Silent Hunter II
Scenario Hand Editing Manual

Introduction

Hand editing Silent Hunter II mission files and creating new missions is not very difficult if you get to know the system. The best way to get started is to copy an existing mission, rename it, and use parameter blocks from other missions to change things up. We always test our missions extensively in the game before considering them complete. Making sure the ships and planes go where they are supposed to, timing the radio messages, and populating the players assigned patrol grid with a realistic number of entities will be necessary for a quality mission. To observe how the AI units are behaving, turn the Limited Visibility option off. It also helps to set the Vulnerability option to 10% or so in order to survive an attack and continue testing the mission. I recommend keeping the fuel and dud torpedo options on, to gauge the effects your mission has on the player's operating range.

After you spend hours making, testing, adjusting, and testing some more, your mission won't be exactly thrilling or unpredictable to you, the creator. In a sense, you are endeavoring to make missions for other players; you will profit by getting missions from other players.

Please e-mail any completed missions to us at **subsim@subsim.com** so we can share them with the community. Zip up the mission files, radio messages, and briefing file with a short readme including your name and e-mail. Your efforts are appreciated.

SHII Mission files are stored in
C:\Program Files\SSI\Silent Hunter II\Shell\Scenarios

The briefing text files are stored in
C:\Program Files\SSI\Silent Hunter II\Text\English

The radio message text files are located in
C:\Program Files\SSI\Silent Hunter II\Text\English\Messages

Good luck and good hunting!
Neal Stevens
SUBSIM.com

About the Sim Program

The Silent Hunter 2 game consists of two main programs. The top level program, from which all others are run, is the game shell. The game shell can then spawn various other programs including the main game engine itself.

The game shell is encountered when Silent Hunter 2 (SH2) is first run, and can prompt the user to play or continue a single mission, historic mission, or the main campaign. In each of these situations the game shell then spawns the game engine and provides the game engine with the parameters needed to execute the selected mission. These parameters are passed from game shell to game engine via a text document called a scenario file (SDF). The game shell specifies the SDF (or even creates it), and the game engine then reads the SDF and parses out individual parameters.

The manual you are now reading describes the components that make up an SDF, and describes how to create scenarios from scratch or modify existing scenarios.

Scenario Header

Each SDF file contains an initial comment header (with leading semicolons ";") that is not read by the game engine. You may add to, or edit, this header.

Scenario Parameter Blocks

The rest of the SDF file consists of a series of parameter blocks that are read by the game engine and which give specific information about where the game takes place, the starting date, weather, group and unit descriptions, game goals and other information. Each of these blocks contains additional parameters, one parameter or parameter group per line. Here is a parameter block that describes when and where the scenario takes place:

```
[SCENARIO]
Date = 19410928
Time = 1600
Duration = 1500
Type = TYPE A
```

```
Major = 10
Minor = 5
Origin = -23.744507,15.349678
InitialZoom = 9
Weather = HIGH_CLOUDS
```

Here is one of several blocks that describe an individual unit

```
[UNIT1]
Group = 1
Class = NSSUBoat9C
Name = U-176
Control = HUMAN
Flag = TRUE
Guide = TRUE
Location = -6.300000, 52.150002
Orientation = 311
Captain = 80
Crew = 80
```

The easiest tactic is to duplicate and rename an existing full scenario and then modify it.

When editing an SDF file, most any editor may be used as long as the final product remains a standard unformated ASCII file. Wordpad has been used successfully.

Parameter Protocol

This might be a good time to mention that, when hand editing an SDF file, the parameters blocks may be placed in any order. In addition, the individual parameters within each block may be placed in any order (within that block).

The SH2 game engine (sim) does not care what order things are in. But, in the case of redundant or missing parameters within a single block, and redundant or missing blocks within the SDF file, the sim will follow these rules:

1. Only the first item in a redundant pair will be read.
2. Missing items will cause all subsequent numbered parameters of that type (within a block) and all subsequent blocks of the same type (within an SDF) to be skipped.

For example, in the following set of 3 blocks, only the first and third blocks will be read.

```
[UNIT1]
...parameters...
```

```
[UNIT1]
...parameters...

[UNIT3]
...parameters...
```

In the following example, only the first block will be read.

```
[UNIT1]
...parameters...

[UNIT3]
...parameters...

[UNIT4]
...parameters...
```

As you might surmise, it's pretty easy to delete a block by hand and forget to renumber the subsequent blocks, or to replicate a block to add, say, another unit, and then forget to renumber those.

