

## Editing Scenario Files in America Invades

Q: What is a scenario file ?

A: A scenario file (.SCN suffix) is the file the America Invades program reads into memory to start a game. AmInv comes with 7 .SCN files. This tutorial deals only with editing the CAMPAIGN.SCN file. When you save a game in progress, it saves the game as a .SAV, which is quite close in structure to the .SCN file it was generated from.

Each .SCN file is a collection of numbers (bytes). It doesn't look like much, but the game engine reads this data and stores it in RAM when you start a new game. The game engine is dumb; all it does is read the data from the .SCN file. Thus, we can make changes in the .SCN file and the game engine will read it. However, if we make mistakes while making changes, the game engine can bomb.

A few important points :

- The computer deals with hexadecimal ("hex") numbering. Instead of 1 through 10, it goes 1 through 16 as such : 0-1-2-3-4-5-6-7-8-9-A-B-C-D-E-F. Thus, while "20" means "twenty" in base10 (what we normally use), it has a value "32" in hex. A useful utility on the Mac for converting, adding, subtracting hex and base10 is "SciCalc" also on InfoMac.

- In order to examine and change unit values in AmInv, you will need some sort of hexadecimal editor. For the Mac , I recomend HexEdit, which is available at InfoMac sites. For the PC, there a slew of them, just pick one.

- The hex displays are rows and columns of "bytes". A byte consists of 2 digits.

- My platform focus is from a macintosh point of view, although the .SCN files themselves are identical on both platforms.

- Although I talk about AmInv, Stalingrad uses a similar style of .SCN to store the information. There are some differnces though, and I leave it to a reader to find them.

This tutorial is designed to give the reader a basic knowledge on how to edit and change the CAMPAIGN.SCN file of AmInv. While there are limitatrions on just what we can do, enough is known to make very different scenarios.

Each scenario file (not just the CAMPAIGN.SCN) is a string of bytes. The .SCN file is ordered in this general fashion :

Top of File



End of File

Header and misc info  
Commander instructions  
Names and placements of victory locations  
Terrain descriptions  
Terrain permissions  
Partial terrain and overlays  
Units information  
Leaders  
Misc (not explored)

Here is a general description of where particular parts of the data are in the .SCN file (some of the addresses are not exactly known) :

byte address

meaning

000000 - ~00005F

header..contains start date and end date

~000220 - ~0003DF

start date, end date, axis skill level , allied skill level

~0003E0 - ~ 000AEF

commander instructions

~000BE0 - ~00118F

names that appear on the map

~001220 - ~00125F

average temperature, start turn, end turn, weather  
forecast in days, starting calendar dates

001266 - 0066FF	whole hex terrain readout (2bytes/hex)
~006700 - ~00810F	map info ....not sure
00812C - 012A63	permissions..stacking, ownership, victory locations
012BE0 - 01807D	partial map terrain
~018140 - ~ 01A300	map info of some sort
01A31E - 02C0A9	US unit info
02C0AE - 03E649	German unit info
~03E850 - ~03E90F	temperature
~03E650 - ~03E7AF	weather info
~03EF90 - ~03F22F	victory locations
~03F4D0 - ~003F6BF	replacements
~040F00 - ~041D50	HQ info
xxxxxxxxxx	leaders

In more detail, here is what each of the above sections does :

<u>byte address</u>	<u>meaning</u>
000000 - ~00005F	header..contains start date and end date

byte 000048

This byte is the starting turn. In CAMPAIGN.SCN it has a value of 59. Since a value of 59 represents turn 1 , a value of 5A would represent turn 2. Bytes 001238, 00123C, 001244 seem to have the same value; I recommend setting all these values to the same.

byte 0004C

This byte is the ending turn. Ending value - starting value = # of turns. This value can be set to a maximum of FF. Byte 001240 seems to have the same value, I recommend setting both to the same.

~000220 - ~0003DF axis skill level , allied skill level, estimated time

I have lost my notes on this area. From memory, this area deals with the axis and allied skill levels as well as the estimated time it takes to play a game. Cosmetic.

~0003E0 - ~ 000AEF commander instructions

This area is a text readout of the commanders instructions. You can change it but I am unsure how long they can be. Cosmetic.

