Addon Creation Guide

Version 2009.11.21

Graviteam ®

TABLE OF CONTENT

1 GENERAL INFORMATION ABOUT ADDON CREATION

To create addons the patch No.7AE or the newer one needs to be installed over the original game. To unpack archive files of the game and create own addons specific commands are meant. To call a command the next format is used:

starter.exe <command name>, <command parameters>

A command name and parameters are divided by a sign “,”. To launch a command either a file shell of a type Far (http://www.farmanager.com/download.php) needs to be used or OS command file (with extension bat or cmd) needs to be created in the root folder of the game.

Results of transformations are recorded into log-files in the folder out in the root directory of the game (with a name complying with the command name).

By launching a command without parameters a dialog box appears that allows to select file in order to pack or unpack it. The command files for quick launch of commands are located in the folder "docs\modwork\" in directories:

- asets – for work with resources;
- cfgtext – for work with texts and settings;
- flatwork – for work with archives.

By using the command mkflat a directory with a name complying with an archive name, where files that are being packed into the archive and an archive description file <archive name>.!flatpack must be located, needs to be located in that place, where this archive is being created.

Unpacked files are placed into the folder "users\modwork"; it is reasonable to create it before starting a work with modifications.

Examples of command files for batch processing are given in the folder "docs\modwork\examples" and templates of settings files in "docs\modwork\stencil".

1.1 Work with game archives

To work with the game archives the commands mkflat and unflat are used (to create and unpack an archive). The archive files must have an extension flatdata.

An example of new archive creation:

```
starter.exe  mkflat,  users\modwork\my_addon.flatdata, users\modwork\my_addon.!flatlist
```

This command needs to be called from the root directory of the game. Hence, the archive my_addon.flatdata (in the folder "users\modwork") will be created, in which files listed in the file my_addon.!flatlist will be added.

The file containig a description of added files (my_addon.!flatlist) must be made under the following rules:

- it begins with a head i_unflat:unflat();
- A curly brace "{" must stay on the next line;
- Then a list of files, their format, and a local, for which they are actual, must be given; these parameters must be divided by commas;
- the file must end with a sign "}".

An example:

```
i_unflat:unflat()
{
	acrates                , config       , loc_def ;
	ai_plans               , config       , loc_def ;
	ammo                   , config       , loc_def ;
	anims                  , config       , loc_def ;
}
```

The file name is made as follows: <file name>.<local>.<type>

To unpack an existing archive the command unflat needs to be used.

```
starter.exe  unflat,  users\modwork\game_archive.flatdata, users\modwork\game_archive
```

This command unpacks the file game_archive.flatdata into the folder "users\modwork\game_archive" and creates a list of unpacked files game_archive.!flatlist, which can be used for the further packaing.

File types are listed in the Table 1.

Table 1

File Types

Extension	File Type
text	Text files
config	Configuration files
program	Description of commands and parts of programs
mesh	Geometry of objects
sound	Sounds and music
fontmap	Fontmaps
armor	Armor maps
image	Images
texture	Textures

1.2 Text files

The commands text2pd and pd2text are applied to work with text files.

An example of unpacking a text file:

```
starter.exe text2pd, users\modwork\text_file.loc_eng.text, users\modwork\ text_file.loc_eng.engcfg2
```

transforms the file text_file.loc_eng.text into the configuration file text_file.loc_eng.engcfg2. If the second parameter is not assigned, the received file will be located in the same folder, in which the unpacked one is located, but it will have an extension engcfg2.

A text represents a set of tables, each of them begins with an identificator consisting of Latin letters (lowercase) and figures; after it (in curly braces) one or several lines divided by ";" follow. All the tables are placed into a common block that determines a local.

A length of the table identificator must not exceed 31 signs.

For example:

```
//local
loc_reng()
{
	//table consisting of 1 line
	txt_text1[s]() { Текст №1; }
	txt_text2[s]() { Текст №2; }

	// table consisting of several lines
	txt_big_text[s]()
	{
		Table. Text 1;
		Table. Text 2;
		Table. Text 3;
		Table. Text 4;
	}

}
```

The signs "{", "}" and ";" are unallowable in the text. If a necessity to assign such signs as well as specific line feed and tabulation characters appears, it is necessary to use a precedence character "$". To assign tabulation - "$t", to feed a line - "$n", to assign a color - $<color number>: 1 – black, 2 – green, 3 – yellow, 4 – red, 5 – white, 6 – grey, 7 – blue, 8 – violet.