Units

The units available to place in a SH2 scenario include surface warships, submarines, military aircraft, merchant ships and liners.

Each of several nationalities is represented by it's own database of units. These include American, Britain, France, Germany, Denmark, Sweden, Poland, Norway, Finland, Soviet Union and Romania. Each database is a simple ASCII text file and is located in SH2 / DB as UnitsAm.db, UnitsBr.db, etc.

Please do not alter the databases themselves without caution as their data is in a specific order. The first few fields in each record (line) must remain in alphabetical order (top to bottom) for the simulation search engine to work properly. A comma separates each column (field). Otherwise your changes will be lost whenever you update to a new version of SH2.

Here is a sample of the American database. (The end of some lines has been shortened in this example.) In each line the fields, separated by commas, are <nation><merchant, naval, etc><unit type><class><model name><unit name><date in service><date out of service><sequential number [not used]><alternate class to use> :

```
Am,M,SC,Slow Cargo Ship,MLLCargo,Slow Cargo,0,99991231,1056,n/a
Am,N,BB,Iowa,NBBIowa,BB61 Iowa, 19430222,19800101...
Am,N,BB,Iowa,NBBIowa,BB62 New Jersey,...
Am,N,BB,Iowa,NBBIowa,BB63 Missouri,...
Am,N,BB,North Carolina,NBBNCarolina,BB55 North Carolina,...
```

The databases are a convenient place to browse the available unit types and specific unit names (where applicable). Note that you can manually copy a "model name" (located in the 5th column) and "unit name" (6th column) from the database to modify an existing SDF file.

Here is an example of that part of a SDF file that would contain the unit and model names:

```
[UNIT3]
Group = 3
TypeAbrev = BB
Class = NBBIowa
Name = BB61 Iowa
Flag = TRUE
Guide = TRUE
Location = -22.675522, 47.367188
Orientation = 118
Captain = 80
Crew = 80
```

When manually changing a unit, keep in mind that all units in a group must be of the same type. That is, all units in a group must be battleships (BB), or cruisers (CA), etc. In the above example you can change the BB class "NBBIowa" to "NBBTexas", or " BB61 Iowa" to " BB62 New Jersey. Hopefully this description gives you a sense of how scenario files describe units.

Warning: If you have two or more units of, say, class BB in a group and you change one of them to another class, such as CA, it will have unpredictable results. Instead, add a new CA group, or add the new CA to an existing CA group. Be careful when hand editing a file.

If you mean to create a convoy with several types of merchants and an escort screen, you will want to make a Group for one class of ships and add some supporting groups for each of the other class of ship and the escorts. Here is an example of a convoy of one class of ship with an escort:

```
[GROUP8]
Type = SHIPSQUADRON
UnitType = Merchant
Side = 1
Name = Convoy HV-217
Nationality = Britain
Order1.1 = Speed,Telegraph,TWO_THIRDS,-1,-1
Order1.2 = WayPt,-34.469341,45.884911,-1,-1
Order2.1 = WayPt,-34.552677,46.490124,-1,-1
Order3.1 = WayPt,-33.404758,48.860954,-1,-1
Order4.1 = WayPt,-32.023510,50.105747,-1,-1
Order5.1 = WayPt,-30.961010,49.668247,-1,-1
```

```
Order6.1 = WayPt,-8.758926,57.923454,-1,-1
Order7.1 = WayPt,-6.756322,55.280746,-1,-1
Order8.1 = WayPt,-7.087573,55.151058,-1,-1
Order8.2 = Speed,Telegraph,STOP,-1,-1
WayptDisplay = DO_NOT_SHOW

[GROUP8.1]
Type = SHIPDIVISION
UnitType = Escort
Name = escorts
WayptDisplay = DO_NOT_SHOW
```

Note that each supporting GROUP will be listed as [GROUP8.1] and [GROUP8.2] and so on. Usually I keep it simple, two class of merchies and one or two groups of escorts. The supporting groups do not need waypoints, they follow the lead group.

Use SHIPSQUADRON for a top-level group that will have other groups attached to it, SHIPDIVISION for most other surface ships.

Surface Warships, Submarines, Aircraft, etc.

