AttySpace Tutorial: Getting Started

Ben "Damnal" Cotton (Damnal@ATS)
12/23/02
See Change Log at the end of the document

Note: *This tutorial assumes you are logged in as One on Mordak and Yarb's stock database or my own stock database. Unfortunately for many of you, my stock database, which contained fixes to many of the glitches in the vanilla one is not compatible with newer compiles. Also because of certain features kludged into my server, I won't be providing another copy of mine, since odds are you do not have my hardcode patches.*

If you're completely new to AttySpace (if you've never used it on ATS), then I suggest you read Shaard's Beginner's Guide, which is available at http://www.trekmush.org/files/sguide.txt, otherwise enjoy the tutorial.

If you have never, ever used soft or hardcode, do not attempt to start a public MUSH. Learn to softcode and then move onto bigger and better things. **To clarify: this tutorial was not written to teach you how to code. It was written to demonstrate how to use AttySpace; there is a general assumption throughout the tutorial that you are capable of understanding basic softcode concepts. If you don't understand softcode, do not start with AttySpace… start with basic coding, http://moosh.net/mush/guide/ is likely the best tutorial you'll find for learning softcode. Once you've learned these programming concepts, you'll find this tutorial and the resources available on the ASpace support site much more useful.**

Table of Contents:

First we'll cover the basic creation of a ship, as creating classes is a little difficult and someone else is writing a tutorial about it. If you've installed Yarb's @space/makeship command on #10 you should skip this next part of the tutorial as this part is about the manual creation of a ship.

Special thanks to the people who made this tutorial possible, Andy, Yarb, Mordak and others, you know who you are. A very special thanks to my family and my girlfriend for dealings with my coding obsessions despite not always understanding them.

1 – Creating your first ship

The first step is to create the object, which will represent your ship. This object will hold all the fun stuff like the coordinates of your ship, it's allocations, it's damage etc. For this tutorial we'll create one called SS Shuttle. This will be our ship, or at least a near resemblance of one.

Next we have to decide what class our ship should be, a plain old shuttle looks good, so @parent SS Shuttle to #66. We now know what class our ship is.

The next step is to add the zone lock to our ship, the standard one I've seen used in the database is: **iswiz/1** this uses the iswiz lock that is on #30, the master space object. Now that this is done, create the bridge for your ship by @dig'ing it. @chzone the room to your ship object.

Next we need a few consoles, so lets create a few. @create 3 objects called Console 1, Console 2, and Console 3, respectively. Drop all three objects and paste the following code segment into your telnet window.

```
@dolist remove(lcon(here),%!)=&NAME ##=s(name(##))
@dolist remove(lcon(here),%!)=@set ##=wizard
@dolist remove(lcon(here),%!)=@set ##=monitor
@dolist remove(lcon(here),%!)=&SHIP ##=s(zone(here))
@dolist remove(lcon(here),%!)=@parent ##=#21
@dolist remove(lcon(here),%!)=&MODES ##=command communications
engineering helm inactive monitor navigation off operations science
security tactical transporter weapons damage fighter
```

Next we need to determine the next available sdb#. To do this we'll call sdb(e), which will give us the next available number. Now we'll add an sdb attribute to our ship object/zone with a value of the number we were just given. **(&sdb <shipobj dbref#>=<the number returned by sdb(e)>)** Now we have to make sure the consoles have the proper data on them. We'll do this by pasting the following line into our telnet session.

```
@dolist remove(lcon(here),%!)=&SDB ##=s(get(zone(here)/sdb))
```

There, now our consoles are set to work with our ship, and our ship is almost set to work in space. We just need to do a few more little things.

First we have to load our ship into the space database, otherwise we can't fly. Now, a very critical step, set the ship object (in my example #212) SHIP-OBJECT by typing: @set #212=SHIP-OBJECT. So now we'll execute the sdb() function with the following parameters, (substitute the number of your ship for the number in this code): sdb(read, #212)
If all goes well, you'll be given a nice return of 1 for your trouble, if you somehow got 0, there's a bug in your ship object, (most likely it's not parented properly). If you got a 1 however man a console and try to start your ship.