~000BE0 - ~00118F names that appear on the map

Once again I did have good notes on this..but now lost. This area is a readout of what names appear on the map. Also in this section is the font size of the name, coordinate position, and color.

~001220 - ~00125F average temperature,start turn, end turn, weather  
forecast in days, starting calendar dates

bytes 001238, 00123C, 001244

Starting turn. These bytes seem to have the same value as byte 000048. Keep them the same, I think.

byte 1245

The average (starting?) temperature. Small changes in this value seem to give great changes in temperature. Not sure if this really has an effect.

byte 001240

Ending turn. Apparently the same as byte 00004C. Keep the same ?

As for the rest in this section, my notes are too fragmentary, but they are in here.

001266 - 0066FF whole hex terrain readout (2bytes/hex)

This, I have mapped out much better. Each whole hex has two bytes that describe it. (Partial terrain is in a later section which includes terrain spillover into other hexes). In the CAMPAIGN.SCN, the first whole hex terrain (top left corner) is at 001266-001267, and the last hex (bottom right corner) is at 0066FE-0066FF. Between each row of terrain, there is a 2 byte fence of 00 00.

Here are the byte locations of the victory locations hexes :

Cherbourg 00223A-00223B

Valognes 002FAA-002FAB

Montebourg 0031F0-0031F1

Carteret 003A9E-003A9F

St. Saveur 003B80-003B81

Carentan 0043D2-0043D3

la Haye du Puits 004466-004467

Isigny 00432A-00432B

Treviers 004582-004583

Lessay 004BD2-004BD3

Periers 004EDC-004EDD

St.Lo 005732-005733

Caumont 005BD0-005BD1

Coutances 005CEE-005CEF

I still cannot find a pattern of what hex values give what terrain types, but here are some examples of terrain with no roads, rivers, hills, etc...

Bocage : 00 00, 0C 00, 48 00

Forest : 1C 00, 8F 00

Clear : 5A 00, 61 00, 14 00, 33 00, 1F 00

airport : 36 02

Swamp : AC 01

Town : 16 00

Pillbox (in clear) : 83 04

City : 87 01, A3 01

There are several slightly different graphical representations for each type of hex.

I cannot find a pattern on how to get the roads/hillsides/rivers drawn. There should be a pattern there, but I cannot see it (these may be in a different section though).

~006700 - ~00810F

map info ....not sure

I think this section deals with the map in some fashion. Small changes seem to create large effects.

00812C - 012A63

permissions..stacking, ownership, victory locations

Terrain permissions :

Each hex on the map has 4 bytes associated with it in the section we call the "terrain permissions". In the CAMPAIGN.SCN, the permissions of the first map hex (upper left corner) are at 00812C - 00812F and the permissions of the last hex (lower right corner) are at 012A5C - 012A5F. Each of the 4 bytes in a hex record have some sort of importance. Between each row of hex-permissions there is a 4 byte fence : 00 00 01 00.

For example, if we examine the permissions for the Cherbourg Victory Location hex :

30 C1 07 40

Here is my interpretation :

-The first byte ("30") appears to correspond to the ZOC value generated from that hex. In this case, "30" seems to mean a ZOC value of "3". Allied hexes have different values.

- The second byte appears to be a constant for hexes controlled by the axis. Allied controlled hexes have a different value.

-The third byte is how many stacking points are currently in the hex.

00 = empty allied  
01 = empty axis  
02 = 1 allied stacking point  
03 = 1 axis stacking point  
04 = 2 allied stacking point  
05 = 2 axis stacking point  
06 = 3 allied stacking point  
07 = 3 axis stacking point  
08 = 4 allied stacking point  
09 = 4 axis stacking point  
0A = 5 allied stacking point  
0B = 5 axis stacking point  
0C = 6 allied stacking point  
0D = 6 axis stacking point  
0E = 7 allied stacking point  
0F = 7 axis stacking point  
10 = 8 allied stacking point  
11 = 8 axis stacking point  
12 = 9 allied stacking point  
13 = 9 axis stacking point

-The fourth byte seems to only have three different values :

00 = normal

30 = allied reinforcement hex (?)

