Dune II Scenario Format Specification

Document Version 0.4.0

18. July 2011

Content

1. Introduction....3
   1.1. Purpose of this document....3
   1.2. Conventions in this document....3
   1.3. SCENARIO.PAK....4
   1.4. INI Files....5
2. Regions....7
   2.1. File naming....7
   2.2. Section “GROUP<LevelNumber>”....7
      Colored regions....7
      Selectable regions....7
      Text shown below the selection map....8
   2.3. Section “PIECES”....8
   2.4. Section "INFO"....8
3. Scenarios....9
   3.1. File naming....9
   3.2. Map Positions....9
   3.3. Timing....10
   3.4. Section “BASIC”....10
      Key “Version”....10
      Keys “LosePicture”, “WinPicture” and “BriefPicture”....10
      Key “TimeOut”....10
      Key “MapScale” (Version < 2)....11
      Key “CursorPos”....11
      Key “TacticalPos”....11
      Keys “WinFlags” and “LoseFlags”....11
      Key “License” (Version ≥ 2)....12
      Key “Author” (Version ≥ 2)....12
      Key “TechLevel” (Version ≥ 2)....12
   3.5. Section “FEATURES”....13
   3.6. Section “MAP”....14
      Key “Field” (Version < 2)....14
      Key “Bloom” (Version < 2)....14
      Key “Seed” (Version < 2)....14
      Key “Special” (Version < 2)....14
      Keys “SizeX”, “SizeY” and Key “000” up to Key “128” (Version ≥ 2)....15
   3.7. Section “<HouseName>”....16
      Key “Quota”....16
      Key “Credits”....17
      Key “Brain” (Version < 2)....17
      Key “MaxUnits”....17
   3.8. Section “CHOAM”....17
   3.9. Section “TEAMS”....18
   3.10. Section “UNITS”....19
   3.11. Section “STRUCTURES”....20
   3.12. Section “REINFORCEMENTS”....21

1. Introduction

1.1. Purpose of this document

Dune II is a real-time strategy game from 1992. It was the first popular RTS game and paved the way for later games in this genre like Command & Conquer or Starcraft.

With it's great popularity in the 90th the game is still known today by many gamers. This led to many Mods and Remakes. All these project have in common that they need to figure out how Dune II works. Especially the data files were reverse engineered and are now well understood. This document covers one of these data files; the scenario description which tells Dune II how the game map looks like.

For Dune II remakes it is possible to define several extensions to the format used by Dune II. This document marks such extensions in the following way:

> *Extension:*
>
> *The text describing the extension.*

1.2. Conventions in this document

This document uses the following notation for text to be substituted in format specification:

`<digit>`	One of the number 0 to 9
`<integer>`	A sequence of `<digit>`
`<HouseName>`	Atreides, Ordos, Harkonnen, Fremen, Mercenary or Sardaukar *Extension:* *For players that are randomly chosen on game start the following values are possible: Player1, Player2, Player3, Player4, Player5 or Player6.* *There shall be only 6 houses in total on the map.*
`<HouseLetter>`	A, O, H, F, M or S
`<House3Letter>`	ATR, ORD, HAR, FRE, MER, SAR
`<Language3Letter>`	ENG, FRE, GER
`<LevelNumber>`	The Dune II campaign consists of 9 Levels (1 to 9)
`<ScenarioNumber>`	For each house there are 22 different Scenarios

1.3. SCENARIO.PAK

All game data of Dune II is stored in PAK archive files. The PAK file format very simple and is well known but will not be described in this document. One PAK file contains one or more files. You may think of them as a zip file without compression. SCENARIO.PAK contains the campaigns for all three houses.

All the scenarios are specified in a file called SCENARIO.PAK with the following content:

REGIONA.INI	REGIONH.INI	REGIONO.INI	SCENA001.INI	SCENA002.INI
SCENA003.INI	SCENA004.INI	SCENA005.INI	SCENA006.INI	SCENA007.INI
SCENA008.INI	SCENA009.INI	SCENA010.INI	SCENA011.INI	SCENA012.INI
SCENA013.INI	SCENA014.INI	SCENA015.INI	SCENA016.INI	SCENA017.INI
SCENA018.INI	SCENA019.INI	SCENA020.INI	SCENA021.INI	SCENA022.INI
SCENH001.INI	SCENH002.INI	SCENH003.INI	SCENH004.INI	SCENH005.INI
SCENH006.INI	SCENH007.INI	SCENH008.INI	SCENH009.INI	SCENH010.INI
SCENH011.INI	SCENH012.INI	SCENH013.INI	SCENH014.INI	SCENH015.INI
SCENH016.INI	SCENH017.INI	SCENH018.INI	SCENH019.INI	SCENH020.INI
SCENH021.INI	SCENH022.INI	SCENO001.INI	SCENO002.INI	SCENO003.INI
SCENO004.INI	SCENO005.INI	SCENO006.INI	SCENO007.INI	SCENO008.INI
SCENO009.INI	SCENO010.INI	SCENO011.INI	SCENO012.INI	SCENO013.INI
SCENO014.INI	SCENO015.INI	SCENO016.INI	SCENO017.INI	SCENO018.INI
SCENO019.INI	SCENO020.INI	SCENO021.INI	SCENO022.INI	