Uh oh… no fuel, that could be a problem if we want to go anywhere. Well we can remedy this problem very easily, just paste in the following code. Substitute the sdb number of your ship for the 7 in this code.

```
th sdb(put,7,320000000,fuel,anti)
th sdb(put,7,640000000,fuel,deut)
```

There, we now have a fully functional Shuttle in space. It should only take you about 15 hours to get to where the ships that Mordak and Yarb included are. ☺

2 – Adding Rooms to your Ship

Adding a new room to your ship that will display things such as the destruction messages is quite simple. First @dig the room and link it to the first, if you want to. @chzone the room to the ship you want it to be part of. Next you have to add the number of the room you just created into the ROOMS attrib on ship obj. The attrib is a space delimited list so you should have any troubles. Enjoy the multiple rooms on your ship.

3 – Creating Planets and Bases

Wow, creating planets and bases is really easy, to be exact it's the same as creating a ship, just with different parents. The only real different I see is that you don't have to fuel planets, though, if you create a planet class with sensors or the like, I'd recommend giving them some.

4 – Fun stuff with SDB()

SDB() is the function that give AttySpace life, it handles everything, from putting your ship in space to changing it's allocs etc. As you saw earlier in this tutorial, sdb is used to initialize your ship.

A lot of people seem to want to know how to change the coordinates of their ship so I'll use this section (for now) to show how this is done.

```
th sdb(put,<sdb#>,[pc2su(100)],coords,x)
```

That line would move the x value of your ship to 100. I think you can figure out the rest easily.

Using sdb() you can also refuel ships, gate anomalies and essentially everything you can do with the consoles.

You can take a look at the sdb() reference table at the end of the document if you need to find specific sdb() commands.

5 – Class Creation (and some other really tedious stuff)

Well, today's lesson is about class creation. Class creation takes a little bit of thought, and a whole lot of typing. You have to think about how big a ship you want, if you want weapons on the ship, and if so, where they should face. How much power should the ship have? How strong should the shields be? These are all questions you'll ask yourself when designing your class. My first suggestion is to write down on paper most of the things you want to remember about the class you're going to create. Mordak's description of each attribute is a must when designing your ship, as it's essentially your template. There are a few formulae that you may want to remember when designing your class. The important ones relate to warp speeds.
The first determines **p** which, I'll say, stands for power for lack of a better term.
p = (0.99 * total MW produced by main) + (0.01 * total power * power % alloc'ed to move) The other important variable, **a**, is the 5th variable in the movement list. This variable is very important in a lot of the other calculations too.
Warp Speed = sqrt(10.0 * **p / a**)

On a side note, when the ship is travelling with a cloaking device active, **p** is multiplied by 0.75. If the shields are active, it is multiplied by 0.75 for each shield that is up. IE, All four shields are up and **p** has a value of oh 10. The final **p** would be: 3.16. Cloaking changes the speed in the same way.

Impulse speeds are calculated in almost the same way, however instead of the total MW produced by the main, it is based on the MW of the aux. However impulse isn't affected by the shields or cloak.

Maximum fuel is fun. Full antimatter is determined by multiplying your movement ratio (5th variable in the movement list) by the ship's light year range (the final variable in the technology list) multiplied by 640000000. Deuterium is based on the same formula but substituting 320000000 instead of 640000000.

With this information and Yarb's crib sheet you should be able to create a half decent class within an hour or so.

Here is our example ship class (do not use this on a public MUSH without my permission). I call it the Avalon class Raider. It's a small ship, a little larger than some of the fighters that come in the stockdb, but unlike the fighters has cargo space. It also includes a cloaking device, whose power usage is balanced the same way as that of the D'deridex III in the stockdb. The normal warp speed for this ship is 15.31 and the impulse speed is 98.

```
@create Avalon
&CLASS Avalon=Avalon
```

This part should be fairly self-explanatory.

```
&TECHNOLOGY Avalon=1 1.05 1 0.9 1 1.3 1.3 1 1
```

This sets the basic technology ratings. As you can see, there is nothing really fancy here, just a slightly worse than normal cloaking device and some medium duty reactors.

```
&CLOAK Avalon=0 11 0 1 0 1
```

This sets up the cloak. It uses 11MW of power and it exists.

```
&BATT Avalon=0 0 1 2.75 1
&AUX Avalon=0 0 1 20 1
&MAIN Avalon=0 0 1 18.92 1
```