40 = victory location brown hex ring

This section of the file actually encompasses many things. In this section each map hex has 4 bytes that give information on that hex, with such information as :

- who owns the hex
- is this hex a victory/supply location
- how many stacking points are here
- what units are sitting in this hex

Here are the hex addresses of all the victory location hex permissions :

Cherbourg 9F54-9F57

Valognes BBB4-BBB7

Montebourg C040-C043

Carteret D19C-D19F

St. Saveur D360-D363

Carentan E404-E407

la Haye du Puits E52C-E52F

Isigny E2B4-E2B7

Treviers E764-E767

Lessay F404-F407

Periers FA18-FA1B

St.Lo 10AC4-10AC7

Caumont 11400-11403

Coutances 1163C-1163F

Some other important hex permissions :

top Utah beach invasion hex CDD8-CDDB

middle Utah beach invasion hex CF54-CF57

bottom Utah beach invasion hex D0D4-D0D7

This section of permissions is why you cant just change a units location (see editing units below) without the unit disappearing. This area does not appear to contain data on supply points for either side.

012BE0 - 01807D partial map terrain

If you examine the map carefully you will notice that most of the map hexes contain a small maount of terrain spillover from adjacent hexes. This is due to partial terrain. Each partial terrain layover is 2bytes/hex. I have not mapped out the permutations..

~018140 - ~ 01A300 map info of some sort

Again, I am unsure of this section except that it deals with the map (I think).

Oddly enough, I have yet to locate hilltops in the .SCN file or ship bombardment hexes.

01A31E - 02C0A9 US unit info

This section deals with the US units, air ground and sea. Each unit in the game had a record of 172 bytes. I will deal with this section in great detail below.

02C0AE - 03E649 German unit info

This section deals with the axis units, air ground and sea. Each unit in the game had a record of 172 bytes. I will deal with this section in great detail below.

~03E850 - ~03E90F temperature

Not well mapped out. See Ciril's notes.

~03E650 - ~03E7AF weather info

This section contains the typical weather for each day (I think).

00=clear

01=light overcast

02=medium overcast

03=heavy overcast

04=storm

~03EF90 - ~03F22F                      victory locations

This section contains the names of the victory locations, and presumably the value. I do not have this mapped out well.

~03F2C0 - ~03F4C0                      leaders

This contains info on the leaders including thier attached division and value.

~03F4D0 - ~003F6BF                      replacements

See Ciril's file.

~040F00 - ~041D50                      HQ info

This area contains some information on every HQ in the game, such as whether the HQ is currently on the map or not. I think this area might contain supply information, but am not sure. If you wanted to make a scenario that had a differnt number of HQ start on map for either side, this section would be crucial (I need to find some older notes).

Obviously there are things missing; this is very much a work in progress. Anything you can find on your own, take good notes (unlike me).

#### .SCN and .SAV files

As it turns out, the .SAV files generated when you save a game are virtually identical to the .SCN file it is derived from. In fact, they differ in size only by 28 bytes; and that happens to be the first 28 bytes in the .SAV file. If you delete the first 28 bytes of the .SAV file, you will have a file that is exactly the same size (and have the same hex addresses too) as the .SCN file it was derived from. So, knowing that the .SAV and .SCN are virtually identical, AND knowing that the most important information is kept in the permissions and unit information sections, this allows us all we need to create new .SCN files. Read on !

"I want to make a new scenario that involves changing the hex placement of units that start on the map."

This is one of the simplest things we can do to essentially create interesting variants of existing scenarios. I have done this myself, in creating a scenario that swaps the at start positions of the 709 and 352 German divisions in the campaign game. To do this, this is an overview of the process :

- A - we estimate the # of game days it will take to move all the units around
- B - reinforcements that would arrive during that time will have their arrival time moved back so they dont show up
- C - start a new 2 player local game using the altered .SCN using all the variants you want
- D - play the game you started, but only move units around you want moved...leave all others alone
- E - once you have all the units in the new places, save the game during the next 6:00 AM turn
- F - remove the first 28 bytes of the save file, then cut and paste the permissions and unit information sections INTO a fresh copy of CAMPAIGN.SCN
- G - Cleaning up - moving the reinforcements back to thier normal time of entry

It may sound daunting, but it is reltively easy to pull off, although a bit tedious in places. Now for a closer look at the steps :

### **A**

This is pretty self explanatory. For example it took me 4 game days to switch the places of the 352nd and 709th divisions in one hacked .SCN. For the campaign game, 5 days is about the most time you should need.