Each of the basic units types, including surface warships, submarines, military aircraft, merchant ships and liners, are added to a SDF file, and must be maintained in the SDF file, in the same manner. Again, don't mix unit types, be it airplanes, tankers or war ships.

Only submarine (SS) units may be mixed, and there we have another consideration: In most cases you will still want to have a separate group for each submarine. That way the subs may be controlled separately during game play.   One reason that the other unit types such as airplanes, tankers and war ships, should be in groups is so that the game can control the entire group as one block (more or less). This is more efficient for the game engine, and often better reflects the real life situation.

Waypoints

Get a pad and pencil, fire up an SH2 mission and then use the mouse to lay in your waypoints. Whereever you hover the mouse, there will be longitude and latitude readings in the miniature world map insert, such as 34.43W 46.93N Remember, the longitude reading will change at the Prime Meridian in England. For SHII scripting purposes, the number is negative to the west of the line and positive to the east of the line. So longitudes around the US East Coast to Iceland will be negative and longitudes from Wilhelmshaven and further east will be positive.

Just write these down and fill out the rest of each coordinate with zeroes, so it looks like this:
Location = -33.430000,46.930000

Objectives have the coordinates on two lines

[OBJECTIVE1]
Objective = REACH
Longitude = -33.430000
Latitude = 46.930000

Waypoints are similar
Order1.2 = WayPt,-33.430000,46.930000,-1,-1

Bases

Essentially, bases consist of a location and a radius. Each represents an area on the ocean that emulates Allied or Axis spotter aircraft or other wartime surveillance.

When an Axis sub enters an Allied aligned base area, there is a probability that it will be detected. If detected, an Allied group will be spawned to attack the sub.

Conversely, when an Allied unit enters an Axis aligned base area, there is a probability that it will be detected and a message sent to the sub revealing the Allied unit's type and location.

In the SDF file, an Axis aligned base looks like this. No other data is needed. The Sim will generate the text of the radio message to alert the Axis submarine.

```
[BASEn]
Alignment = AXIS
Location = -8.246877, 49.538544 ; longitude, latitude
AirRadius = 80000.000000 ; meters
AirProbability = 0.500000 ;  0.01 to 0.99 percent
Timer = 12.5; minutes
```

The Axis base is used for those occasional intel reports that pop up a contact icon on the map. Set the Air radius big and the AirProbability low with a 45 minute timer setting if you don’t want real-time intel reports.

An Allies aligned base looks like the following. Note that the only real difference is that a group has been called out also ("Group = 4").

```
[BASEn]
Alignment = ALLIES
Location = -8.261324, 49.173823 ; longitude, latitude
AirRadius = 4000.000000 ; meters
AirProbability = 0.200000 ;  0.01 to 0.99 percent
Group = 4
Timer = 20
```

To make this work we have to have a normal [GROUP4] along with it's [UNITn] designation. And we need to add the following to [GROUP4] (or whatever group we are calling out):

```
[GROUP4]
...other parameters...
EntryTime = 1200
EntryDate = 21000101
EntryInterval = 0130
```

This is a variant of the parameters used for what is called a "delayed entry" group (described elsewhere) only here it need only be a time and date which is some time in the future (or else the unit may be spawned before anything is detected in the base area). The EntryInterval parameters describe how often a unit may be spawned.

Bases are a great way to set randomly spawned air patrols if the player ventures into the area of the base. Really wide radius bases should have a smaller AirProbablity factor or you will besiege the player with attacking planes. I use small high probability bases to defend ports where I dare the player to enter. Keeps the scenario from becoming the sitting duck shoot where ships stop in port and no longer defend themselves as we encountered in SH1 frequently.

Orders

Attached to each group is a default set of orders. Default orders cause the group to go into a formation (column), maintain a speed (two thirds) and live by a default set of the rules of engagement (ROE) orders. The default set of ROE orders is specific to each unit type. Battleships will probably have a different outlook on life then, say, a torpedo bomber.

The default orders may be overridden, and/or additional orders may be added. Available orders include the following:

```
Attack (id of another unit)
Formation (column, line_abreast, wedge, etc.)
Loop (back to a previous way point and it's associated
orders)
Patrol
ROE (Rules of Engagement)
Speed (stop, one_third, two_thirds, etc.)
WayPt (location to reach)
```

When you add a new formation order (changing the units from, say, column to line abreast) you must associate it with a particular waypoint. Another way to look at that is that you can never issue a new order in between waypoints. (The Sim may do that during game play, but you can't script it out ahead of time.) Associating waypoint orders with all other orders is a logical way to think of how to direct the behavior of your unit groups.