There are two types of files:

- REGION<HouseLetter>.INI:

  These files describe in which order the scenarios are played. They are needed for the map selection screen between two scenarios (The screen where you can select between 3 regions which to attack next)

- SCEN<HouseLetter><digit><digit><digit>.INI:

  These files describe one scenario. Dune II has only 9 levels but on the map selection screen you choose between different scenarios to play.

Both types are so called INI files (see Chapter 1.4).

Extension:

The file may be named in a different way than SCENARIO.PAK for saving different campaigns or single maps. It may also contain different files as long as there is one file with the name SCEN?001.INI. If it is a complete campaigne the file REGION?.INI is also needed.

1.4. INI Files

INI files are plain ASCII (Codepage 437) text files with a special structure. They consist of many key-value pairs. There is one pair per line. The format of one line is as follows:

```
<key> <whitespace> = <whitespace> <value>
```

<whitespace> is either spaces (' ') or tabulators ('\t') or both.

<key> may contain any characters except '[', ']', ';', '=' or line break. A whitespace is not allowed at the beginning or end of the name.

<value> may contain any characters except '[', ']', ';' or line break

The following example tells Dune II that the starting credits are 2000:

```
Credits=2000
```

Every pair belongs to exactly one section. A section is started with a line in the following format:

```
[ <sectionname> ]
```

where <sectionname> may contain the same characters as <key>.

A section is closed with the start of the next section or with the file end. The shall be no key-value pair before the first section is started.

Basically a INI file is a tree of height 3 with the keys being the leafs, the root node being the file/document and the nodes in-between being the sections. Given the filename, the section and the key, every key-value pair can be accessed. This implies that there shall be no two keys with the same name in one section.

INI files may also contain empty lines and comments. Comments are started with ';' at the beginning of the line and end after the next line break.

> *Extension:*
>
> *An '#' can be used to start a comment like other software uses in it's INI files.*

Attention: Some INI files in SCENARIO.PAK (like SCENH002.INI) have some garbage at the end. All software handling scenarios should be able to handle this. The easiest way is to treat it as a comment (without the starting ';'). If a line is not empty and the first non-whitespace character on the line is neither '[' nor ';' it must be a key-value pair or garbage. If the line contains an '=' it must be a key-value pair, otherwise it is garbage.

Attention: Dune II does not always detect comment correctly. Especially in “UNITS” and “STRUCTURES” sections Dune II fails to recognize comments and will parse the string thereafter. The best way to force Dune II to ignore a line is starting it with “;= “.

Extension:

The handling of INI files shouldn't be that strict:

Whitespaces at the beginning of a line shall be ignored (Whitespaces before <key> shall be allowed but they are not part of the key).

Whitespace after <sectionname> and the closing bracket should be allowed. Whitespaces after <value> shall be ignored.

Comments at the end of the line shall be allowed.

Extension:

If it is necessary to use a different text encoding than DOS Codepage 437 the used encoding should be UTF-8. To distinguish between the two encodings the UTF-8 encoded file shall contain the following line at the very beginning of the document:

```
;encoding=utf-8
```

2. Regions

2.1. File naming

There is one REGION<HouseLetter>.INI file for every house. When playing the campaign you first have to select your house. Then you always play SCEN<HouseLetter>001.INI. Afterwards you get to the map selection screen. The presented map consists of 27 pieces and REGION<HouseLetter>.INI controls what pieces are changing its colors and what pieces (usually 3) are selectable by the player.

2.2. Section “GROUP<LevelNumber>”

The REGION<HouseLetter>.INI file contains 8 GROUP<LevelNumber> sections. Depending on the just played level (1 to 8) the sections is used for the map selection screen. After Level 9 there is no map selection screen and therefore no section in the file. Each GROUP<LevelNumber> Section may contain any of the following keys:

```
<House3Letter> = <ListofRegions>
REG<digit> = <RegionNumber>, <ArrowNumber>, <XPos>, <YPos>
<Language3Letter>TXT<RegionNumber> = <description>
```

Colored regions

<ListofRegions> is a comma separated list of region numbers. These regions are colored in the house color of <House3Letter>, one region in a time. Example:

```
ATR = 5, 13, 22
```

This will first color region 5 with blue, than region 13 and last region 22.