These are the reactors. Nothing particularly fancy here, if you don't know these variables… you're in too deep already.

```
&MOVEMENT Avalon=0 0 0 0 0.8 1000 0 0 0
```

This sets the movement ratios and the movement rating (?). The fifth variable is the one that adjusts the warp speed and the impulse speed. The movement rating is related to turning speeds, but that is really messy code, which doesn't need to be explained in too much depth here.

```
&SHIELD Avalon=4 14 1 1 0 0 0 0 1 1 1 1
&STRUCTURE Avalon=1 23020 2000 0 20 20 0 0 1 1 20 20
```

The final bit of the class… if you've looked at Mordak's crib sheet you'll understand the variables involved here.

6 – Save Yourself! (Some trouble)

An interesting title don't you think? That's because a lot of people have been asking the same questions over and over, and despite the question having been answered just as many times, people still ask them. This section is for answering them.

Q: When I adjust the settings on my ship (using SDB, the attributes, whatever) AttySpace doesn't see the changes.
A: This is because of one of a few things. You're trying to change the warp and impulse speeds with SDB… this isn't going to work, read the class section. Or… you're forgetting to sdb(read,<obj>) so that ASpace reloads the ship's statistics.

Q: My reactors don't work.
A: ENABLE TinyMath if you recompile.

Q: Ship A and Ship B have the same coordinates, but don't see each other on sensors.
A: Ensure that the space attribute on the ship objects is identical *and* make sure that they are both located in the same space room.

Q: I try some of the sdb() commands, but they don't seem to work on my ships.

A: Take a look at the sdb command table at the end of this document. Maybe you're calling the wrong command.

Q: How do I make a wormhole?
A: @create Wormhole 1
@create Wormhole 2
@parent Wormhole 1=#995
@parent Wormhole 2=#995
&STATUS Wormhole 1=0 0 0 0 0 0 0 0 <SDB of wormhole 2>
&STATUS Wormhole 2=0 0 0 0 0 0 0 0 <SDB of wormhole 1>
think sdb(read,<sdb of wormhole 1>)
think sdb(read,<sdb of wormhole 2>)
think sdb(put,<sdb of 1>,<value>,coords,x)
think sdb(put,<sdb of 1>,<value>,coords,y)
think sdb(put,<sdb of 1>,<value>,coords,z)
(etcetera)
If the object is in another space, any ship gating will be transferred to that space (ala the Alternate Universe on ATS, and the inside-wormhole-space).
(This is taken directly from Andy's post on the Aspace Forums)

Q: How do I change the territories?
A: The territories are hardcoded into the server. You have to edit space_variables.c to adjust the current ones and to add more you have to edit space.h. If you understand C the bits in those two files you have to edit are very straightforward. Important Note: In order to use the edited files you must recompile.

Q: Where can I get the help files from ATS?
A: You can't. Some people have logged ATS' help files in their entirety. Neither Mordak, Yarb nor myself endorses this activity, as technically it qualifies as copyright infringement.

Q: How do I recompile if I'm running Windows?
A: You can use either the freely available cygwin tools or the tools provided by Microsoft in Visual Studio 5 and higher. Do not ask for the VS tools, we can't provide them to you. However some of us are willing to use our resources to compile a custom server for you.

Q: My ships won't work… even if I follow the tutorial's instructions.
A: The tutorial forgot an important piece of information. Before you can sdb(read,#????) you have to set the ship object SPACE-OBJECT.

Q: The MUSH tells me that SHIP-OBJECT isn't a valid flag, you're wrong.
A: No, despite what you may think, I am not wrong. The original release of AttySpace, which this tutorial is based on, uses SHIP-OBJECT not SPACE-OBJECT. The flag was changed in more recent releases of ASpace.

Q: @zone isn't a valid command!
A: Again, this depends on the patch level of Penn you're using. If you're using an older version @zone is an alias for @chzone. (I use an older version of ASpace than what is present on the site.) Since some of you felt like complaining so much, all references to @zone are now redone to @chzone.

Q: Why don't my channels work? I'm using the ASpace site's win32 releases.
A: This is courtesy of some of the Penn features being disabled in those releases. You have three choices, live with it, recompile with those features enabled, or convince one of us who have experience recompiling to make a special compile for you.