Here is how the Sim reads and executes your orders for a single group, at game time:

1. The game starts, the sim looks at the first scripted unit group (a Destroyer Group), and reads all the orders in its first "order set".

```
Order1.1 = Speed,Telegraph,FLANK,-1,-1
Order1.2 = ROE,Submarines,Unlimited,Destroy,Cruising,Full
Order1.3 = WayPt,-23.743465,15.342126
```

Note that most order sets contain one waypoint. The way the Sim processing things is that it reads and executes each order in turn before going on to the next. Since the **WayPt order always falls at the end of each order set** (it is sorted that way) it is the last to be executed. Thus the behavior becomes (in this example)...

a. Set speed to FLANK
b. Set the ROE as requested
c. Go to location -23.743465,15.342126.

The speed and ROE orders are "completed" immediately since they are just setting states. (It may take the unit some time to get up to flank speed, but we do not wait for that to happen before going on.) But it may take some time to get to the waypoint location requested in Order1.3. So all further order set will wait until we reach the appointed location (or until the Sim determines that it wants to direct the group in some other manner, such as when it is attacked by the user submarine.)

Once the first waypoint is reached, the next set of orders is acted upon, and so on. If a loop order is encountered (usually as the only order in an order set) it will direct the Sim to go back and start reading an earlier order block.

Here is the rest the orders for this group. Note that order set 9 causes the Sim to immediately jump back and start executing orders at 2.1. It will execute that loop five times before going on to read and execute Order10.1.

```
Order2.1 = WayPt,-23.759350,15.343948
Order3.1 = WayPt,-23.740341,15.365563
Order4.1 = Speed,Telegraph,ONE_THIRD
Order4.2 = WayPt,-23.693987,15.366605
Order5.1 = WayPt,-23.748154,15.352803
Order6.1 = WayPt,-23.712475,15.374157
Order7.1 = WayPt,-23.678621,15.371032
Order8.1 = WayPt,-23.736174,15.381449
Order9.1 = Loop,2,5
Order10.1 = WayPt,-23.692163,15.383272
```

One or more waypoints must be added to a group before any additional orders may be added.

Static Orders

Orders that set specific static states include the Speed order, Formation Order, Profile Order and the Rules of Engagement (ROE) order. Once set, those states persist until another explicit order, or the A.I. code in the Sim, resets them.

The Speed and Formation orders specify just that. The Formation order also sets a spread in yards between units. The Profile order allows the mission to specify crew and ship readiness. The ROE order tells the group how to react in certain conditions such as when under attack.

Some orders (shadow, harrass, etc) have limited use and/or designed for future functionality. You'll have to experiment and report back to me to help me compile a list of orders that are fully functional.

Note that all states may be modified by the AI to conform to game rules or common sense. This might be the case when several ships have to pass through a narrow strait while in the line abreast formation, and you have specified a spread between ships that is too large to allow this. The AI might then modify your order, at least temporarily, to bring the ships closer together, or put them in a column formation.

Loop Orders

A Loop order should be placed in it's own "order set". If, instead, you were to place a WayPt and Loop order in the same set...

```
Order9.1 = Loop,2,5
Order9.2 = WayPt,-23.692163,15.383272
```

...to WayPt order would not execute. Actually it might after all the loops had occurred, but I wouldn't bet on it. Better just to place it just ahead or after the Loop order as desired.

In the following two examples of valid order placement, the WayPt order will be executed five times in the first example, and one time in the second:

```
Order9.1 = Loop,2,5
Order10.1 = WayPt,-23.692163,15.383272
```

...or...

```
Order8.1 = WayPt,-23.692163,15.383272
Order9.1 = Loop,2,5
```

Action Orders

The only current action order is the Attack order, although the ROE order indicates a general set of actions to take in specific situations, including how to attack. The Attack order specified a single unit to attack. If you need to attack multiple units, use multiple Attack orders, or set the appropriate ROE (specifying, say, to attack Submarines).

Order Modifiers

Each of the order types listed above includes two additional parameters: Duration and Target time.