For every house there can be one key. It is important to note here that this key only determines which regions are recolored in a new color. If you want to know how the map looks like after e.g. level 5, you have to first calculate how it looked before level 5 based on GROUP1, GROUP2, GROUP3 and GROUP4. Than you can apply GROUP5 visually, e.g. blending in regions one by one.

Selectable regions

REG<digit> describes the selectable regions. There can be multiple REG<digit> entries, but the numbers should be continuous starting with 1, e.g. REG1, REG2 and REG3 (Dune II uses up to REG4). <RegionNumber> is the piece on the map that gets selected when clicking the region. <ArrowNumber> is the zero-based index in ARROW.SHP (in DUNE.PAK) which is used to load the right arrow picture. There are 8 different directions; Arrow 1 pointing to the upper left, Arrow 2 pointing upwards, etc. <XPos> and <YPos> determine the position the arrow shall be drawn.

The following example shows an upper right arrow to piece 8, drawn at (80,80):

```
REG1    =    8, 3, 80, 80
```

There shall be 3 (different) selectable regions after mission 1 to 6. After mission 7 there shall be only 2 selectable regions and after mission 8 only one. There might be more than one arrow to one region but additional arrows to the same region must be listed after all arrows to distinct regions. When an region is selected the game determines with “REG<digit>” of the first arrow pointing to the selected region what scenario INI file to load next. This is done as follows:

```
if previousLevel <= 7
   nextMission = (previousLevel-1) * 3 + 2 + <digit>
else if previousLevel == 8
   nextMission = (previousLevel-1) * 3 + 1 + <digit>
else
   // should not be reached since there are only 9 levels
   nextMission = (previousLevel-1) * 3 + <digit>
```

Text shown below the selection map

<Language3Letter>TXT<RegionNumber> encodes for which language and when the text is shown. <RegionNumber> must be one of the regions listed in one of the <ListofRegions> entries in this section. When the specified the region is faded in the text is also shown. The shown text is the value of this key. For Dune II is has to be encoded in DOS Codepage 437 (see Chapter 1.4). Example, that will show the text when region 13 is fading in, but only if the language is english:

```
ENGTXT13     =  The Atreides claimed strategic regions.
```

2.3. Section “PIECES”

This section is needed to tell Dune 2 where the different pieces of the map have to be drawn. It contains keys with the format

```
<RegionNumber> = <XPos>, <YPos>
```

In Dune II the RegionNumber goes from 1 to 27. <XPos> and <YPos> determine the position the arrow shall be drawn.

2.4. Section "INFO"

This section contains only one key:

```
TOTAL REGIONS = <NumRegions>
```

<NumRegions> is 27 in Dune II.

3. Scenarios

3.1. File naming

There is one SCEN<HouseLetter><digit><digit><digit>.INI for every scenario and every house. In total there are 22 scenarios per house. On most levels there is the choice between 3 different scenarios. In level 1 and 9 you have no choice, in level 8 only 2.

Level	Available Scenarios/Maps
1	1
2	2, 3, 4
3	5, 6, 7
4	8, 9, 10
5	11, 12, 13
6	14, 15, 16
7	17, 18, 19
8	20,21
9	22

Extension:

For Dune 2 remakes it shall be possible to load INI files directly. These scenarios/maps may have any name. If possible the map name shall follow a naming scheme which mentions the number of players and the map size, e.g.

4P – 128x64 – Name of the Map.ini

3.2. Map Positions

Positions on the map are saved in one integer value. All cells on the map are continuously numbered row by row starting with 0 at the top left map corner. These coordinates are always based on a 64x64 map, even when MapScale=1 and the map is displayed as 32x32 (see Chapter 3.4). The transformation between 2D map coordinates and the position value is done as follows:

```
position = y * 64 + x
```

The transformation in the other direction is

```
x = position mod 64
y = position div 64
```

where mod is the modulo operator and div is the integer division.

3.3. Timing

Dune II can be run with different game speeds. Depending on the game speed a different number of game ticks is performed per second:

Game speed	Slowest	Slow	Normal	Fast	Fastest
Ticks/Second	15	22.5	30	45	60

All timings in the scenario INI file are given as minutes when using the game speed “fastest”.

3.4. Section “BASIC”

This section contains the following keys.

Key “Version”

This key gives the version of the scenario/map format. The original Dune II format does not contain this key which equals Version 1. Version 2 and above extend the format. The default value for this Key is 1

Keys “LosePicture”, “WinPicture” and “BriefPicture”

The keys LosePicture, WinPicture and BriefPicture specify an WSA file that gets shown when the player loses the scenario, wins it or in the briefing before the scenario started.

