HEROES V
OF MIGHT AND MAGIC

The Basics of Heroes V Scripting

Version 2.01 – for Heroes V, Hammers of Fate and Tribes of the East

Version 1.0 of The Heroes V Scripting Guide, for Heroes 5
was written for Celestial Heavens by Pitsu

Proofreading and constructive criticism
by Grumpy Old Wizard, Gaidal Cain and Robenhagen

Version 2.0 of The Heroes V Scripting Guide
for the Tribes of the East expansion
is an update of Version 1.0 by rdeford, Mage of Soquim

Proofreading and constructive criticism by Pitsu and Robenhagen

Layout: Robenhagen

If you have feedback, questions or comments, please contact us at:
staff@celestialheavens.com

or post at the relevant topic on The Heroes Round Table:
http://www.celestialheavens.com/forums/viewforum.php?f=8

Foreword

Scripting your maps is a way to make them more interesting, and personal. Also, scripting allows you to control the course of the game to a much greater extent than you can with even the most creative terrain building. The Heroes of Might and Magic Editor (game patch 1.3 or later, or either the HOF or ToTE expansions required) allows you to add scripting to your maps. However, even though the game folder contains documentation for the using the Editor and describes the commands and script functions required to create scripts, it is easy to get lost when opening the script editing window and seeing the blank white page. Just what are you supposed to write there, and is there a special format or syntax for doing it?

This humble guide is meant as a supplementary material for the official editor documentation. For example, it does not describe all the individual script functions, since they already are quite well covered in official materials. But, it does cover the details of SCRIPT writing: the syntax, how to build a custom function and set triggers properly. However, even with this guide and the official editor documentation at hand, be ready to experience a little frustration when doing you first scripts ☺. Luckily, there are online forums where other people can help you during the darkest hours. I must admit that, without people asking for help and without other mapmakers demonstrating their neat solutions, this little guide could not have been written. My thanks to all the regulars of Round Table Mapmaking guild!

Wishing you good luck with scripts,
Pitsu

Table of Contents

1. Script file layout

When you create a new map and open its script file (View/Map Properties/Script tab, and click Edit Script) you see only a blank white sheet. When you open a heavily scripted map script file you see quite an extensive amount of text. Is there any particular layout to follow? The answer is no; the various elements in the script do not have to appear in any particular order, though there is a command syntax that must be adhered to.

Depending on the style of the script writer, a script file can have the following things in a chaotic array or an orderly arrangement:

- **Functions** – user-defined groups of pre-defined H5 script functions, commands, and Lua programming statements. They make up the main part of the script file. Basically, you must create one or more functions for each custom action or object behavior that you desire in your map. The scripts for lightly customized maps may require only one or two functions, while the scripts for highly customized maps require many functions.

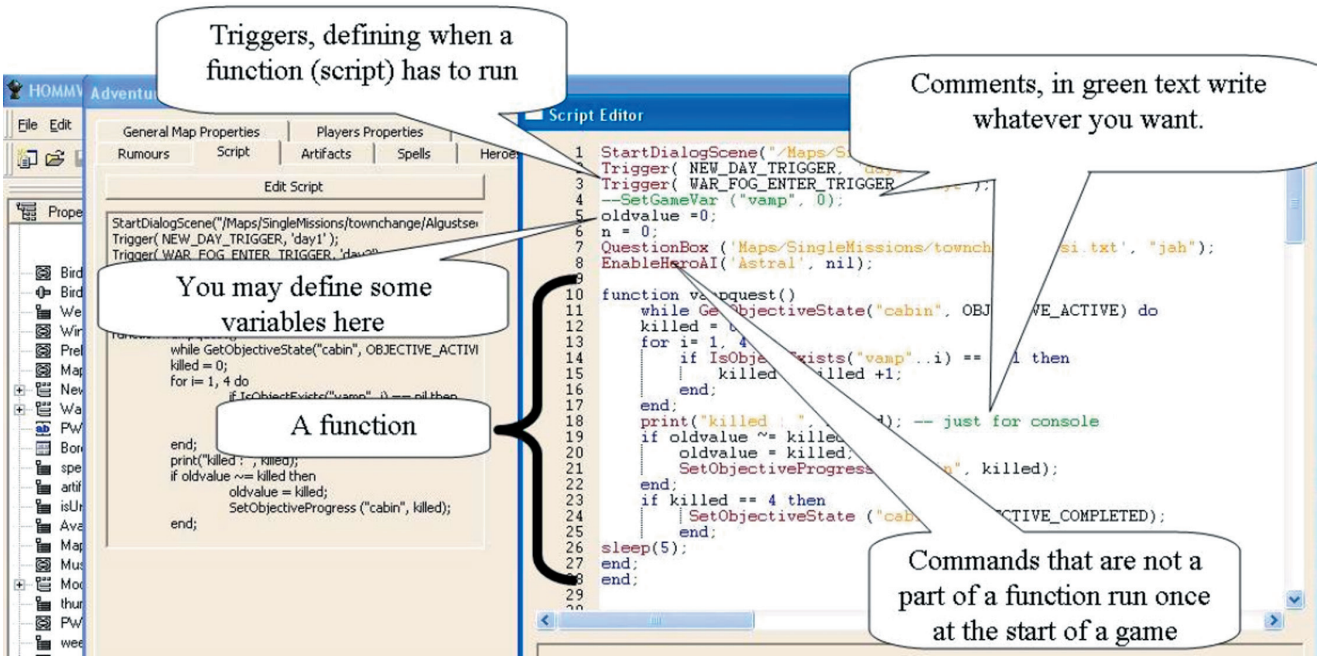
  Often it is best to split a complex function into several smaller functions just to have a better overview of the different parts.
- **Trigger** –a specialized pre-defined defined H5 script function that determines when a function in your script is run during the course of the game. As a general rule, you'll find you will need to tie a particular function to a specific action taken by the player while playing the map. For example, you may wish to display a warning message to the player when a hero first enters a certain area on your map.

