILE only. Shot velocity. This is the speed of the shot, a number from -1.0 to 1.0 where positive numbers fire the shot forwards and negative numbers fire backwards. In practice mines have a slight movement instead of being set to zero because a stationary shot is easy to accidentally crash into just after firing. A typical missile has a velocity of 0.1. Remember that totalrange should be shotvel*lifetime.

Example

shotvel=0.1

for an average speed weapon.

Technical notes: It is possible to shoot yourself but only after your shot has left your collision sphere and re-entered it.

ammoid=(integer)

GENUS_WEAPON, GENUS_TURRETWEAPON and GENUS_ECM only. This is the ID number of the item used as ammunition for this weapon or flare launcher.

Example

ammoid=1002

is the ID for Minigun Ammo.

terminationexplosion=(bool)

GENUS_WEAPON, GENUS_TURRETWEAPON and GENUS_MISSILE only. If true then weapons will visually explode when they impact a target. This should be true for all

missiles. It is false for most weapons (a simple flash of impact will appear instead). This represents a visual effect only and has no impact on damage levels.

Example

terminationexplosion=true

will make an explosion effect when the weapon hits something.

startsfxnumber=(integer)

GENUS_WEAPON and GENUS_TURRETWEAPON only. A number 0 to 93. This is the sound effect number that starts when the trigger is first pressed. For miniguns or beam weapons for example it might be a looping effect. The best way to identify any sound effect is to copy it from an existing item that uses the sound you want.

Example

startsfxnumber=29

is the flamer gas-burn sound effect.

stopsfxnumber=(integer)

GENUS_WEAPON and GENUS_TURRETWEAPON only. A number 0 to 93. This is the sound effect number that plays when the trigger is released. This might be an recoil or reload sound. The best way to identify any sound effect is to copy it from an existing item that uses the sound you want.

Example

stopsfxnumber=8

is the M1 Machine Gun recoil sound effect.

firesfxnumber=(integer)

GENUS_WEAPON and GENUS_TURRETWEAPON only. A number 0 to 93. This is the sound effect number that plays when a shot is fired. The best way to identify any sound effect is to copy it from an existing item that uses the sound you want.

Example

firesfxnumber=17

is the Rothspar Laser fire sound effect.

techlevel=(integer)

GENUS_MISSILE and GENUS_ECM only. This is the technology level of the missile or ECM system. A missile with a techlevel of 1 will be stopped by an ECM with a techlevel of 1 or more, but evade an ECM with a techlevel of zero. It can range from zero to 100, although only 4 tech levels are used by the default equipment.

Example

techlevel=0

is the level of the Kuzai-M1 missile and the basic Razor Flare Launcher.

missileagility=(float)

GENUS_MISSILE only. This is missile agility where higher numbers represent a more agile missile with a tighter turning circle. An unguided weapon like the Razor-T1 Torpedo would have zero agility. A very agile missile would have an agility of 2048. Missiles in the game use 512 (poor), 1024 (medium) or 1536 (good) for agility factors.

Example

missileagility=512

is the agility factor of the Kuzai-M1 missile.

warhead=(integer)

GENUS_MISSILE only. This is zero for normal missiles and 1 for nuclear weapons. Nuclear missiles always kill their target and also damage any ship within a certain range.

Example

warhead=0

represents a normal missile.

nucleardamage=(float)

GENUS_MISSILE only. This only applies when warhead is set to 1 for nuclear weapons. It indicates the splash damage done to nearby ships when a target is hit. The damage factor diminishes with distance from the point of impact. With nuclear warheads the target vessel is always destroyed.

Example

nucleardamage=50000

is the default damage factor for the only nuclear warhead in the game. It is very dangerous!

launchsfxnumber=(integer)

GENUS_MISSILE only. A number 0 to 93. This is the sound effect number that plays when the missile is fired. The best way to identify any sound effect is to copy it from an existing item that uses the sound you want.

Example

launchsfxnumber=36

is the standard missile launch sound effect.

hyperjumpforce=(float)

GENUS_HYPERDRIVE only. This determines the power of a hyperdrive, combining with the mass of the vessel to determine the maximum jump range. This is 320 for the smallest hyperdrive, and goes up to several thousand for a larger drive. The maximum jump range is the hyperjumpforce/vesselmass*100. See the existing hyperdrive definitions to get an idea of factors to use. This number can range from zero to 50000.

Example

hyperjumpforce=320

would make this drive as powerful as the smallest Gray Hyperdrive.