### **B**

Knowing that there are 6 turns per day, you can calculate which reinforcements need to be moved back in time. For example, if you need 3 days, that is 18 turns. Since turn 1 is "58" in the turn track, and unit that arrives on turn : 58 (this is a hex value) + 12 (18 converted to hex) = 6A. So , any unit that arrives on 6A or earlier should be set back. I recommend setting such units back to "99", as this value is rarely used, and thus easy to find later.

I suppose I should mention why this is done, as this is the tedious part. As you play your set up game, reinforcements will enter the board, and sit on the reinforcement hexes, as you probably dont want to move them. Now when you cut and paste the proper sections into the fresh CAMPAIGN.SCN and start your new scenario, the reinforcements that had arrived are no longer there, because they arent scheduled to arrive until the proper time.

However, the permissions of those entry hexes still thinks there are units in those hexes, and so no unit can enter those hexes if they were stacked full. As the allies, this means your beachhead entry hexes are clogged up and no reinforcements will arrive. Big problem. For some reason the engine doesnt do any "garbage collecting", checking each hex to see if there are really units there, every turn. Sure would make our lives easier. SO, the easiest way (not the only way though) is to prevent reinforcements from arriving.

Later, when you are almost done, you have to re-set the proper turns for the reinforcements.

### **C-E**

After you have altered a fresh copy of CAMPAIGN.SCN (by moving reinforcements back in time (above)), you start a 2 player, local game with whatever variants you want (although only variants that involve units are like to come through the whole process intact). Move the selected units around the board until you have them placed where you want them. If you want to attach units around, make sure to do this before you save the game later.

Dont forget that certain units start the game out of supply. If you dont airsupply them , they may surrender and remove themselves from the game. Be careful not to ambush your moving units, this may deplete or kill them off. Also realize that units that arent moving will decrease in dis/fat values - so you may have to adjust these values later on. Once every unit is in the right place, make sure all units have defend orders. Not doing so can cause problems later. . Save the game during the next 6:00 AM turn. This is so each HQ has supplies delivered to it before you save.

### **F**

Make a backup of the .SAV file you just generated. Take one of the .SAV files and delete the first 28 bytes using the hex editor of choice.



reinforcement HQ start the game on board (unless you can figure out the section on HQ late in the .SCN file).

- If you get a "Bad Read" error in your new .SCN that means you forgot to set some unit/s/ in the .SAV to defend orders.
- You can have units that start on board enter as reinforcements by setting their turn of entry ("58") to a corresponding higher value. In that case you might also want to change the hex placement. Unlike units that start on map, reinforcements can pop up just about anywhere and not cause problems with the permissions.
- Dont forget to reset dis/fat levels for units that you want to start with such.
- When you use the new .SCN, some variants should not be used. For example, in the .SCN I created where the 709 and 352 switch, if I use the 11th Airborne variant, it basically overwrites and screws my changes (not permanently, only in play).
- It is possible to have a hacked .SCN file that starts with a different number of HQ on map. For example, if you wanted an additional HQ to start on map, you would have to find its entry near the end of the .SCN file and paste into an entry of another HQ that starts on map.

Stalingrad : While I dont have SG fully figured out, here is some of what I do know.

If you want to edit SG, you should follow the same steps as above except :

- the difference in size of the .SAV and .SCN is unknown
- in the .SCN files, the permissions section is 00649C-00FF47 (I think), and the unit information is 0100E2-03D70F (I think) . Try it out.

Included files :

Besides this pdf file, I have included a text file from Ciril Rozic who has been working in conjunction with me. When in doubt, compare both of our notes.

I have also included several altered .SCN files for illustration. You will need to rename them all to "CAMPAIGN.SCN" (no quotes) before using them in place of the original CAMAPAIN.SCN file.