Dune II uses the following values for these keys:

LosePicture (from DUNE.PAK)	**WinPicture (from DUNE.PAK)**	**BriefPicture (from MENTAT.PAK)**
LOSTBILD.WSA	WIN1.WSA	HARVEST.WSA
LOSTVEHC.WSA	WIN2.WSA	HEADQRTS.WSA
		QUAD.WSA
		LTANK.WSA
		REPAIR.WSA
		HVYFTRY.WSA
		IX.WSA
		PALACE.WSA
		SARDUKAR.WSA

Key “TimeOut”

The key TimeOut is not correctly implemented in Dune II. It is enabled with Bit3 in WinFlags/LoseFlags. Unfortunately the timeout is not copied into the right variable in Dune II. This results in a TimeOut of always 0.

> *Extension:*
>
> *If implemented the Key “TimeOut” shall determine the minutes until the game ends. The player should be informed about the timeout with a count down.*

Key “MapScale” (Version < 2)

A Dune II map is always 64x64 tiles big but only a portion of this map is displayed depending on the value of MapScale:

MapScale	Position of Top Left Tile	Position of Bottom Right Tile
0	(1,1)	(62,62)
1	(16,16)	(47,47)
2	(21,21)	(41,41)

All positions are still calculated on the 64x64 map (see Chapter 3.2). Do not specify this key if you use the Keys “SizeX” and “SizeY” in the “Map” section.

Key “CursorPos”

This position specifies the position of the selected unit or structure when the map is loaded.

Key “TacticalPos”

This position specifies the top left corner of the screen when the map is loaded. The screen is 15x10 tiles big which gives the center of the screen at

TacticalPos + 64*5 + 7 = TacticalPos + 327

Keys “WinFlags” and “LoseFlags”

The Keys “WinFlags” and “LoseFlags” determines under which conditions the game ends and who has won. They are specified using normal numbers but their meaning is better explained with their bit pattern (they both use the same bit pattern):

Bit 4: Unused; Dune II scenario files sometimes have set it, but it isn't checked	Bit 3: Check if time ≤ timeout (see Key “TimeOut” why this does not work in Dune II)	Bit 2: Check if player credits ≤ quota	Bit 1: Check if player has at least one building	Bit 0: Check if AI player has no buildings

The bit pattern of the Key “WinFlags” determine which of these conditions should be checked every 5 seconds. The first check is done 2 minutes after the game starts. If any of the specified conditions is met the game will end. “LoseFlags” now determines who has won. If any of the bits in “LoseFlags” is set and the corresponding condition holds true the player has won (and the computer has lost). If all of the specified conditions are not met the computer has won.

There is one exception to this rule (only for the “LoseFlags”): If Bit 1 and Bit 0 are both set than their conditions have to be both met for the player to win, otherwise the computer has won.

Examples:

WinFlags	LoseFlags	Meaning
6	4	Bit 2 is set in both, thus the player wins if the quota is reached. “WinFlags” has additional set Bit 1 but is not set in “Lose Flags”. This means the player has lost when his base is destroyed.
7	5	The same as above but additionally Bit 0 is set in both. If the enemy base is destroyed the game ends and the player has won. Even if the player base is destroyed at the same time as the AI base (or both bases in the first 2 minutes of the game), the player has won.
1	0	The only way to end the game is the destruction of the enemy base. Since Bit 0 is not set in “LoseFlags” the player has lost (and the AI won).
1	4	This construction is particular interesting. It forces a particular order in which the conditions have to be met. First you should recall that the “LoseFlags” are not checked before one of the “WinFlags” conditions are met. The only specified condition in the “WinFlags” is the destruction of the enemy base like above. But this time Bit 2 in “LoseFlags” is set. If the enemy base is destroyed and the player has more credits than the specified quota he has won; if he has less the AI wins (even if it's base is destroyed).

Key “License” (Version ≥ 2)

The License information can be used to specify under which license this map can be distributed and modified. If you want everybody to be able to modify your map and redistribute the derived work you might want to choose the Creative Commons Attribution ShareAlike License:

```
License = CC-BY-SA
```

See http://creativecommons.org/licenses/ for other Create Commons Licenses.

This key can contain any other license as well.

Key “Author” (Version ≥ 2)

This key contains a list of authors that created the map. It is possible to include an email address after the name, e.g.

```
Author = John Doe <john.doe@mail.com>, Average Joe
```

Key “TechLevel” (Version ≥ 2)

This key specifies which buildings and units are available in this mission. Everything that is available in one tech level is also available in higher tech levels. The following table lists structures that are added at a certain tech level:

TechLevel	Additional available structures	Additional available units
1	Concrete, Windtrap, Refinery	
2	Radar Outpost, Barracks (A, O, F, M), Light Factory (A, O, F, M), Spice Silo, WOR (H, F, S)	Soldier (A, O, F, M), Trooper (H, F, S), Trike (A, F, M), Raider (O)
3	Light Factory (H, S)	Trike (H, S), Quad
4	Concrete4, Wall, Heavy Factory	Tank, Harvester, MCV
5	Gun Turret, WOR (O, M), Hightech Factory, Repair Yard	Trooper (O, M), Launcher (A, H, F, S), Carryall
6	Starport, Rocket Turret	Siege Tank (A, H, F, S)
7	House IX	Sonic Tank (A, F), Deviator (O, M), Devastator (H, S), Siege Tank (O, M), Ornithopter (A, O, F, M)
8	Palace	Fremen (A, F), Saboteur (O, M)

If this key is not present the tech level depends on the mission played. It is increased with each mission up to a maximum of 8. If no campaign is played it defaults to 8.