Q: Are you that ASpace guy?
A: Quite obviously, it's nice that a lot of you use this tutorial, but if you're going to IM me asking if I'm the guy who wrote this tutorial… don't. Obviously if you're getting my IM information from the end of this document, you know that I'm the tutorial guy.

Q: Will Atty give me support for AttySpace or ASpace if I log onto ATS?
A: Definitely not. Atty released AttySpace without any support. If you log onto ATS and expect someone to help you with AttySpace coding you may find yourself booted. As stated in the disclaimer for ATS, they're happy that you want to emulate them, but ATS does not give away code. (AttySpace is an exception and the version released by Atty was an old version that doesn't have all the same features as the current version that ATS runs.)

Q: Didn't there used to be working hyperlinks in this tutorial?
A: Yes, there were. However I have had the interesting situation of several full system reinstalls and in the process misplaced my Adobe Acrobat CD, so as a temporary measure (that has become less than temporary) I'm using FreePDF and GhostView which unfortunately do not support many of the advanced PDF features.

Q: Why does this tutorial suck so much? It doesn't hold my hand through the whole creation process.
A: It's not supposed to. This tutorial is aimed at teaching a programmer of advanced or intermediate skill the tools they need to create a functional AttySpace environment. Though many portions of the tutorial contain very specific references, many of the basic points such as object creation are glossed over under the assumption that you understand this concept.

Q: Can you teach me how to code? Hack my server? Take over the world?
A: Unfortunately, no. I barely have time in my own life right now to practice these things, so I definitely do not have time to teach someone else. My plate is full, I am however willing to help you out occasionally if I have time, so don't hesitate to contact me with questions via IM or email. It may take a little time for them to be answered but I will answer them if they're within the scope of my knowledge.

Q: How do I… what is… how does…
A: Read over the older posts in the ASpace Support Forums. Chances are your question has been answered previously. If it's been brought up more than once and has a definite answer, you'll find it in here.

98 – Contact Information

There are several ways that you can contact me.

The first is email, I've recently changed my accounts a bit, so my primary email address is: damnal@the0rem.net. Older versions of the tutorial show cyberspeed@geocities.com as my primary account, please disregard this since Yahoo has recently changed their POP3 service and I only check that account on a random basis. Do not email my hotmail account. There is a 99% chance your message will not be received.

Do not email me with complaints that the tutorial sucks. If you're having trouble with a section ask me for clarification, everyone learns differently.

Another way of contacting me is through the Message Boards at http://aspace.whizy.com, however just like the email, it may take a day or two before I read your message.

My IM clients are now open much more than they had been before, so here's the communication info:
MSN Messenger: damnal@hotmail.com
ICQ: 34920675
AOL Instant Messenger: Damnal Sakotd
Yahoo! Instant Messenger: cyberspeed_b@yahoo.com

99 – Change Log

12/23/02 – Added more information to the disclaimer at the beginning of the document due to some confusion over the intent of this document. Also added a few minor questions to the Q&A section that complement the disclaimer nicely. Added a little bit of information about adding an SDB attribute to a ship object since there seems to have been a lot of confusion over what was meant by this. Wow… two updates in two days… amazing seeing as I usually wait 6-8 months. ☺
12/21/02 – Added a few more questions to the Q&A section. Updated the note at the top since there has been a little confusion regarding my stockdb. Some minor grammar and spelling fixes throughout the tutorial. If anyone has any major issues I'll fix them when I get the chance. This tutorial has been helping people for two years?! Happy Holidays.
05/01/02 – Added a few new bits of info into the Save Yourself some trouble section. Updated the contact information to accurately reflect changes in my account info. All references to @zone have been changed to @chzone.