Here is a short description of each :

file One: This scenario changes the place of the 709th and 352nd divisions. Mac or PC.

folder Two: this folder contains a big change from the normal file. It has new graphics for new units; however, only Mac users can use it and only after doing some resedit work. The PC version had all of its graphics stored in a file I am unable to read.

folder fixme: this folder contains a fix (mac only, see above) that corrects a few cosmetic errors in the NATO style icons. Needs ResEdit.

file three: this scenario simply adds the 30th Schnelle HQ to start the game on map.

file four : changes all the hexes on the map to clear terrain, no partial terrain.

These files illustrate just how much you can do, with a bit of perseverance.

If you have any questions, email me at :

gionpeters@interjetnet.net

And as an added bonus, here is some game information gleaned by Jeff Wrana :

#### GAME timing (by Jeff Wrana)

Here is topic that is not documented in the manual very much at all but is very important. It come down to , 'when' to things occur during the execution phase. This is what I have figured out.

- 1) All arty and ship attacks occur after exactly 5% of the turn has elapsed.
- 2) fighter barrage attacks : 20% elapsed
- 3) Allout assaults : 50%
- 4) Assault: over 90% elapsed
- 5) bomber attacks : over 80%
- 6) probes : 95%
- 7) transport drops : very end

The 2 -7 numbers are only accurate if no arty or ships were involved

This means that for example, You use a bomber to interdict some troops, well guess what, they get to use 80% of their movement before the bomber ever gets there, hence you'll miss and waste a bomber.

Same for fighters, he gets to use 20% of his movement before the plane

hits. An example of how far a unit can move using 20% of his total. The 12SS mech can has  $32 \times .2 = 6.4$  movement points before any of the fighters bombs hit the ground. Since he can move 4 primary road hexes per point, this means he can move 25 road hexes before the plane drops. Obviously, plotting interdiction airstrikes too close to motorized units means guaranteed misses.

Same for arty and ships, he gets to use 5% of his movement before the shells hit. This is why, Doug, you miss my rocket arty 75% of the time when going for it. For example an arty with 30 mp's has  $30 \times .05 = 1.5$  mp's to move before being hit. This is equal to 6 road hexes before the shells hit. Your long gone.

There is one very major catch to this whole thing. Its very important. Using arty or ships in ANY attack means the attack will happen with only 5% of the turn used. Use a bomber to intradict a road with a long ranged (22) arty. The attack happens almost right away. Nothing will make it far through that hex IF IT LIVES TO TELL !!!!. Assault a hex with 1 points of ship strenght. The attack happens right away. Any retreat means an advance(unless ZOC is too high).

It comes to this. Every assault you make, have at least .1 measley points of art in it to trigger the easly assault.

Used on close interdiction with planes, the same thing. (Except I guess I am in charge of planes so I'll take care of it)

## Editing the Units in America Invades

The .SCN files (and the .SAV files too) have a general organization that is only broadly understood (see below) at the moment. For now, I will talk only about the section on Unit information. The other sections I will talk about later. It is easy enough to make changes in this file, but you have to remember to save any changes before having AmInv read the file.

These files are pretty touchy. Accidentally adding or deleting even a single byte in some places can cause AmInv to crash. You must be careful, and make numerous backup copies.

OK, here we go. Lets take a look at part of a page from the CAMPAIGN.SCN in AmInv :

CAMPAIGN.SCN												
Length:	437ACh	Type:	uSCN	Creator:	uAGI	Fork:	data					
000000:	30 12 00 00 11 00 00 00 05 00 00 00 0A 00 00 00						0.....					
000010:	08 00 00 00 05 00 00 00 08 00 00 00 00 00 00 00						.....					
000020:	0A 00 00 00 14 00 00 00 05 00 00 00 7D 00 00 00						.....}					
000030:	64 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00						d.....					
000040:	00 00 00 00 00 00 00 00 59 7C 01 00 FF 7C 01 00						.....Y ... ..					
000050:	97 54 00 00 28 60 00 00 00 00 00 00 00 00 00 00						.T..<.....					
000060:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00						.....					
000070:	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00						.....					
Left	Center					Right						

We are looking at the top of the hex file as shown by HexEdit. The read-out consists of 3 separate parts. On the left is the hexadecimal address. The Center contains the data that we want to look at. The Right is the ascii translation of the hex data in the center. See the "FF" byte ? That byte is at address 00004C (just a little practice there...). Want some more ?