To pack a text file the next command is used:

```
starter.exe pd2text, users\modwork\text_file.loc_eng.engcfg2, users\modwork\ text_file.loc_eng.text
```

transforms the configuration file text_file.loc_eng.engcfg2 into the text file text_file.loc_eng.text. If the second parameter is not assigned, the received file will be located in the same folder, where the unpacked one is located, but it wll have an extension text.

An example of assignment of text configuration file stencil\text_example.loc_eng.engcfg2

1.3 Settings Files

Commands cfgp2pd and pd2cfgp are used for work with settings files.
An example of unpacking a settings file:

```
starter.exe cfgp2pd, users\modwork\tab.loc_def.config, users\modwork\tab.loc_def.engcfg2
```

transforms a settings file tab.loc_def.text into a configuration file tab.loc_def.engcfg2. If the second parameter is not assigned, the received file will be located in the same folder as an unpacked one, but it will have an extension engcfg2.

Settings represent a set of two blocks of two types: a list of constants and a table. Each block begins with an identificator consisting of Latin letters (lowercase) and figures; after it (in curly braces) several lines divided by ";" follow. A name and a format (in square brackets), after which a value of the constant with a character "=" stays, must be indicated for each constant.

A length of an identificator of block or constant name must not exceed 31 characters.

A format for each table cell must be indicated in the table name (in square brackets) and a character "=" – for the list of constants. A format of configuration file is considered in Section 3.2.

To pack a settings file the following command is used:

```
starter.exe pd2cfgp, users\modwork\tab.loc_def.engcfg2, users\modwork\tab.loc_def.config
```

transforms a configuration file tab.loc_def.engcfg2 into a settings file tab.loc_def.config. If the second parameter is not assigned, the received file will be located in the same folder as an unpacked one, but it will have an extension config.

An example of assignment of a configuration file meant for a description of settings stencil\desc_example.addpack.engcfg2.

1.4 Game Resources

To store game resources specific formats are used: ATF – for storage of textures and images, AAF - for storage of sounds and music, GO2 - for storage of geometry of objects. These formats are not meant for immediate changing and editing. Thus, to work with them it is necessary to transform these formats of files into other formats meant for immediate editing. After editing an inverse transformation is performed.

1.4.1 Textures and Images

To convert textures commands atf2dds and dds2atf are meant that allow to transform textures from the format ATF into DDS and vice versa.

An example of transformation of a texture:

```
starter.exe atf2dds, users\modwork\reg_tex_dift.loc_def.texture, users\modwork\ reg_tex_dift.loc_def.dds
```

transforms a texture reg_tex_dift.loc_def.texture into reg_text_dift.loc_def.dds. If the second parameter is not assigned, the received file will be located in the same folder as an unpacked one, but it will have an extension dds.

To edit textures in the format dds a number of programs can be used:

1 Paint.NET, link http://www.getpaint.net/index.html;
2 GIMP, link http://gimp-win.sourceforge.net/stable.html;
  DDS plugin http://nifelheim.dyndns.org/~cocidius/dds/;
3 nVidia® plugin for Adobe® PhotoShop®
  http://developer.nvidia.com/object/photoshop_dds_plugins.html.

During editing textures it is necessary to draw attention at its format and availability of MIP-levels. These parameters must not be changed!

A suffix "dift" in the texture name means a diffuse map (RGB channels) and transparence map (A channel), a suffix "norsp" – normal map (RG channels), shininess map (A channel), and roughness map (B channel).

The characteristics of basic texture formats are listed in the Table 2.

Table 2

Texture Formats

Prefix	Format/ MIP levels	Description
bump	DXT5	texture bump (normals, shininess, and roughness)
trans	DXT5	diffuse textures with translucency
reg	DXT1	diffuse textures with 1-bit alpha-channel
lbump	DXT5	landscape textures
clouds	DXT5	clouds and horizon textures
detail	DXT5/1	detail textures
coc	DXT1	текстуры кабин (not used)
menu	DXT5	menu and interface textures
map	DXT5	map textures
mapback	DXT1	substrate of tactical map
font	DXT5/1	font images
uncomp	RGB8	uncompressed textures
user	DXT5	user images (not used)

1.4.2 Sounds and Music

To convert sounds in a format used by the game a command wav2aaf is meant that is meant for transformation of sounds into the format WAV.

An example of transformation of a sound:

```
starter.exe wav2aaf, users\modwork\my_snd.loc_def.wav, users\modwork\ my_snd.loc_def.sound
```

A format 44100 Hz (44KHz) 16-bit MONO is used for dimensional sounds (shots, explosions, vehicle sounds, etc.) and 44100 Hz (44KHz) 16-bit STEREO – for system sounds.