Technical notes: For the 13 Gray Hyperdrives in the game the formula for hyperjumpforce is (i*10.0f+20.0f)*16.0, and the drive mass is i*2+4, where i ranges zero to 12. For the 10 Adams Hyperdrives the formula for hyperjumpforce is (i*100.0f+200.0f)*16.0, and the drive mass is i*20+40.

hyperdriverecharge=(integer)

GENUS_HYPERDRIVE only. This is the maximum recharge time for the hyperdrive. Smaller numbers will recharge the drive faster. The actual time depends on the distance jumped such that if you jump the maximum range of the drive it takes the full time to recharge, and if you jump 50% of the maximum range, it takes 50% of the time to regenerate. By default this is 1750 for all drives, which at the standard 50 frames per second is 35 seconds for a full recharge. It seems longer when you're waiting to leap, doesn't it!?

Example

hyperdriverecharge=1750

would make this a standard hyperdrive.

thrust=(float)

GENUS_THRUSTER only. This is the thrust power of the thruster, a number between zero and 1. The actual speed of a flying ship depends on thrust force and mass, as well as the default universal drag added to the game (we don't want actual frictionless space physics do we!). The thrust parameter is very sensitive, partly because fitting a thruster into a ship with different mass makes balancing more difficult. This value is 0.04 for the smallest thruster, and 0.08 for the 2t VL Series Thruster, growing by just 0.008 with each ton of thruster mass.

Example

thrust=0.08

would make this thruster the same as a 2t VL Series Thruster.

Technical notes: Thruster design is very boring.

thrustsfxnumber=(integer)

GENUS_THRUSTER only. A number 0 to 93. This is the sound effect number that plays when the thrust is applied. The best way to identify any sound effect is to copy it from an existing item that uses the sound you want.

Example

thrustsfxnumber=39

is the smallest thrust burn sound effect, with 40, 41 and 42 increasing in boom and power.

thrustfairy=(integer)

GENUS_THRUSTER only. The main sprite code for the glow of the thrusting engine. As with any fairy code, set to -1 for none. The range is zero to 73 or 328 to 339. The best way to use this is to copy it from an existing thruster with the effect you want, but generally for thrusters you'll want to use the standard glows because there are no special effects used here. The thrust fairies are 332 to 339.

Example

thrustfairy=332

is the TL-1 Thruster thrust effect.

thrustemitter=(integer)

GENUS_THRUSTER only. This is the particle effect code attached to the glowing thruster. In effect this is a plain streak because thrusters don't tend to scatter. Set to -1 for none/off. The best way to use this is to copy it from an existing thruster with the effect you want. Values of 25, 26, 27 and 28 are designed for thrusters. The maximum number is 34.

Example

thrustemitter=25

is the particle effect for TL-1 Thruster.

maxenergy=(integer)

GENUS_GENERATOR only. The energy capacity of the generator. It must be positive and should be greater than zero unless you want to sit still in space without any power.

Example

maxenergy=1024

is the energy capacity of the smallest Core Generator.

rechargerate=(integer)

GENUS_GENERATOR only. The number of energy units gained per frame. For smaller slower generators this is 1 or 2 (it's 1 for the solar generators). It ranges up to 4 for the largest generator. It's a fine balance because with low drain thrusters, shields and weapons it's all too easy to make a generator that recharges faster than it could use energy up so in practise this number works best when low. It must range from zero to 10.

Example

rechargerate=2

is the recharge rate smallest Core Generator.

maxshield=(integer)

GENUS_SHIELD only. The maximum shield value in armour/damage units. This can range from 1 to 10000. The Tetron shields start at 100 units and grow by 100 with each upgrade.

Example

maxshield=50

is the value of the Warhawk shield.

ecmtype=(integer)

GENUS_ECM only. A number that determines the type of ECM system, which should be as follows: 0, for ECM_FLARE, a flare launcher, 1 for ECM_EMP, an EMP device, 2 for

ECM_REPROGRAMMER, a reprogrammer.

Example

ecmtype=0

indicates a flare launcher.

ecmradiussquared=(float)

GENUS_ECM only. For ECM_REPROGRAMMER and ECM_EMP types this is the radius of effect squared of the ECM. Missiles beyond this range will not be affected by the ECM. The standard radius is 6 (so this value should be 36, 6*6). That's about the area of the visible screen. This number must range from 0 to 1000. Flare launchers effectively have infinite range.

Example

ecmradiussquared=36.0

sets the range to the visible screen, the standard range for current ECM systems.

triggersfxnumber=(integer)

GENUS_ECM only. A number 0 to 93. This is the sound effect number that plays when the ECM is triggered. The best way to identify any sound effect is to copy it from an existing item that uses the sound you want.