11/11/01 – Added an important piece of forgotten information to Section 1. I'm amazed no one mentioned that problem yet. Also fixed a few minor typos. Added the IM info to the end of the Contact Info since my OLT is much higher than it used to be.
11/04/01 – Added to Section 6. Covers territories and some other useful advice. Gave Yarb the credit for the work on the sdb() reference.
10/03/01 – Edited several mistakes. Added more contact info. Added a slightly modified version of Mordak's sdb() reference table to the end of the document for easier reference. More updates to save yourself some trouble.
07/11/01 – Section 6 added. Newest entry in changelog is now colorful. As are new section headings. (I'm bored). Added additional info before section 1, including a table of contents.
06/14/01 – Change Log renumbered to 99 (Contact info to 98) to prevent it from ever moving again. A few sections revised for wording and grammar. PDF releases start (I don't have the patience for multi-platform issues lately)
08/16/00 – Fixed parent # typo.
08/01/00 – Revised ship creation… forgot to upload newer document. Contact information added as section 6. Change Log moved to 7.
06/03/00 – Fixed a lot of typos. Change Log moved to Section 6, Section 5 created.
05/23/00 – Updated section 2, 3, and 4.
05/22/00 – Updated information within section 2
05/16/00 – First copy, basic coverage of creating ships.

A – Yarb's SDB() Crib Sheet

Command	*Function*
set a shipobj debug	sdb(do,sdb,0)
Dump Space.	sdb(function,<anything>,dump)
Repair all damage on a space obj	sdb(function,sdb#,fix)
Manually Iterate Space?	sdb(iterate)
aux.exist	sdb(put,sdb#,<0|1>,aux,exist)
batt.exist	sdb(put,sdb#,<0|1>,batt,exist)
beam.exist	sdb(put,sdb#,<0|1>,beam,exist)
cloak.active	sdb(put,sdb#,<0|1>,cloak,active)
cloak.exist	sdb(put,sdb#,<0|1>,cloak,exist)
engine.impule_exist	sdb(put,sdb#,<0|1>,engine,impule,exist)
engine.warp_exist	sdb(put,sdb#,<0|1>,engine,warp,exist)
alloc.beams	sdb(put,sdb#,x,alloc,beams)
alloc.cloak	sdb(put,sdb#,x,alloc,cloak)
alloc.ecm	sdb(put,sdb#,x,alloc,ecm)
alloc.helm	sdb(put,sdb#,x,alloc,helm)
alloc.miscellaneous	sdb(put,sdb#,x,alloc,misc)
alloc.missiles	sdb(put,sdb#,x,alloc,miss)
alloc.movement	sdb(put,sdb#,x,alloc,move)
alloc.operations	sdb(put,sdb#,x,alloc,oper)
alloc.sensors	sdb(put,sdb#,x,alloc,sensors)
alloc.shield aft	sdb(put,sdb#,x,alloc,shield,aft)
alloc.shield fore	sdb(put,sdb#,x,alloc,shield,fore)
alloc.shield port	sdb(put,sdb#,x,alloc,shield,port)
alloc.shield starboard	sdb(put,sdb#,x,alloc,shield,star)
alloc.tactical	sdb(put,sdb#,x,alloc,tactical)
alloc.tractors	sdb(put,sdb#,x,alloc,tractors)
alloc.transporters	sdb(put,sdb#,x,alloc,transporter)
alloc.version	sdb(put,sdb#,x,alloc,version)
aux.damage	sdb(put,sdb#,x,aux,damage)
aux.gw	sdb(put,sdb#,x,aux,gw)
aux.in	sdb(put,sdb#,x,aux,in)
aux.out	sdb(put,sdb#,x,aux,out)
batt.damage	sdb(put,sdb#,x,batt,damage)
batt.gw	sdb(put,sdb#,x,batt,gw)
batt.in	sdb(put,sdb#,x,batt,in)
batt.out	sdb(put,sdb#,x,batt,out)
beam.banks	sdb(put,sdb#,x,beam,banks)
beam.freq	sdb(put,sdb#,x,beam,freq)
beam.in	sdb(put,sdb#,x,beam,in)
beam.out	sdb(put,sdb#,x,beam,out)
cloak.cost	sdb(put,sdb#,x,cloak,cost)