3.5. Section “FEATURES”

This section is optional. It may list any required map features that go beyond what is specified in this document. Any such feature must be assigned a unique name that is given a boolean value of “enabled” in the FEATURES section. While loading a map any program implementing this specification must first check for the FEATURES section. If this section exist it must further check all keys given and reject further loading of the map, if it finds any key, which it does not understand or does not support.

Take for example the support of an additional terrain type “lava” (which is not specified in this document). A game that wants to implement such an terrain type would like to use an additional character “$” in the map data (see Chapter 3.6) to store locations of lava. In order to not confuse other programs reading the map it is necessary to indicate the use of a not standard feature in the feature section. A unique name for the new feature has to be chosen, e.g. “Terrain Type Lava” and be included in the FEATURES section:

```
[FEATURES]
Terrain Type Lava = enabled

[MAP]

000=------------$$$$$----------%%%%%
001=-%%%%%%%%----------------%%%%%--

...
```

This allows any program to check if it supports lava terrain. If not it can reject further loading of the map.

3.6. Section “MAP”

Key “Field” (Version < 2)

This key specifies positions on the map where additional spice fields should be created. Its value is a comma separated list of positions. The field is about 7x7 fields big but no exact information about the field creation is known.

Example:

```
Field = 2000, 3048
```

Do not specify this key if you use the Keys “SizeX” and “SizeY”. Instead you shall specify the spice location directly on the map.

Key “Bloom” (Version < 2)

This key specifies positions on the map where spice blooms should be placed. Its value is a comma separated list of positions. The spice bloom will explode if a unit drives over it or it is shoot. When the spice bloom explodes it will create a field of about 7x7 spice fields.

Example:

```
Bloom = 2000, 3048
```

Do not specify this key if you use the Keys “SizeX” and “SizeY”. Instead you shall specify the bloom location directly on the map.

Key “Seed” (Version < 2)

Dune II creates its maps based on a random map generator. This random map generator uses a pseudo random number generator. The pseudo RNG is seeded with the value specified by the key “Seed”, which is a single number.

Example:

```
Seed = 1234
```

Do not specify this key if you use the Keys “SizeX” and “SizeY”. Instead you shall specify the map directly.

Key “Special” (Version < 2)

This key specifies positions on the map where special blooms should be placed. Its value is a comma separated list of positions. The special bloom looks very similar to a spice bloom. It will open up if a unit drives over it but it does not open up if it is fired upon. When triggered something random happens: You get either some money or a unit is spawned. Unfortunately Dune II has a bug resulting in very strange behaviour. The expected behaviour is that one of the following 4 things happen:

- The player gets an randomly choosen amount of credits between 150 and 400.

- The player gets a Trike for free. It spawns beside the special bloom.
- One of the AI players on the map (one that has at least one unit) gets a Trike for free. It spawns beside the special bloom.
- One of the AI players on the map (one that has at least one unit) gets an Infantry unit (3 Soldiers) for free. The spawn beside the special bloom.

The Dune II programmers mixed up the parameters for the unit creation function and switched the house ID with the unit ID. This causes very weird results because the created unit depends now on the AI players on the map. Trike has ID 13 but there is no house with ID 13 (Harkonnen = 0, Atreides = 1, Ordos = 2, ...). When now a Trike shall be spawn, another unit is spawn depending on your or your enemies house. This unit belongs to the invalid house 13 and will be quite useless because they will not interact with the player or the AI. It will be either an ornithopter, a carryall or nothing since the game will not create a ground unit with an invalid house. More interesting is the case where a infantry should be spawned. The ID of the infantry is 2 which is the same as house Ordos. Depending on the houses of the AI there will be an Ordos Ornithopter, an Ordos Carryall or Ordos Infantry spawned. And these units “work” as expected. If there are Fremen, Sardaukar or Mercenary on the map it might be 3 Ordos Troopers, 1 Ordos Soldier or 1 Ordos Trooper spawned.

If the player already has explored a Special Bloom which produced one of these useless units from house 13 there might be 1 Ordos Trike spawned when exploring further Special Blooms.