Example

triggersfxnumber=84

is the standard flare launcher sound effect.

hoovermass=(integer)

GENUS_HOOVER only. This is the maximum mass of the vessel, pod or asteroid the scoop can assimilate. This value can be zero for things like tractor beams or vaporisers. A value of 10 will allow the hoover to scoop lifepods, or 5 for cargo containers. Larger values could be used to scoop larger asteroids or vessels.

Example

hoovermass=10

is the standard hoovermass of a mining scoop.

hooveralpha=(float)

GENUS_HOOVER only. This is the starting transparency of the hoover beam as it shines from your ship. Generally these start opaque and fade to transparent as they fly away. A value of 1.0 is opaque and zero is transparent.

Example

hooveralpha=1.0

is a fully opaque sprite.

hooveralphavel=(float)

GENUS_HOOVER only. Alpha velocity. This is a negative number that indicates how quickly the hoover beam should fade. It's a number from -1 to just under zero because if you set it to exactly zero the sprite will never fade out. A value of -1.0 will make the beam fade quickly and be very short, so a number like -0.05 is better, and -0.075 is usually used for the hoovers in the game.

Example

hooveralpha=-0.075

is a typical value for this.

hooverrate=(integer)

GENUS_HOOVER only. This is the firing rate for the graphical effect pulses. 4 is a typical number, so you get a new pulse every 4 frames. This is only for the graphical effect and doesn't affect the power or operation of the scoop.

Example

hooverrate=4

indicates one pulse every 4 frames.

hooverfairy=(integer)

GENUS_HOOVER only. This is a sprite code for the beam blob. As with any fairy code, set to -1 for none. The range is zero to 73 or 328 to 339. The best way to use this is to copy it from an existing scoop with the effect you want, although all of the hoover fairies are 69, 70, 71, 72 and 73.

Example

hooverfairy=73

is the fairy used for the L1 Tractor beam.

hooveremitter=(integer)

GENUS_HOOVER only. This is the particle effect code for the scoop beam. Set to -1 for none/off. The best way to use this is to copy it from an existing scoop with the effect you want. The existing hoover emitters are 30, 31, 32 and 33. The technical maximum number is 34.

Example

hooveremitter=30

is the emitter effect used by the Baker V-1 Man Trap.

hoovervel=(float)

GENUS_HOOVER only. This is the speed of the beam graphic as it leaves the ship. It's a number that ranges from -1 to 1, where 1 is very fast. A value of 0.05 is used for all of the scoops in the game. Negative values would make the beam move backwards and, frankly, have never been tested.

Example

hoovervel=0.05

is a typical hoover velocity.

hooversfxnumber=(integer)

GENUS_HOOVER only. A number 0 to 93. This is the sound effect number that plays when the scoop is used. The best way to identify any sound effect is to copy it from an existing item that uses the sound you want. The existing scoop sounds are 48, 49, 50, 51 and 52.

Example

hooversfxnumber=48

is the standard first scoop sound effect.

boolhooverscoopcrewascrew=(bool)

GENUS_HOOVER only. A flag to "Scoop Crew As Crew". If true the scoop would scoop any live passengers and put them in your ship. Used for life saving scoops and man traps. Do not set this to true AND boolhooverscoopcrewasmeat because that would confuse the computer. If both hooverscoopcrew flags are set to false then crew are killed/vaporised.

Example

boolhooverscoopcrewascrew=true

would scoop live crew in lifepods and place them comfortably in your ship.

boolhooverscoopcrewasmeat=(bool)

GENUS_HOOVER only. A flag to "Scoop Crew As Meat". If true the scoop would scoop any live passengers and convert them into meat, as mining scoops often do. Do not set this to true AND boolhooverscoopcrewascrew because that would confuse the computer. If both hooverscoopcrew flags are set to false then crew are killed/vaporised.

Example

boolhooverscoopcrewasmeat=true

would scoop live crew and convert them into dead meat. How horrid.

boolhooverscoopcargoascargo=(bool)

GENUS_HOOVER only. A flag to "Scoop Cargo As Cargo". If true the scoop would scoop any cargo and bring the items on board, like normal cargo scoops. Do not set this to true AND boolhooverscoopcargoasrefined because that would confuse the computer. If both hooverscoopcargo flags are set to false then cargo is destroyed/vaporised.

Example

boolhooverscoopcargoascargo=true

would scoop cargo like a cargo scoop.

boolhooverscoopcargoasrefined=(bool)

GENUS_HOOVER only. A flag to "Scoop Cargo As Refined". If true the scoop would scoop any cargo and bring the items on board as its refined version, as specified in the cargo definition. Do not set this to true AND boolhooverscoopcargoascargo because that would confuse the computer. If both hooverscoopcargo flags are set to false then cargo is destroyed/vaporised.