Lets now look at a sample set of hex data for a single unit. If we scroll down to hex address 031140 or so, we will look at the II-920-243 infantry battalion :



consumption of a unit seems proportional to its strength, so if you increase the attack/defense values, you may have to increase this supply tonnage to compensate, otherwise the altered unit will start with an initially lower supply state than normal.

### C

This 2 byte sequence is the current attack strength. 2 bytes are requires as this value is floating point (K.Z. told me), but is rounded and displayed as an integer value in the game. Although I cannot make sense of why these 2 bytes readout the way they do, it is easy to map out which float values give what integer results :

Float Value	Integer Result		Float Value	Integer Result
E8 03	1		40 1F	8
D0 07	2		28 23	9
B8 0B	3		10 27	10
A0 0F	4		F8 2A	11
88 13	5		E0 2E	12
70 17	6		C8 32	13
58 1B	7		B0 36	14

The current attack value should always be equal to or less than the maximum attack value (see **E**). If the current attack is less than the maximum attack value you can take replacements. This value does not take into account supply level or terrain or other factors, which the game calculates on the fly.

### D

This 2 byte sequence is the current defense strength. It functions exactly like the current attack value, but involving the defense strength.

### E

This 2 byte sequence is the maximum attack strength. This is the highest raw attack strength the unit can get (not counting supply or other factors).

### F

This 2 byte sequence is the maximum defense strength. This is the highest raw defense strength the unit can get (not counting supply or other factors).

## **G**

This 2 byte sequence is the current armor strength. Our example shows "00 00" as it is not an armor unit. Only armor/SP-AT/Assault gun/mech recon/mech infantry units should have armor values. Otherwise this functions exactly like attack and defense strength

## **H**

Current anti-tank (max ? normal?). Im a little hazy on this one. If it follows the sequence that other value do , this should be the current AT value (see **J**). Anyway, changing this value will change the current AT value, in the same fashion as attack and defense strengths.

## **I**

This 2 byte sequence is the maximum armor strength. Our example shows "00 00" as it is not an armor unit. Only armor/SP-AT/Assault gun/mech recon/mech infantry units should have armor values. Otherwise this functions exactly like attack and defense strength.

## **J**

Maximum anti-tank value (?). While logically this should be the maximum AT value, changing this seems to have no effect at all. Maybe the game calculates AT strength reductions due to strength losses on the fly and doesn't need a maximum value.

## **K**

Movement value. This is also a floating point and uses the same values as attack/defense strength.

## **L**

Macintosh :

This 2 byte value tells the game which counter icon to display for the unit. The actual counter icons are in the PICT directory of the Resource Fork of the AmInv program. To see them , you will need ResEdit or some similar tool. Below is a part of the PICT resource, ID=307 (mac). The counter icons are basically stacked together, hub-to-hub. See the 7A counter ? It has a value of "00 00", or the very first icon. If we wanted to change the II-920-243 to display the icon for the SB-7A we would change the value of "78 00" to "06 00". The 2 bytes are needed as there are more than 256 icons ( a single byte can have a max value of 256). The 257th icon in this PICT resource would have a value of "00 01". You can have a unit show any icon, the game doesn't care if more than one unit shows the same icon. Also note that the 2 bytes preceding have the same value, but I think they are dummies; changes seem to have no effect.



PC :

I think that all the graphics are stored in the PCWAW.REZ file. I am also guessing that there exists a program that takes the entire macintosh resource fork and converts it into a single binary file. What this program is, I dont know. I do know it is not a common every-day program. If we can find out what the program is, we could make changes to the mac version and convert to the PC. Otherwise, I dont see a way to change the graphics in the PC version.