Do not specify this key if you use the Keys “SizeX” and “SizeY”. Instead you shall specify the location of the special bloom directly on the map.

Keys “SizeX”, “SizeY” and Key “000” up to Key “128” (Version ≥ 2)

This extension to the Dune II Scenario Format allows custom maps. Custom Maps should not contain the keys “Seed”, “Field”, “Bloom” or “Special”. Instead they contain the keys “SizeX” and “SizeY” specifying the map size. Valid values are “32”, “64” and “128”. “SizeX” and “SizeY” do not have to be necessary the same thus non quadratic maps are possible. The actual map is give in the keys “000” up to Key “<SizeY> - 1” with “000” specifying the top row of the map. The value of these keys have to be <SizeX> long, filled with the following characters:

Character		**Meaning**
-	(minus)	Normal sand
^	(caret/hat)	Sand dunes
~	(tilde)	Spice
+	(plus)	Thick spice
%	(percent)	Rock
@	(at)	Mountain (Must have rock or mountain to the left, right, top and bottom)
O	(uppercase o)	Spice Bloom (You should use this instead of Key “Bloom”)
Q	(uppercase q)	Special Bloom (You should use this instead of Key “Special”)

Example:

```
[MAP]
SizeX=32
SizeY=64

000=----------------------------%%%%%
001=-%%%%%%%%----------------%%%%%--
002=--%%%%%%%%%%----O-----%%%%%%----
003=-%%%%%%%%%%%%--------%%%%%%-----
004=----%%%%%%%%%%%%---%%%%%%-------
005=-Q--%%%%%%%%%%%%%%%%%%%%--^^^--
006=-----%%%%%%%%%%%%%%%%%%--^^^^--
007=-----%%%%%%%%%%%%%%%%%%---^^^^-
008=-------%%-----%%%%--------~~~--
009=-------%%----------------~++~~~--
010=-------%%%-----~~~~-~~~~~~+++~---
011=--O-----%%%-----~~~~~~++++++~~--
012=-----------------~~~~~~~~+++~~---

...

061=--------%%--------%%%-----------
062=-------%%%%-----%%%%%-----------
063=----%%%%@@%%%%%%%%%------------
```

3.7. Section “<HouseName>”

For every house that has units or structures on the map there shall be one house section with the name of the house. There are two exceptions to this rule:

- Players that only have Sandworms on the map (they usually belong to Fremen) can omit their house section
- Players that appear in the middle of a game (e.g. Sardauker reinforcements or Fremen Troopers called by the Atreides palace.

> *Extension*
>
> *If the house is given by “Player1”, “Player2”, … or “Player6” it is chosen randomly at the beginning of the game.*

Key “Quota”

This key specifies the amount of credits this player needs to harvest and refine to win. It is only used if Bit 2 in “WinFlags” or “LoseFlags” is set. Setting “Quota” for AI players is meaningless.

Starting credits and refined credits are separated by Dune II. Both together give the current amount of credits the player owns. When spending money first

the starting credits are reduced until they are 0. Not till then the refined credits are spend. To reach the winning condition the player has to own the specified amount as refined credits.

Key “Credits”

The starting credits are specified by this key. Should be below 32768 to be displayed properly.

Key “Brain” (Version < 2)

This key can only have the value “Human” or “CPU”. Only one player can be a human player. All others have to be AI Players and therefore of type “CPU”. This key may be ignored by Dune II remakes.

Key “MaxUnits”

This key specifies up to which unit count the player is allowed to build units. It is only relevant for building units; not for ordering them at the Starport. Nevertheless ordering units at the Starport will increase the unit count and may later disallow the player to build units.

This key may be ignored by Dune II remakes.

3.8. Section “CHOAM”

This section describes what vehicles are available at the Starport and in what amount. For every available vehicle there is one key-value pair in this section with the key being the name of the vehicle (see Chapter 3.10) and the value giving the amount of vehicles available for order at the beginning of the game. The special value -1 indicates that the vehicle will be available during the game but the amount at the beginning is 0.

Example:

```
[CHOAM]
Launcher=2
Trike=3
Quad=-1
```

This makes 2 Launchers and 3 Trikes available at the beginning of the game. Quads will be later available.

Every 30 seconds a unit is chosen randomly and its amount is increased by 1 (up to an amount of 10). If the chosen unit is not listed in the “CHOAM” section and thus not available nothing happens. Furthermore the prices for all available vehicles are changed every 60 seconds. The price is calculated with the following formula:

```
price = min( rand(4,15) * UnitPrice / 10 , 999 )
```

where `rand(a,b)` gives a random number between `a` and `b` and `UnitPrice` is the normal price the unit costs when building it in a factory.