Example

boolhooverscoopcargoasrefined=true

would scoop cargo like a cargo scoop.

scansfxnumber=(integer)

GENUS_SCANNER only. A number 0 to 93. This is the sound effect number that plays

when the scanner key is pressed. The best way to identify any sound effect is to copy it from an existing item that uses the sound you want. The existing scanner sounds are 44, 45, 46 and 47.

Example

scansfxnumber=44

is the basic ID scanner sound effect.

boolscannerpilotname=(boolean)

GENUS_SCANNER only. If true, the scanner will return the pilot name when used.

Example

boolscannerpilotname=true

would create a scanner that returns the pilot name.

boolscannerpilotbountydead=(boolean)

GENUS_SCANNER only. If true, the scanner will return the kill bounty of the pilot.

Example

boolscannerpilotbountydead=true

would create a scanner that returns the kill bounty.

boolscannerpilotbountyalive=(boolean)

GENUS_SCANNER only. If true, the scanner will return the capture bounty of the pilot.

Example

boolscannerpilotbountyalive=true

would create a scanner that returns the capture bounty.

boolscannercargo=(boolean)

GENUS_SCANNER only. If true, the scanner will return the cargo of the scanned vessel.

Example

boolscannercargo=true

would create a cargo scanner.

boolscannerimportsexports=(boolean)

GENUS_SCANNER only. If true, the scanner will return the imports and exports of the scanned vessel (well, base station). It's really nothing to do with sex sports.

Example

boolscannerimportsexports=true

would create a trading scanner.

boolscannerhyperspacetracking=(boolean)

GENUS_SCANNER only. If true, the scanner will have hyperspace tracking ability.

Example

boolscannerhyperspacetracking=true

would create a tracking scanner.

radarrange=(float)

GENUS_RADAR only. The radar range which is a float value that can range between zero to 100 (although zero itself isn't permitted). The visible screen is about 6 units, and the smallest radar has a range of 10 units. The largest radar in the game has a range of 46 units.

Example

radarrange=16.0

sets the range to 16 units.

blipsfxnumber=(integer)

GENUS_RADAR only. A number 0 to 93. This is the sound effect number of the ping that plays when the radar pings a blip, or blips a ping, well when it detects a ship. This is currently 43 for all radars. It would sound odd if not distracting to have anything but a very subtle sound for this, so the standard 43 is probably the best option, although you might want to create a silent radar by setting the number to zero.

Example

blipsfxnumber=43

is the standard radar blip.

escapepodhullid=(integer)

GENUS_ESCAPEPOD only. This is the hull ID number of the escape pod and should be 200, 201 or 202 for the three different hulls that can be used as escape pods. Although theoretically any hull could be an escape pod, it would really be no fun to "escape" into an asteroid, Battlestation or Starcity.

Example

escapepodhullid=200

is the Genie Escape Pod hull.

escapeprobability=(integer)

GENUS_ESCAPEPOD only. This is percentage chance of a successful ejection when escaping. This is 90 in existing escape pods.

Example

escapeprobability=99

would mean there is a 99% chance of a successful escape into this escape craft when your ship is destroyed.

boolmaintenance=(boolean)

GENUS_ROBOT only. If true this robot can perform ship maintenance.

Example

boolmaintenance=true

means that this robot can perform maintenance.

boolmedical=(boolean)

GENUS_ROBOT only. If true this robot can perform medical tasks.

Example

boolmedical=true

means that this robot can heal injured crew.

boolsecurity=(boolean)

GENUS_ROBOT only. If true this robot can perform security tasks.

Example

boolsecurity=true

means that this robot can act as a guard.

purchasedmessage=(string)

GENUS_USELESS only. This contains the text [BRACKET] of an optional message to display when the item is purchased. The message to display should be on the line after the [BRACKET] line and placed in the Items0.txt and Items1.txt files in the Data folder.

Example

if you put

[ITEM HELLO MESSAGE]
Thank you for buying.

in Items0.txt and Items1.txt

and set

purchasedmessage=[ITEM HELLO MESSAGE]

then the game would display "Thank you for buying." when this item was purchased.

discardwhenbought=(bool)

GENUS_USELESS only. Essentially for messages, if true the item will be deleted when purchased, after any message had been displayed. No items in the game use this flag, but it's here all the same and almost certainly works.

Example

discardwhenbought=true

would make the item vanish when bought.