### M

This sequence of 4 bytes records the units current position on the map. The first byte ("0F" in this case) records the number of hexes from the left border. The third byte ("19") is the number of hexes from the top border. The other two bytes ("00") are spacers and do nothing. This sequence is like an X,Y coordinate system. You can change where a reinforcement arrives by changing these values. Attempts to change the placement hexes of units that start on the map requires more work involving other sections of the .SCN file.

### N

This similar sequence as "M" above may be the units destination hex. Or maybe this section is the units placement hex and the other is a dummy. Or maybe this is a dummy. To make things simple I keep this the same as the previous (M=N). Artillery units sometimes have "FF FF FF FF" for this sequence (?).

### O

This 2 byte sequence relates to the units position on the O.O.B. display from the toolbar. Every unit has a unique (maybe) value for this number which tells where it appears on the OOB display. Mess with this and then accessing the OOB display will likely cause a crash. I dont logically see why this value even is needed. I dont mess with this. If you want to have different unit attachments in the game, it can be done..keep reading.

### P

Unit type. These two bytes determine what type the unit is. The first of the 2 bytes is the general type :

First byte value	Type	First byte value	Type
0	infantry	5	ship
1	armor	6	plane
2	engineer	7	HQ
3	anti-tank	8	flak
4	artillery		

The second byte is the specific unit type. There are lots of specific types, too many to list (maybe 75-100, not all I understand), but here is a small sample :

Second byte value	Specific type	Second byte value	Specific type
0	none ?	5	mot light
1	airborne	6	mot heavy
2	heavy	7	mech recon
3	semi motorized	8	glider
4	motorized	9	fortress

These two bytes determine special abilities. For example, changing the general type to engineer ("02") gives that unit the ability to build fortifications.

### Q

This byte is the strategic movement multiplier. When using strat movement this value multiplies the movement value (**K**). These values may not be completely correct and may vary with the units type and movement value. There are probably other values but I have not thoroughly explored it yet.

Byte Value	Multiplier
00	x2 or x1.5
01	x2
06	x2.5
07	x2
08	x2
10	crashes

## R

This byte determines the size of the unit, for the purpose of stacking a digging-in (are there any other uses for this value?). A company size unit should have a value of "01". while a battalion should have "03". You can also use "02" or "06" if you want to create an oddly sized unit.

## S

These 4 bytes determine which HQ the unit is attached to. At the beginning of the game only the first and last byte have a value other than "00". During play they change to all having the same value. I don't know why. In any case, I do not recommend changing these values with the hex editor. Doing so will cause crashes because you would also need to change the values in section O also. There are ways around this, but I will talk about that later.

## T

This appears to be the unit's current order type. I think "00" means defend. I don't have this mapped out.

## U

These 3 values are related to the morale of the unit. The first of the 3 is never changed (base morale?) while the 2nd and 3rd can change from circumstances (supply, etc). When altering morale values give the same value to all three. I have a theory that the second value is related to attack success, while the third is defensive success, but I haven't explored that.

## V

Disruption and fatigue. The first byte is the disruption level of the unit and the second is the fatigue level.

## W

Current supply level. There are 5 values :

Byte Value	Supply Level
0	none
1	minimal
2	defensive
3	general
4	attack

Changing this doesn't necessarily change the unit's supply state; you may have to alter supplies on hand **(B)**.

## **X**

Unit name. The unit's name starts after the "FF 00" 2 bytes. I don't know what the maximum length is but is at least 10 characters.

What's Left ? You can see there are many non "00" bytes that I don't detail, as I don't know what they are for. Some may be dummies or bogus.

Artillery Units : Artillery units seem to have the same layout as ground units but oddly enough (why did they do it this way ?) you cannot change the barrage and range values by altering the unit record. Instead, I think the range and barrage values are determined by the relative position of the artillery unit record from its parent HQ. For example, if you look at the 5-1st and 7-1st infantry artillery battalions. One has a range 14, barrage 7, while the other has range 10, barrage 4. Nothing you do in the unit record will change those values. But if you switch the 2 unit records in place, they will switch barrage and range values.

This is really odd because I know that in Crusader the range value is in the unit record.

I don't know about Stalingrad.

Air and Naval units : I haven't looked into these, but they should follow the same general pattern as the ground units.