Adding a duration (in minutes) will cause that order to persist for that length of time. This only works where appropriate, such as in an Attack order, i.e. you might want to instruct a Destroyer to attack for 30 minutes.

Adding a target time will cause the AI to attempt to execute that order at that time, where appropriate. In some cases, if it becomes impossible to meet that time objective, the AI may (or may not) bypass the order altogether, and jump ahead in the order list to execute subsequent orders (which may have target time directives of their own).

Order Sets

As stated above, orders come in sets, one set per group. In fact, it is probably the various sets of orders within each scenario – more then the individual units themselves – that really make the game what it is. A bad set of orders can make for an unplayable game, or worse yet, a boring game. A good set of orders can create a challenging and exciting game.

What makes a good set of orders? There are objective and subjective aspects to that, of course. What kind of game do you like to play? Fast and furious, or slow and methodical?

Some objective criteria include the fact that you must specify the basic states that any given group will operate in, including it's speed, formation and the like. The game will not function without these orders being given. To insure that each group functions properly, there are default states for most of the state orders so you can, if you so desire, live with whatever defaults there are. These are usually a speed of two thirds ahead and a basic set of ROE orders.

But the heart of a great strategic game is, of course, great strategy. And strategy in our game is largely defined by the interaction of the various units on the map as directed by the order sets given to both Allied and Axis units. If you use formation or waypoint orders to spread your units out too far from each other, they may never find each other. If you get them too close, you may just end up with a big shooting match where the biggest and most ships win. If you issue orders that are too repetitive they become predictable and boring.

Objectives

Objectives are the mission goals that the player should complete in order to advance to the next mission and to win medals. Here is an example of the objectives from an upcoming mission in the Subsim.com **Second Kampaign**:

```
[OBJECTIVE1]
Objective = REACH
Longitude = -31.316421
Latitude = 50.455429
Radius = 10.0
```

```
Range = 10000
Primary = TRUE
Score = 40

[OBJECTIVE2]
Objective = TONNAGE
Tonnage = 25000
Type = CARGO
Score = 150

[OBJECTIVE3]
Objective = RETURN_TO_BASE
Home = Wilhelmshaven
Time = 1440
Primary = TRUE
Hidden = TRUE
Score = 40
Trigger = TIME
DateTrigger = 19400325
TimeTrigger = 1100

[OBJECTIVE4]
Objective = SURVIVE
Unit = 5
Hidden = TRUE
Score = 80
```

The REACH objective specifies a point the player needs to sail to. We assign the player a primary objective of reaching his assigned (in the briefing and the first radio message) patrol grid. Therefore, as long as he goes where HQ tells him (and historically, U-boat commanders ALWAYS followed orders) he can complete his primary objective and advance to the next mission in a campaign. If he completes the other objectives, he gets the “score” added to his resume and can win medals faster.

Objective SURVIVE is usually assigned to neutrals to compel the player to identify ships before pulling the trigger.

Be careful not to set the tonnage too high. Dud torpedoes and Type II boats will mean 5000 tons is an achievable goal. Later in the war with a Type IX, 15,000 to 30,000 is reasonable.

Two low key but fun objectives are:

```
[OBJECTIVEn]
Objective = RADIO_WEATHER
Count = x

[OBJECTIVEn]
Objective = RADIO_CONTACT
Count = x
```

where n is the objective number and x is the number of reports required. Note that you will need a minimum of 4 hours between weather reports to get credit and a minimum of 1 hour between contact reports. The contact objective is great for using in conjunction with a radio message to the player from BdU instructing him to shadow a convoy but not attack. Therefore, the player must find the convoy and shadow for the number of hours you specify. If the Count = 3, then the player must shadow until three contact reports are made. Remember, the player will not know how many contact reports are needed; you must time the events and have the program send another radio message commanding him to engage the enemy.

I wouldn't make weather reports a primary objective, but they might be a nice way to add bonus points to a scenario if a player can follow orders. Note that no transmissions will be made while the U-boat is submerged, but they may be queued for later transmission when the U-boat surfaces.

HIDDEN OBJECTIVES

There is now a way to make objectives hidden for a particular scenario. A hidden objective will not show up on the help screen or the results screen, nor will they count against the player if they are never revealed.

Hidden = TRUE

Add this flag to the objective block and the objective will be hidden.