3.9. Section “TEAMS”

This section is used by the AI. It defines several teams of units controlled by the AI. All units produced by the AI will be added to one of the teams. A team is defined by a key-value pair with the following format:

```
<TeamID>=<HouseName>,<Behaviour>,<Type>,<MinUnits>,<MaxUnits>
```

<TeamID> is the ID of the team. It is a number between 0 and 15. Normally team ID starts with 1 and goes up continously.

<HouseName> is the house of the team.

<Behaviour> can have one of the following values:

Normal	Normal means a team can retreat and regroup
Kamikaze	Units in Kamikaze teams will not only attack until destroyed, but also ignore enemy fire and focus on taking out priority targets (i.e. structures). All other team types return fire and switch their targets to those units (but, as it seems, not turrets) that attacked them.
Staging	This is the only defensive team type. The AI will amass units inside the base and position them so that the buildings and approaches to the base are guarded. The units in a Staging team will remain where they are for most of the time, and only attack hostile units in range.
Flee	The Flee team type is designed as a quick strike force. Once produced, units immediately leave for the enemy base, and the entire team is being assembled on the move. Teams of this type can easily overwhelm the player because they generate an almost constant flow of units (killed units get replaced immediately).

<Type> determines the type of units to produce for this team:

Foot	Infantry and Trooper units
Wheel / Wheeled	Raider and Quad units.
Track / Tracked	All tracked units like Tanks, Launchers, Siege Tanks, Sonic Tanks, Devastators and Deviators.
Winged	'Thopters
Slither	Sandworms
Harvester	Not known yet.

Attention: Dune II has a bug with Wheel and Track teams. The program checks for “Wheeled” or “Tracked” but the scenario files contain 'Wheel' and 'Track'.

<MinUnits> and <MaxUnits> determine the minimum and maximum size of the team.

> *Extension:*
>
> *Teams are only a hint for the AI in Dune II remakes.*

3.10. Section “UNITS”

This section contains all the units that are placed on the map when the game is started. For every unit there is a key with the following format:

```
ID<ID>=<HouseName>,<Unit>,<Health>,<Position>,<Angle>,<Mode>
```

<ID> is a unique 3 digit number used internally by Dune II.

<HouseName> is the house of the owner of the unit

<Unit> is the type of unit to place. The following are valid names:

Carryall	'Thopter	Infantry	Troopers	Soldier
Trooper	Saboteur	Launcher	Deviator	Tank
Siege Tank	Devastator	Sonic Tank	Trike	Raider Trike
Quad	Harvester	MCV	Sandworm	Frigate
Special*				

* Special is only available since Version 2. It is used to specify a Devastator (Harkonnen), Sonic Tank (Atreides) or Deviator (Ordos) depending on the owners house. Fremen, Sardaukar and Mercenary have Sonic Tanks and Devastators: The first special unit for these houses is a Sonic Tank, the next special unit a Devastator, the third special unit a Sonic Tank, ... This alternating is per house and independent of the special units of other houses.

<Health> gives the initial health of the unit in percent of the maximum health of this unit. A value of 256 equals 100% health; 128 equals 50% health.

<Position> is the position on the map.

<Angle> gives the initial heading of the unit. The value gives the angle in which the unit is orientated. The following table shows the possible values:

0	North	32	Northeast	64	East	92	Southeast
128	South	160	Southwest	192	West	224	Northwest

<Mode> gives the attack mode of the unit. The following modes are possible:

Guard	The unit will attack enemy units but will not move or follow enemy units.
Ambush	Ambush means a unit will remain in position until sighted by the enemy, and then proceed to attack any enemy units it might find on the map.
Hunt	Hunt makes a unit start from its position towards enemy units, even if the player has not sighted the AI (normally the AI will not attack until there has been a contact between the player's and the AI's units). Also works for human units, they'll go towards any enemy units on the map just as the mission starts.
Area Guard	Area Guard is the most common command for pre-placed AI units. They will scan for targets in a relatively large radius, and return to their original position after their target was either destroyed or left the immediate area.
Harvest	If used for a harvester it will immediately start to harvest spice.
Sabotage	Sabotage is the default command for Saboteurs, who will head towards the highest priority enemy structure on the map (that is already explored) and attempt to destroy it. Other infantry with this order will head towards an enemy structure and enter it, either dealing damage or capturing it.

Example:

```
[UNITS]
ID034=Atreides,Sonic Tank,256,1837,32,Guard
ID024=Ordos,Siege Tank,192,2649,96,Area Guard
```

This places an Atreides Sonic Tank at position 1837 with 100% health. The sonic tank is headed towards northeast and put in guard mode. The second unit is an Ordos Siege Tank at position 2649 with 75% health and headed towards the southeast. It is in area guard mode.

3.11. Section “STRUCTURES”

The section “STRUCTURES” contains all the structures that are placed on the map at the beginning of the game. For every structure there is one key-value pair. For Walls and Concrete the format is as follows:

```
GEN<Position>=<HouseName>,<StructureType>
```

<Position> is the position on the map.