For music and background sounds tha format xWMA is used that can be received by means of an utility from DirectX SDK® xWMAEncode, through converatation of uncompressed sound file in the format WAV into the format 44KHz, 16-bit STEREO.

An example of transformation:

```
xWMAEncode -b 160000 amb_can_0.wa_ amb_can_0.wav
```

A file "amb_can_0.wa_" in the format WAV will be transformed into a file "amb_can_0.wav" in the format xWMA.

DirectX SDK can be downloaded under the following link: http://www.microsoft.com/downloads/details.aspx?FamilyID=b66e14b8-8505-4b17-bf80-edb2df5abad4&displaylang=en (553.3 Mb)

1.4.3 Change of Object Material

To change a material of existing geometrical objects (vehicles, soldiers, constructions, green, etc.) a command tex_changer is used.

An example of a change of material:

```
starter.exe tex_changer, users\modwork\mesh.loc_def.mesh, ginf_dift, ginf_norsp
```

The file of geometry (with extension mesh) is transferred as a first parameter, the name of color and transparence textures (without prefix, but with suffix "dift") – as a second one. The normal, brightness, shininess, and roughness textures (without prefix, but with suffix "norsp") are transferred as a last parameter.

This command cannot be called without parameters!

1.4.4 Geometry of Objects

To convert a geometry from the format X a command x2go is meant. An example of the geometry transformation:

```
starter.exe x2go, users\modwork\my_meshl0.X, users\modwork\ my_mesh.loc_def.mesh
```

transforms the geometry my_meshl0.X, my_meshl1.X, …. into my_mesh.loc_def.mesh collecting all levels of detail into one file. If the second parameter is not assigned, the received file will be located in the same folder as an unpacked one, but it will have an extension mesh.

The levels of details and the physical level must be converted into the format X by means of intrinsics of the DCC-medium (ex., Blender) or by means of exterior plugins, ex. **Panda DirectX Exporter** that can be downloaded under the next link: http://www.andytather.co.uk/Panda/directxmax_downloads.aspx.

The file names must be as follows:

1) <name>l0.X for physical level;

2) <name>ln.X for visible levels of detail (которых должно быть 3 штуки), where n – a number of level from 1 to 3; **l – lower-case letter "L".**

While launching the command without parameters a file selection dialog appears where any level of details needs to be selected. All the levels of detail must be located in one folder!

To review the converted geometry (or any geometry from the game) a command "model_view" is used.

An example of the model review:

```
starter.exe model_view, users\modwork\ my_mesh.loc_def.mesh
```

displays an appearance of the model togeather with textures installed on it. If the command is launched without parameters, a window opens where the model for review needs to be selected. **Textures, applied on model, must be located in the game resources!**

1.5 Armor Maps

To transform armor maps from the format TGA into the format used by the game a cpmmand tga2am is meant.

An example of transformation of the armor map:

```
starter.exe tga2am, users\modwork\armor.loc_def.tga,
```

While calling the command without parameters a dialog appears where the file with the map can be selected. The TGA-format of the file must have 32bits. The channels R, G, and B must be identical, points having an armor must be indicated through white color in the alpha-channel, fully transparent point - through black one.

The converted armor maps must have an extension "armor".

2 ADDON CREATION

User addon must be structured as follows:
- a file readme.txt – text description of the addon;
- a folder CORE – contains addon files and installation information.

In the folder CORE a file "desc.addpack" must be contained in order to install the addon as well as folders loc_eng and shared where local and comman addon resources (in folders папки packed_data in the archives) are located. An instanse of the addon hierarchy is shown below:

```
CORE      |
          | LOC_ENG     |
          |             | PACKED_DATA |
          |                           | text_loc.flatdata
          | SHARED      |
          |             | PACKED_DATA |
          |                           | tabs.flatdata
          |                           | textures.flatdata
          |
          |
          | desc.addpack

readme.txt
```

The addon description (desc.addpack) can be created by means of editing and subsequent transformation of a template "stencil\desc_example.addpack.engcfg2".