But how, you ask, can I reveal my hidden objectives? A hidden objective must have a trigger condition which causes it to appear. There are three types of triggers (so far) each with its own syntax:

Time trigger:

Trigger = TIME
DateTrigger = 19420613
TimeTrigger = 1400

The above lines, when added to an objective block, cause the hidden objective to be revealed at the specified date and time.

Contact trigger:

Trigger = CONTACT
TriggerContact = 3

The above lines, when added to an objective block, cause the hidden objective to be revealed when the player's group makes contact with the group with the specified ID.

Note that TriggerContact is a major group ID and cannot be used to specify a subgroup.

Reach trigger:

Trigger = REACH
TriggerLongitude = 5.1
TriggerLatitude = 56.5
TriggerRange = 1000

The above lines, when added to an objective block, cause the hidden objective to be revealed when any unit in the player's group reaches the specified location. The range specifies the minimum distance to that point which must be reached before the trigger occurs. This is in meters.

Note that a hidden objective that is a primary objective only counts if it is revealed. If it remains hidden then the player does not have to achieve it, even if it is primary.

So what are hidden objectives for? Well, the main thing is probably to allow an objective to be dynamically revealed when an unsuspected enemy group is detected (Force H, anyone?)

REACH OBJECTIVE RANGE

This specifies the distance within which the reach objective may be achieved.

Range = 1000

By default, range is 200 meters, which is the old hard-coded range value. A large range will save the player the hassle of hitting the bulls-eye. 1000 ~ 5000 is good.

Objectives which reference specific units/groups

There are some restrictions which concern specific mission objectives. Any objective which references a unit ID explicitly (DESTROY, SURVIVE and REACH objectives may do this) must refer to a unit that is defined in the scenario file, but NOT in a template file. This will probably break the sim if not done correctly.

Editing a Single Mission SDF File

A single mission SDF file is a smaller version of a regular SDF. It only specifies a single formation of American or British warships, or a single convoy formation. At game time the shell provides the remaining parameters needed to create a full SDF based on the parameters that the user selects at that time for his or her single mission.

For example, if the user chooses to play in the North Atlantic against a small British warship formation, without a destroyer escort, at the beginning of the war -- the single mission template file named "BrWarshipSmlNoEarly.sdf" will provide the Allied warships needed, in a proper formation. The shell will then add additional parameters including the scenario block, a submarine group and a submarine unit. See the appendix for an example of a single mission SDF file.

By the way, you can create a single mission template from a larger existing template, mainly be removing the parameter block and the Axis submarine. But since the single mission templates have already been created you may not need to. It is more likely that you will want to edit an existing template. Or, if in another full SDF file you create an Allied formation that you particularly like, you could cut and paste if over the existing Allied formation represented in one of the existing single mission SDFs.

Note that the single mission SDFs consist of a single top level squadron group, [GROUP2], and one or more division groups ,[GROUP2.1], etc. well as the units needed in those groups.

Adding Parameter Blocks

All single mission templates are located in
SH2 / Shell / Scenarios / Random.

```
[SCENARIO]
Date = 19410928 ; Use an appropriate date
Time = 1600
Duration = 1500
Type = TYPE A
Major = 10
Minor = 5
Origin = -23.744507,15.349678 ; Same as [UNIT2] location
InitialZoom = 9
Weather = CALM ; Calm probably works best for testing

[GROUP1]
Commander = HUMAN
Type = SHIPSQUADRON
UnitType = Submarine
Side = 2
Name = German Squad
```

```
Nationality = German

[UNIT1]
Group = 1
Class = NSSUBoat9C
Name = U-176
Control = HUMAN
Flag = TRUE
Guide = TRUE
Location = -23.744507,15.379678 ; Set close to other units
Orientation = 311
Captain = 80
Crew = 80
```

Appendix B - Example Files

Full SDF File – example only, current stock missions may have more up to date parameter info

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; File:   Scenario1B.SDF (with some blocks removed)
;; Name:   Western Approaches
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


[SCENARIO]
Date = 19390915
Time = 1200
Duration = 4320
Type = SINGLE
Major = 10
Minor = 5
Origin = -8.462501,50.253128
InitialZoom = 4
Weather = CALM