<HouseName> is the name of the house this wall or concrete belongs to. This is important for placing buildings near this wall or concrete

<StructureType> is either 'Wall', 'Concrete' or 'Concrete4'. If using other structures the other format for structures should be used. The game might behave strange if placing regular structures with 'GEN'

The other format for structures can be used for any type of structure, but should be used for anything beside Wall and Concrete.

```
ID<ID>=<HouseName>,<Structure>,<Health>,<Position>
```

<ID> is a unique 3 digit number used internally by Dune II.

<Structure> is the name of the structure to place. The following table lists all possible values:

Concrete	Concrete4	Palace	Light Fctry	Heavy Fctry
Hi-Tech	IX	WOR	Const Yard	Windtrap
Barracks	Starport	Refinery	Repair	Wall
Turret	R-Turret	Spice Silo	Outpost	

<Health> gives the initial health of the structure in percent of the maximum health of this structure. A value of 256 equals 100% health; 128 equals 50% health.

<Position> is the position on the map.

Example:

```
[STRUCTURES]
GEN1629=Atreides,Concrete
GEN1693=Atreides,Concrete
ID000=Atreides,Const Yard,256,1630
ID001=Ordos,Const Yard,128,2650
```

This will create an Atreides Construction Yard at position 1630 with 100% health. To the left of the Construction Yard there are to tiles of Concrete. The Ordos Construction Yard is at position 2650 with 50% health.

3.12. Section “REINFORCEMENTS”

Reinforcements are units dropped near the home base or the enemy base or somewhere else. Dune II uses a table with 16 entries to store them thus there can only be 16 reinforcements specified (If Version ≥ 2 an unlimited number of reinforcements must be supported). They have the following format:

```
<TableIndex> = <HouseName>,<UnitName>,<DropLocation>,<Time>
```

<TableIndex> is the index in the reinforcement table used by Dune II. In the original Dune II missions the developers used only index 1 to 15. Therefore the indices should be continuously and started with 1. If a 16th entry is needed it must have index 0 to not overflow the table in Dune II.

<HouseName> is the house the reinforcement belongs to.

<UnitName> is the name of the unit to drop (See Chapter 3.10).

<DropLocation> specifies where a unit should be dropped. There are 8 different values possible:

<DropLocation>	**Meaning**
North East South West	The unit will suddenly appear at the top, right, bottom or left edge of the map. At the top and left edge the unit is not placed directly at the edge but with a distance of 2 to the edge; at the bottom and right edge with a distance of 1. This DropLocation does only work reliably on maps with a MapScale of 0 (see Chapter 3.4). Otherwise the unit is dropped outside the map.
Air	This will drop a unit with a carryall at a randomly chosen position on the map.

Visible	This will drop a unit with a carryall. The unit is dropped in the map rectangle currently visible. At the start of the mission this is:
	`Drop.x = rand(TacticalPos.x, TacticalPos.x+14)`
	`Drop.y = rand(TacticalPos.y, TacticalPos.y+9)`
	
	Extension:
	
	Make this drop the unit at the middle of the map. The unit is placed randomly in a circle around the middle tile of the map (see Enemybase/Homebase).

Enemybase Homebase	This will drop a unit with a carryall. “Homebase” drops the unit near the base of the owner of the unit. “Enemybase” drops it near the other player. Thus the drop location is relative to the owner of the unit. If there is no base (= no structure) a unit of the player is located. If even this fails the unit is dropped randomly on the map. The unit is dropped in a circle with radius 7 around the located structure or unit (the position base): r = rand(0,7) angle = rand(0°,360°) Drop.x = base.x + r * sin(angle) Drop.y = base.y - r * cos(angle)

If the chosen drop position is occupied the position finding algorithm is simply repeated until a free position is found. If there is no free position the game will hang.

<Time> is the time in minutes when the unit should be dropped. It may be postfixed with a '+' meaning it should repeat infinitely. Unfortunately Dune II has a bug ignoring the '+'.

If multiple units of the same house should be brought at the same time there will be only one carryall delivering them all.

Example:

```
1=Atreides,Trike,Homebase,6
2=Atreides,Infantry,Homebase,6
3=Atreides,Infantry,Visible,11
4=Atreides,Infantry,Visible,11
5=Harkonnen,Trooper,Enemybase,5+
6=Harkonnen,Trooper,Enemybase,5+
```

This will drop a Trike and an Infantry unit at the Atreides base after 6 minutes. After 11 minutes two Infantry units are dropped at the Atreides base. Additionally every 5 minutes two Trooper units are dropped. The are also dropped in the Atreides base because they are Harkonnen and should be dropped in the enemy base (= the Atreides base).