An example of the addon installation template:

```
i_updater:updater=()
{
	//path to addon
	path[s] = <my_updates>;
	//addon name
	desc[s]  = <My Addon>;
	//addon author(s)
	authors[s] = <Vasya Pupkin>;
	//addon version
	version[u] = 100;
	//addon type
	//CAMP - camps/training ranges,
	//RES - resourse upgrade,
	//ADDN - addon
	type[*] =  RES;
	//delete previous addon version
	//it is recommended to assign true
	clear_prev[b] = true;
	//the game version, on which the addon is installed (in hexadecimal notation)
	//if a flag 0x80000000 is installed – installed
	//only on the specified version
	eng_ver[u] = 0x0000050b;
	//path to system files
	//(to leave void for the user addons)
	sys_path[s] = ;
	//the file being used to save replaceable files
	//to leave void
	recover[s] = ;

}
```

ATTENTION! The lines recorded in angle brackets (<>), need to be changed with own ones without angle brackets.

2.1 Using OS Command Files

To process some files it is desirable to use OS command files (text files with extebsion bat or cmd that allow to gradually perform some commands).

Create a text file файл my_addon.cmd in the root directory of the game. Insert in it commands required to assemble the addon using a text editor (notepad). Now it is sufficient to launch this file by means of file manager in order to reassemble the addon.

Examples of using command files:
- for work with archives unflat_example.cmd и mkflat_example.cmd;
- for work with text text2pd_example.cmd и pd2text_example.cmd;
- for work with settings cfgp2pd_example.cmd и pd2cfgp_example.cmd;

3 OTHERS

3.1 Locals

A local needs to be assigned for each resource being located into the archive. If the resource is used in all the game versions, the local must be "loc_def". Such resources as images with captions, texts, and fonts need to be indicated by means of the concrete local of the language, for which they have been created: loc_rus – Russian, loc_eng – English, loc_ger – German.

The common resources are located into the folder shared and local ones - in the folders with appropriate local names.

3.2 Configuration File Format

Each configuration file consists of one or several blocks. Each block may pertain to two types: constant list and table. Inside each constant list may be internal blocks.

Each block consists of a name (length up to 32 characters - lower-case Latin letters and figures). After the constant list name stays a character "=" and for table - format for each cell put into square brackets. The end of the name is marked with characters "( )".

A body of the block is put into curly brackets, in which the constants, tables, and internal blocks are located.

An instanse:

```
//constant list (character = points to it)
build0=()
{
	type[*] = BLD;
	mesh[s] = build01_s01_c0;
	mass[f]= 4000;
	dynamic[b] = true;
	j[v]	= 1, 1, 1, 0;
	no_coll[b] = false;
	imp[v] = 30000, 0, 20000, 30000;
	chunk[s] = d_base01;
	armor_map[s] = arm_ubuild_dift;
	armor_tal[f] = 25;
	material[s] = wood;
	force_max_mip[u] = 2;


	//table with format for each cell
	col_bounds[sfuf]()
	{
		d_coll_01, 0, 0, 0.2;
		d_coll_02, 0, 0, 0.2;

	} //endof col_bounds

} //endof build0
```

Each constant consist of a name (lower-case Latin letters, no more than 31 characters, taking into account a format) and a format put into square brackets. Then a character "=" and the constant value stay.

The allowable formats are shown in the Table 3.

Table 3

Constant Format

Format	Description
u	unsigned integer (in decimal or hexadecimal notation) or color
i	signed integer
b	logical constant, assumes 2 values: true or false
f, c	real number (to separate integral and fractional parts a character "." is used)
v	vector of 4 comma-separated real numbers
a	vector of 4 comma-separated integral numbers
*	FCC code (unsigned integer) – alphabetic code consisting of 2-4 upper case Latin letters

Comments in file are made by means of characters "//" for one-line comment and "/*" and "*/" for opening and closing a multiline comment. All the characters from "//" and to the end of the line are ignored in the one-line comment, and in the multiline - those that located between characters "/*" and "*/".

3.3 Summary Table of Commands

The command names for work with modifications are listed in the Table 4.

Table 4

Commands for Work with Modifications

Work with archives	
mkflat	Create an archive <name and path of archieve being created>, <name and path to description of files that will be added to archive>
unflat	Unpack files from the archive <name and path to archive file>, <path to folder where files being unpacked will be located>
Work with Text and Settings Files	
text2pd	Transform a text file into configuration file <name and path to text file>, <name and path of configurartion file>
pd2text	Transform a configuration file into text file <name and path of configuration file>, <name and path to text file>
cfgp2pd	Transform a settings file into configuration file <name and path to settings file>, <name and path of configuration file>
pd2cfgp	Transform a configuration file into settings file <name and path of configuration file>, <name and path to settings file>