cloak.damage	sdb(put,sdb#,x,cloak,damage)
cloak.freq	sdb(put,sdb#,x,cloak,freq)
cloak.version	sdb(put,sdb#,x,cloak,version)
Set destination coords	sdb(put,sdb#,x,coords,<x|y|z>,d)
Set ship coordinates	sdb(put,sdb#,x,coords,<x|yz>)
engine.impulse_cruise	sdb(put,sdb#,x,engine,impulse,cruise)
engine.impulse_damage	sdb(put,sdb#,x,engine,impulse,damage)
engine.impulse_max	sdb(put,sdb#,x,engine,impulse,max)
engine.version	sdb(put,sdb#,x,engine,version)
engine.warp_cruise	sdb(put,sdb#,x,engine,warp,cruise)
engine.warp_damage	sdb(put,sdb#,x,engine,warp,damage)
engine.warp_max	sdb(put,sdb#,x,engine,warp,max)
Set ship location	sdb(put,sdb#,x,location)
main reactor damage	sdb(put,sdb#,x,main,damage)
main reactor gw	sdb(put,sdb#,x,main,gw)
main reactor in	sdb(put,sdb#,x,main,in)
main reactor out	sdb(put,sdb#,x,main,out)
missile in/out	sdb(put,sdb#,x,missile,<in|out>)
missile exist	sdb(put,sdb#,x,missile,exist)
missile freq	sdb(put,sdb#,x,missile,freq)
missile tubes	sdb(put,sdb#,x,missile,tubes)
movement in/out	sdb(put,sdb#,x,move,<in|out>)
move.cochranes	sdb(put,sdb#,x,move,cochranes)
move.dt?	sdb(put,sdb#,x,move,dt)
move.empire	sdb(put,sdb#,x,move,empire)
move.quadrant	sdb(put,sdb#,x,move,quadrant)
movement ratio	sdb(put,sdb#,x,move,ratio)
last movement time	sdb(put,sdb#,x,move,time)
velocity, can be used to set ships adrift without engines etc	sdb(put,sdb#,x,move,v)
Object?	sdb(put,sdb#,x,object)
power.aux	sdb(put,sdb#,x,power,aux)
power.batt	sdb(put,sdb#,x,power,batt)
power.main	sdb(put,sdb#,x,power,main)
power.total	sdb(put,sdb#,x,power,total)
power.version	sdb(put,sdb#,x,power,version)
Find the next free SDB #	sdb(empty)
Add fuel to a shipobj	sdb(put,sdb#,<amount>,fuel,<type>)
Load a new ship into space	sdb(read,sdb#)
Set the cochrane constant	sdb(variable,cochrane,x)
Set the dbref of console mode parent	sdb(variable,console,mode,parent dbref)
Set Offset Coordinates, used for relative coords.	sdb(put,sdb#,x,coords,<x|y|z>,o)
Land ship on sensor contact #	sdb(do,sdb,34,contact#)
Launch a 'landed' ship	sdb(do,sdb,35)

Enter anomoly/wormhole sensor contact	sdb(do,sdb,36,1)
Repair system	sdb(do,sdb,39,system)
Repair system X amount	sdb(do,sdb,39,system,x)
Lock tractor beams on sensor contact	sdb(do,sdb,48,contact#)
Unlock tractor beams	sdb(do,sdb,49)
Refuel	sdb(do,sdb,50,…
Engage or disengage Cloaking device	sdb(do,sdb,32,<0|1>)
Set ship cloak frequency	sdb(do,sdb,43,x)
Engage or disengate autopilot	sdb(do,sdb,20,<0|1>)
Set an evasive course away from a sensor contact	sdb(do,sdb,28,contact#)
Set relative coordinates	sdb(do,sdb,37,x,y,z)
Reset relative coordinates to default	sdb(do,sdb,38)
set course	sdb(do,sdb,4,x,y)
Set an intercept course for a sensor contact	sdb(do,sdb,40,contact#)
Set a specific set of coords as ship destination	sdb(do,sdb,41,x,y,z)
set speed (.00-.99 imp) 1-x warp	sdb(do,sdb,5,x)
Adjust ship yaw by X (-360 - 360)	sdb(do,sdb,53,x)
Adjust ship pitch by X (-360 - 360)	sdb(do,sdb,54,x)
Adjust ship roll by X (-360 - 360)	sdb(do,sdb,55,x)
Change ship heading towards currently laid in coords	sdb(do,sdb,56)
Set a parrallel course to a sensor contact	sdb(do,sdb,58,contact#)
set main level %	sdb(do,sdb,1,x)
Set sensor power allocation	sdb(do,sdb,10,ECM,ECCM)
Set operations power allocation	sdb(do,sdb,11,Transporter,Tractors,Misc)
set aux level %	sdb(do,sdb,2,x)
Set overall power allocation	sdb(do,sdb,27,helm,tact,oper)
set batt level %	sdb(do,sdb,3,x)
Set overall power allocation	sdb(do,sdb,6,helm,tact,oper)
Set helm power allocation	sdb(do,sdb,7,movement,shields,cloak)
Set shield power allocation	sdb(do,sdb,8,fore,star,aft,port)
Set tactical power allocation	sdb(do,sdb,9,beam,missile,ew)
Turn long range sensors on or offline	sdb(do,sdb,26,<0|1>)
Turn short range sensors on or offline	sdb(do,sdb,29,<0|1>)
Turn EW on or offline	sdb(do,sdb,30,<0|1>)
Perform sensor scan of type on contact# (list types)	sdb(status,5,13,contact#,scantype)
Display Nebula Report	sdb(status,5,14)
Display Border Report	sdb(status,5,15)
Display Planet Report	sdb(status,5,16)
Display Planet Report (w/out regard to sensor contacts?)	sdb(status,5,17)
Display Condensed Sensor Report (Bots)	sdb(status,5,18)
Detailed Sensor Report for Contact #	sdb(status,5,8,contact#)