[SIDE1]
Alignment = ALLIES
Name = Allies
Doctrine = AMERICAN

[SIDE2]
Alignment = AXIS
Name = Axis
Doctrine = German

[SIDE3]
Alignment = NEUTRAL
Name = Neutral

[GROUP1]
Commander = HUMAN
Type = SHIPSQUADRON
UnitType = Submarine
Side = 2
Name = German U-Boat
Nationality = German

[GROUP2]
Type = SHIPDIVISION
UnitType = Merchant
Side = 1
Name = Merchant SLS 65
Nationality = American
Order1.1 = Speed,Telegraph,TWO_THIRDS,-1,-1
Order1.2 = WayPt,-6.008336,51.700001,-1,-1
Order2.1 = WayPt,-5.279169,54.075001,-1,-1

[GROUP3]
Type = SHIPDIVISION
UnitType = Merchant
Side = 1
Name = Merchant SLS 23
Nationality = American
Order1.1 = Speed,Telegraph,TWO_THIRDS,-1,-1
Order1.2 = WayPt,-7.841669,51.408337,-1,-1
Order2.1 = WayPt,-5.612502,51.387501,-1,-1
Order3.1 = WayPt,-5.195836,54.387501,-1,-1
```

```
[GROUP4]
Type = SHIPDIVISION
UnitType = Destroyer
Side = 1
Name = DesRon 12
Nationality = American
Order1.1 = WayPt,-8.725001,50.722919,-1,-1
Order2.1 = WayPt,-7.370835,51.014587,-1,-1
Order3.1 = WayPt,-7.704168,48.889587,-1,-1

[GROUP5]
Type = AIRWING
UnitType = DiveBomber
Side = 1
Name = Sunderland I
Nationality = Britain

[UNIT1]
Group = 1
TypeAbrev = SS
Class = NSSUBoat7B
Name = U-48
Control = HUMAN
Flag = TRUE
Guide = TRUE
Location = -8.345835, 49.677086
Orientation = 208
Depth = 8
Captain = 80
Crew = 80
IconName = SUBMARINE

[UNIT2]
Group = 2
TypeAbrev = FC
Class = MFCCargo
Name = Deidre
Flag = TRUE
Guide = TRUE
Location = -8.401043, 49.645836
Orientation = 80
Captain = 80
Crew = 80
IconName = MSHIP

[UNIT3]
Group = 3
TypeAbrev = FC
Class = MFCCargo
Name = Tama Aretha
Flag = TRUE
Guide = TRUE
Location = -8.145835, 49.482292
Orientation = 60
Captain = 80
Crew = 80
IconName = MSHIP

[UNIT4]
Group = 4
TypeAbrev = DD
Class = NDDBagley
Name = DD386 Bagley
Flag = TRUE
Guide = TRUE
Location = -8.245835, 49.452087
Orientation = 344
Captain = 80
Crew = 80
IconName = NSHIP
```

```
[UNIT5]
Group = 4
TypeAbrev = DD
Class = NDDBagley
Name = DD387 Blue
Location = -8.237501, 49.435421
Orientation = 344
Captain = 80
Crew = 80
IconName = NSHIP

[UNIT6]
Group = 5
TypeAbrev = HB
Class = AScCatalina
Name = Sophie
Location = -73.580002, 31.230000
Orientation = 180
Altitude = 200
Throttle = 0.2
Captain = 90
Crew = 90
IconName = NONE


[BASE1]
Alignment = AXIS
Location = -6.837501,49.190628
AirRadius = 100000.000000
AirProbability = 0.800000

[RADIO1]
Trigger = TIME
Date = 19390915
Time = 1205.00
File = S1B-1
Title = New Orders

[OBJECTIVE1]
Objective = RETURN_TO_BASE
Home = Wilhelmshaven
DateStart = 19390915
TimeStart = 1200
DateEnd = 19390918
TimeEnd = 1200
Primary = TRUE