Work with resources	
atf2dds	Convert texture from ATF format into DDS format <name and path of texture in ATF format>, <name and path to texture in DDS format>
dds2atf	Convert texture from DDS format into ATF format <name and path of texture in DDS format>, <name and path to texture in ATF format>
wav2aaf	Convert sound from WAV format into AAF format <name and path of sound in WAV format>, <name and path to sound in AAF format>
tex_changer	Change textures in material of object <name and path of geometry of object GO2>, <first texture without prefix and extension >, <second texture without prefix and extension>
tga2am	Convert texture from TGA format into format of armor maps <name and path of texture in TGA format>, <name and path to armor map>
x2go	Convert geometry from X-file into GO2 format <name and path of geometry in X format>, <name and path to geometry in format GO2 >
model_view	Review model in GO2 format <name and path of geometry in GO2 format>

3.4 File Functions

The main game files are given in the Table 5 and their functions are described.

Table 5

Game File Function

File	Description
Texts	
loc_kit	main game text
loc_encycl	texts of encyclopedia
loc_qbattle	texts for rapid battle editor
sold_fam_names	first names and surnames of soldiers
loc_redef	functions of buttons
Settings	
common_res_mod	game resources for modification (without duplication)
common_res	main game resources
ammo	ammunition stowage for technique
div_units	subdivisions, soldiers, technique, support (without duplication)
ger_hum_base	base of german soldiers
rus_hum_base	base of soviet soldiers
markers_01	captions and signs on technique
qbattle	subdivisions for rapid battle editor
season_ua_winter	season parameters (winter)
sound_base	sound group parameters
techn_base	base of technique parameters
ui_params	parameters for encyclopedia and epures
Game Archives	
effects	shaders
gos_builds	geometry of buildings
gos_misc	auxiliary geometrical objects
gos_objects	geometry of landscape objects
gos_techn	geometry of technique
music	music files
phys_maps	armor and fighting position maps

sounds	sounds
speech_ger	German speech
speech_rus	Russian speech
tabs	settings files
tex_humans	textures of soldiers
tex_misc	auxiliary textures of menu and some sprites
tex_dummy	textures of horizon and interface
tex_objects	textures of objects and buildings
tex_techns	textures of technique
Local Archives of the Game	
text_loc	text files
textures_loc	textures of fonts

4 HOW TO MAKE …?

4.1 10 Steps to Create the Simplest Mode

A purpose of mode is to add text for encyclopedia about tank T-34.

0) **Create a folder "users\modwork" if it is still not created.**

1) Create in the folder "users\modwork" a folder "test_mod".

2) Create in the folder "test_mod":

- a folder "test_pack".
- a text file desc.engcfg2 by copying it from a template "docs\modwork\stencil\desc_example.addpack.engcfg2" and renaming.

3) Fill in the created file desc.engcfg2 (see Section 2):

- path - a folder, in which a mode will be placed (low-case Latin letters wothout spaces);
- desc - a name of mode, which will be shown during installation (preferably - Latin letters);
- authors - a name of mode's author (preferably - Latin letters);
- version – a version of mode (100 is shown as 1.00);
- type – a type of addon (CAMP - camps/training ranges, RES - update of resources, ADDN – addon).

4) Create a text file **test_pack.!flatlist** in the folder "test_pack" and write there the fnext text (see Section 1.1):

```
i_unflat:unflat()
{
      t34_enc_text                , text        , loc_eng;
}
```

5) Create a text file t34_enc_text.loc_eng.engcfg2 in the folder "test_pack" and fill it with the next text (see Section 1.2):

```
loc_eng()
{
      txt_enci_t34_stz_m41[s]()
      {
            $t$5T-34 sample of 1941 year of manufacture, factory CT3.$n$t$4$n$n
            <…text of article….> ;
      }
```

}

6) Launch a command pd2text and select a file "t34_enc_text.loc_eng.engcfg2", as a result a packed text file "t34_enc_text.loc_eng.text" must appear.

7) Launch a command mkflat and ndicate a name of archieve file "test_pack.flatpack" **in the folder "test_mod"**. As a result this file must appear.

8) Launch a command pd2cfgp to create a description of the mode and select a file desc.engcfg2 in the folder "test_mod". As a result a file desc.config appear, which must be **renamed in desc.addpack**.

9) Collect the mode in one place (see Section 2) in a new folder "my_addon" and copy files in order to accomplish it:

- test_pack.flatdata in "my_addon\core\loc_eng\packed_data";
- desc.addpack in "my_addon\core";

10) Create a file readme.txt in the folder "my_addon", in which brief overview of the mode must be written.