Display sensor report	sdb(status,sdb,1)
Raise or lower shield(s) 1-4 correspond to F P S A anything else means 'all'	sdb(do,sdb,16,<1|2|3|4|all>,<0|1>)
Set ship shield frequency	sdb(do,sdb,42,x)
Activate a ship	sdb(do,sdb,12)
Deactivate a ship	sdb(do,sdb,13)
Dock ship with sensor contact	sdb(do,sdb,14,sensor contact#)
Undock ship	sdb(do,sdb,15)
Display ETA at current speed	sdb(do,sdb,33)
Display ETA at a specific speed (.00-.99 imp) 1-x warp	sdb(do,sdb,33,x)
Display Damage Control Status Report	sdb(status,5,10)
Display Science Status Report	sdb(status,5,11)
Display Condensed Status Report	sdb(status,5,12)
Display Total Power Allocation Report	sdb(status,5,9)
Display Engineering Status Report	sdb(status,sdb,2)
Display Helm Status Report	sdb(status,sdb,3)
Display Tactical Status Report	sdb(status,sdb,4)
Display Tactical/Beam Status Report	sdb(status,sdb,5)
Display Tactical/Missile Status Report	sdb(status,sdb,6)
Display Operational Status Report	sdb(status,sdb,7)
Turn tractors on or offline	sdb(do,sdb,31,<0|1>)
Set ship tractor beam frequency	sdb(do,sdb,47,x)
Transporters on or offline	sdb(do,sdb,17,<0|1>)
Lock Transporter on sensor contact as source(0) or destination(1)	sdb(do,sdb,19,sensor contact#,<0|1>)
Unlock transporter from source & dest	sdb(do,sdb,21)
Set ship transporter frequency	sdb(do,sdb,46,x)
Set all weapons offline	sdb(do,sdb,18,0,0,0,0)
Set all weapons online	sdb(do,sdb,18,0,0,1,0)
Set weapons on and offline (needs to be fully documented)	sdb(do,sdb,18,x,x,x,x)
Lock all weapons on sensor contact	sdb(do,sdb,22,0,0,contact#,0)
Lock weapon(s) on sensor contact (needs to be fully documented)	sdb(do,sdb,22,x,x,contact#,x)
Unlock all weapons	sdb(do,sdb,23,0,0,0)
Unlock weapon(s)	sdb(do,sdb,23,x,x,x)
Set ship beam frequency	sdb(do,sdb,44,x)
Set ship missile frequency	sdb(do,sdb,45,x)
Fire Weapon(s)	sdb(do,sdb,52, …
Defuel	sdb(do,sdb,51,…
Angular velocity from sdb1 to sdb2	sdb(function,sdb1,angular,sdb2)
Bearing from sdb1 to sdb2	sdb(function,sdb1,bearing,sdb2)
Range from sdb1 to sdb2	sdb(function,sdb1,range,sdb2)

Set a shipobj's docked or landed location	sdb(put,shipsdb#,locsdb#,loc)
Get a ships coordinates in space	sdb(get,sdb#,coords,<x|y|z>)