[OBJECTIVE2]
Objective = TONNAGE
Tonnage = 10000
Type = CARGO
Primary = TRUE
```

Single Mission SDF File

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; File:  BrWarshipSmlNoEarly.SDF
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

[TEMPLATE]
Origin = 3.256613, 2.674010

[GROUP2]
Type = SHIPSQUADRON
UnitType = Cruiser
Side = 1
Name = Group
Nationality = Britain
Order1.1 = Formation,Form_As_Are,2,2
Order1.2 = WayPt,3.256613,7.2000
WayptDisplay = DO_NOT_SHOW

[GROUP2.1]
Type = SHIPDIVISION
UnitType = Cruiser
Name = Group
WayptDisplay = DO_NOT_SHOW

[UNIT2]
Group = 2
TypeAbrev = CA
Class = NCALondon
Name = Devonshire
Flag = TRUE
Guide = TRUE
Location = 3.256613, 2.674010
Orientation = 0
Captain = 80
Crew = 80
IconName = NSHIP
Display = SHOW_AS_UNIT

[UNIT3]
Group = 2.1
TypeAbrev = CL
Class = NCLLeander
Name = Achilles
Flag = TRUE
Guide = TRUE
Location = 3.256613, 2.658125
Orientation = 0
Captain = 80
Crew = 80
IconName = NSHIP
Display = SHOW_AS_UNIT

[UNIT4]
Group = 2.1
TypeAbrev = CL
Class = NCLArethusa
Name = Galatea
Location = 3.256613, 2.644779
Orientation = 0
Captain = 80
Crew = 80
IconName = NSHIP
Display = SHOW_AS_UNIT
```

Further Enhancements

This section contains info about the enhanced functionality of certain script features.

The SH2 campaign scenario preprocessing now has some new capabilities, designed with an eye towards making scenarios more dynamic:

1. Mutual-exclude selection of different group options.

This option allows you to select among up to three alternative groups, which will appear on a mutually-exclusive basis. This is set up through the use of a MUTEX parameter block. Thus:

```
[MUTEX1]
Group1 = 2
Group2 = 3
Group3 = 4
```

where each group entry refers to a major group number (i.e. [GROUP2], [GROUP3], or [GROUP4].) One of those three will be chosen for inclusion in the scenario and the other two (and their associated units) will be omitted.

Naturally, multiple MUTEX blocks are permitted.

2. Probability of group inclusion.

Alternatively, you can cause a group to be chosen on a random basis simply by using the Probability parameter:

```
[GROUP6]
Type = SHIPDIVISION
UnitType = Merchant
Probability = 50
```

The above group has a 50% chance of being included in the scenario. If the die roll fails, then this group and all subgroups and units associated with it will be omitted.

3. Use of group "template" files.

This option allows the scenario designer to load all subgroups and units for a particular group from a "template" scenario file, which is simply an abbreviated SDF file that has only the basic group organization and all of the group's units defined. It is possible to specify up to three alternative template files and the preprocessor will choose from one of the three on a random basis. Thus:

```
[GROUP6]
```

```
Type = SHIPDIVISION
UnitType = Merchant
File1 = BrConvoyMedNoEarly
File2 = BrWarshipSmlYesEarly
Location = -1.19,55.24666
Orientation = 135
```

In the above example, two files are specified, of which one will be selected at random.

Note the Location and Orientation parameters. These MUST be included where template files are specified. This is because the template file doesn't include valid location information, only location relative to the group. The preprocessor applies the location and orientation specified here to each unit in the template when it inserts it into the output scenario so that everything will appear in the correct starting location.

4. Mixing and matching randomization features.

The MUTEX and Probability features will work as well for a group that is included as a template, or which is defined wholly within the SDF file, so it should be easy to apply these features to existing scenarios. You can specify a single group in multiple MUTEX blocks and mix MUTEX and Probability features with the same group, but the results may not be predictable (which could be a good thing, I suppose...)

You can also, within a single scenario file, use groups which are loaded as templates and groups which are defined wholly within that scenario file. You might even be able to have units in the scenario file that belong to a group which is loaded from a template, though I have not tested this.

5. Template files

The current version of the preprocessor uses the same template file format as the scenario generator. This was chosen for expediency's sake, more than anything else since we already have a ready-made supply of these template files.

Template files for the campaign preprocessor are located in SH2\Shell\CampaignFiles\Templates. A template file looks just like an abbreviated SDF file which has only GROUP and UNIT blocks, plus a special block designed for templates:

```
[TEMPLATE]
Origin = 3.222475,2.657433
```

Origin specifies the location where the group was initially laid down. The location to use here is probably the longitude/latitude of the flagship for the group. Currently, if you want to make a new template file, you must include this parameter block. The Origin is used by the preprocessor when positioning units from the template. If it is not specified, units will not be positioned correctly.