{{>TOC}}

*.ECD File format

Format *.ecd*(Essence Cascade Descriptor)* in fact is the text file in the UTF-8 (without BOM) encoding. (It is important, using any other encoding can lead to errors of reading and as a result to mistakes in logic of the application or even to become the reason of runtime exceptions). For work with this fils recomended to use Notepad++ editor, for version 6.3+ was maded syntax higlighting for *.ecd format. For installation use the file "ecd-highlight.xml" from xudevkit (in \toolsadd\notepad++\ folder), further in the main menu choose "Syntax" -> "Set the language... " in the appeared window press the Import button and choose the ecd-highlight.xml file. Now it is possible to close a window, at the bottom of Syntax menu must appear the ECD format, selecting which will applied syntax highlight to current file.

Main syntax

In the elementary case the file consists of sequence of couples **key value**, divided by a colon, thus value is any symbols since a colon and to the following managing director of a design. Before a key the symbol of a dollar of '$' is put. When reading all empty symbols are ignored (i.e. gaps, tabulation, etc.), therefore they can be used quietly for text formatting for its more evident visual representation. Syntax isn't sensitive to the register, and possible values of keys depend on the program handler of the file (i.e. admissible keys can change in different files or even in its parts). Each key can appropriate only one value, the following types of simple values are possible:

- Logical type - is the value allowing only two options "yes" or "no". For descriptive reasons the text some options of designation can be used: true\false, yes\no, on\off.
- Integer number - may contain at the beginning a sign "+" or "-" and as to begin with a prefix "#" in this case number is hexadecimal. By default it is considered that number positive in decimal system.
- Not integer number - surely has to contain the fractional part separated by a point (comma use instead of a point is a mistake) as well as the integer number may contain at the beginning a sign "+" or "-".
- String (text) - sequence of symbols the component has to be concluded a line in double quotes. In quotes use of any symbols, except quotes and square brackets ('[', ']'), including transfer of lines is allowed. Term can begin with "ru" locale. \"en". line defining belonging to this or that language. In case the locale isn't present that a line is universal for any language.
- Array (a set of values) - in certain cases keys can demand consecutive values (for example coordinate consists of value of X and Y) in this case the consecutive consists in braces, and values are divided by commas. The massif surely has to consist at least of two values.

Sometimes it is useful to make any marks for itself or for others, comments are for this purpose used. The comment begins with a symbol "~" and comes to an end from the beginning of a new line.
Example:

```
1  $btnEnable: true              ~ The logical type defining activity of the button
2  $btnRect:{300,220,200,40}     ~ The array setting an arrangement (x, y, width, height)
3  $btnScale: 0.5                ~ Button scale
4  $btnSoundId: #001A            ~ Id used a sound (#001A = 26)
5  $btnImage:"pic_menu_button"   ~ The file name with texture
6  $btnLabel:RU."Новая Игра"     ~ The button text for russian language
7  $btnLabel:EN."New Game"       ~ The button text for english language
```

Files * .ecd are read consistently therefore the order of an arrangement of elements can be important, for example in this case:

```
1  $btnLabel:RU."Новая Игра"     ~ The button text for russian language
2  $btnLabel:EN."New Game"       ~ The button text for english language
3  $btnLabel:"Unknown Button"    ~ The button text for other languages
```

Value of the text of the button always will be "Unknown Button" as the last description redefines made earlier. What to correct it it is necessary to place the last line in the beginning. In certain cases, such as the description of prompts, files are package gradually. In more detail about keys and formatting you look in the description of concrete files. Thow all depended on application logic, commonly it depends on **.ecd** puprporse, if it used like config - usualy is used first found valid key, if it used as script descriptor - last value will oveerride previous.

Shielding of symbols

For use of special symbols in lines the special symbol '\' that will transform a symbol conforming behind it on conforming to the rules serves:

Initial combination	\'	\<	\>	\-	*	\p	\a	\n	
Transformed by the parser	"	[	]	~	\	%	@	new line	

Syntax expansions

The expansions of syntax demanded for the description of some data are given below. It is necessary to remember that the logic of work of expansions entirely depends on the concrete file, the general syntax here is only described.

Sections

Above the elementary option, with linear structure was considered. In certain cases the file can be divided into the sections, each section isn't dependent and can consider as the separate file. As the beginning of new section its name serves in square brackets. After the announcement the section will proceed or to the announcement of the following section or until the end of the file. Example:

```
1 $btnLabel:"Next"    ~ The button text for the game menu
2 [HighScore]
3 $btnLabel:"Return"  ~ The button text for the account page in the game menu
```

Blocks

todo

Macros

todo

Functions

todo

Actions

todo