As a result it must be as follows (the folder "my_addon"):

```
CORE       |
           | LOC_ENG    |
           |            | PACKED_DATA |
           |                          | test_pack.flatdata
           |
           | desc.addpack

readme.txt
```

The mode is ready and it can be installed by means of internal game utility!

4.2 Creation of New Subdivisions and Change of Existing Ones

A description of subdivisions is stored in a file div_units.loc_def.config, which is located in an archive tabs.flatpack. To extract it it is necessary:

1) To unpack an archieve tabs.flatpack (from patch) using a command unflat.

```
starter.exe root\programs\unflat.progpack, data\k43t\dev_updates\shared\packed_data\tabs.flatdata,
users\modwork\tabs_uf
```

2) To convert a file div_units.loc_def.config using a command cfgp2pd.

```
starter.exe root\programs\cfgp2pd.progpack, users\modwork\tabs_uf\div_units.loc_def.config,
```

Copy an unpacked settings file "div_units.loc_def.engcfg2" to other folder to work further. The subdivisions files act according to storage system, i.e. each installed patch or addon, where the file with such a name is located, adds a description of units or subdivisions to the common list. In case of duplication of the subdivisions the first one that was searched in the procedure of installation of patches and addons.

In a section **units()** the description of separate units of technique and soldiers (with prefix "rkkau_" – technique and soldiers of the USSR, and with prefix "weru_" – technique and soldiers of Germany) is located.

In a section **squads()** the description of divisions and technique togeather with calculations (with prefix "rkka_" – subdivisions of the USSR, and with prefix "wer_" – subdivisions of Germany) is located.

For example, we want **to create a new squad for the USSR with 5 soldiers with rifles and one sergeant and change a crew of the tank T-34 up to 3 persons.**

1) Delete contents of a block "units()", leaving only curly brackets and a block name because we cannot add new units of technique or soldiers.

2) Delete contents of a block "supports()" (same as previous).

3) Delete all lines from a block "squads()" except for the lines:

    rkka_squad_inf_43a, sq_inf, txt_ce_rkka_squad_inf_43a, ….;
    rkka_crew_t34, sq_crew, txt_ce_rkka_crew_t34, …..;

4) Rename in the first line:

    -"rkka_squad_inf_43a" in "rkka_squad_inf_43b",
    - "txt_ce_rkka_squad_inf_43a" in "txt_ce_rkka_squad_inf_43b".

5) Search for "rkkau_inf_sergant, 1" in the first line – from here on the list of units and technique that is included into the subdivision (here the subdivisions mentioned above may be indicated) begins. After each inclusion a number of

units/technique (in this case - 1 pcs.). Leave a sergeant and change the next record "rkkau_inf_rifle, 6" to "rkkau_inf_rifle, 5".

Delete the remaining records "rkkau_inf_arifle, 1, rkkau_inf_mgun, 1," by changing them to" , 0, , 0, ". Therefore, we have a division consisting of 6 persons with a name "rkka_squad_inf_43b".

6) Search for " rkkau_tank_agun, 2," in the second line "rkka_crew_t34" and change to " rkkau_tank_agun, 1,". Now the crew of the tank T-34 consists of 3 persons. In this case the name of the subdivision remains the same and changes the value of the original subdivision.

Additionally, it is necessary to create a text file with a table "txt_ce_rkka_squad_inf_43b" (see Section 1.2) containing two records separated by ";": "Squad" and " Rifle squad early 1943 №2". They will be shown in the list and the interface. This file must be included into the mode the same manner as the created file with description of subdivions.

The procedure of mode creation is described in details in the Section 4.1. **The created mode needs to be placed higher on the list, than updates from the developers.**

A new subdivision may be used as follows:

- adding it into reserves of the subdivions for rapid battle "qbattle.loc_def.config" (tabs.flatpack), for example, into a block "p_ussr_rd_04_550()" or in the structure of active platoon, for example, in a block "ussr_rd_rifles", by changing one of the subdivisions there;

- adding it into the structure of the subdivisions in one of the operations, it is necessary to convert a file in order to accomplish it:
"data\k43t\dev_updates\shared\camps\..\<opertaion>\<operation>.campack",
a command cfgp2pd, and add a subdivision into the reserve (the block "reserves") or into one of the active platoons (the block "act_platoons") and then to pack a file using a command pd2cfgp.

All the modified files needs to be added into the mode structure. The files of operation description do not need to be packed into the archieve, it is only necessary to assume a true path!