Commands – a command is simply a complete, executable scripting statement of some kind. Most of the time commands are user-defined or pre-defined H5 script functions, but they can be any executable block of scripting statements consisting of any combination of Lua programming statements and functions.

Once defined in a script, functions become commands that can be used within the definitions of other functions in the script, or simply placed in the script outside any of its functions.

All commands in your script that are NOT part of any of the functions in your script are triggered when the map is started and the script file is read the first time. In other words, they are executed by the game on dawn of day 1, week 1, month 1 before any action can be taken by the player. For example, you may wish to show a message box that displays a prolog to the player before he or she begins playing.

- **Comments** – Comments start with double minus (double hyphen) -- and end with a line break. Comments are ignored by the game and are meant for script writer only. For example, if you are afraid of forgetting the meaning of a function in your script, simply add a comment to it. Also, placing double minus in front of a block of statements in your script is a simple way to test how your script runs without that block. For example, you may wish to see if a particular command is causing the bug you are hunting for.



Layout Example

Note the text color code shown in the example. This code helps a bit to avoid spelling and other mistakes. When it is green (comment), you can write anything, it is ignored by the game. When it is blue, it is a number or has a certain meaning for the Lua programming language. When it is red, it is a properly spelled pre-defined H5 script function. Text between a pair quotation marks (parameter names) is orange. The rest is black.

2. Syntax

2.1. Empty lines, semicolons, and the number of spaces

Each `end;` has to have a semicolon, and, although it is not a must, I recommend that you place a semicolon at the end of each line except the ones that start with `function`, `if`, `else`, `while`, `for`, or `repeat`.

You could write all the script in one row without any new lines, but for better tracking, you are advised to make a separate line for every individual command. And, for the sake of your sanity, indenting lines and groups of lines to improve readability and make relationships clear is an excellent habit to get into. (The Layout Example uses an excellent, systematic approach to indenting and the use of spaces. Go forth and do ye likewise.)

Breaking a long line of text into one or more lines, or adding half a page of empty lines between two functions does not affect the script's functionality.

2.2. Case sensitivity

Commands as well as variable names and map object names all are case sensitive. Spelling and capitalization mistakes are one of the most common errors when a script does not run.

2.3. Quotation marks and their use

Proper using of quotation marks with function names, variable names, and map object names is quite confusing, but it follows a system nevertheless. (You may need to read the function and parameter descriptions in the official documentation before fully understanding the samples presented here.) Also, note that there are two types of quotes that can be used " and '. Generally it does not matter which kind of quotation marks you use as long as the starting and ending marks of each pair are the same. A specific usage example for using both types of quotes in the same line is given later on in this guide under "Triggering fname(parameter) type of functions".

2.3.1. Use No Quotes When

- Declaring a function:

Example:

```
function blabla() -- blabla is without quotes.
   MessageBox("Maps/SingleMissions/test/
   blah.txt");
end;
```

- Typing numbers and the pre-defined H5 "all capital letters" parameters (like NEW_DAY_ TRIGGER or PLAYER_1).

- The function name acts as a command. Namely, each function that you write can be called up in the same way as the pre-defined H5 script functions.

Example

```
function blabla()
   MessageBox("Maps/SingleMissions/test/
   blah.txt");
end;
function day2()
   if GetDate(DAY)== 2 then
      blabla(); -- no quotes around blabla.
   end;
end;
```

- Typing the names of local and global variables, which are always without quotes.

NOTE: "Game variables," which are accessible via `SetGame Var()` and `GetGameVar()`, are different.

Example

```
function transform(townname) -- townname with-
                             out quotes since it is a variable.
local townype  =  5; -- defining a local variable
                  called towntype. No quotes any-
                  where.
TransformTown(townname, towntype); -- town-
          name and towntype are both variables. The
          value of the first is defined when the function
          is called up (see examples below), the value of
          the second was set to 5 right before.
end;
```

2.3.2. Use Quotes When

- The name of a function, adventure map object, scenario objective etc. is a parameter for a command. In other words, use quotes for everything that is inside parenthesis and is a name of a function, message text file, hero, adventure map object, or scenario objective.

Example

```
function blabla()
   if GetDate(DAY)== 2 then
      MessageBox("Maps/SingleMissions/
      test/blah.txt"); -- quotes required for the
                              text file name.
   end;
end;
Trigger(NEW_DAY_TRIGGER, "blabla"); -- quotes
                  required for function blabla name
```

- Calling up or saving game variables. (See official documentation for examples.)

2.3.3. Example script for quotes usage

Suppose you have a map with towns that have the script names town1 and town2. (In order to refer to an object on your map from within your script, you must give the object a script name in its properties window.) Then further suppose that you desire to change town1 to an Academy upon capture. As script code goes, this example is unnecessarily long, but it shows quotes usage quite well:

```
function transform(townname)  -- townname without quotes since it is a variable
    TransformTown(townname, TOWN_ACADEMY);
                  -- townname is a variable name and TOWN_ACADEMY is one of those all-capitals game variables.
end;
function town1capture()
    transform("town1"); -- transform, which is the name of another function, acts as a command. It uses parameter town1, which is the name of one of the towns.
end;
Trigger(OBJECT_CAPTURE_TRIGGER, "town1", "town1capture");  -- town1 and town1capture are names of map object and function, respectively. They are parameters for Trigger command and not variables, thus must have quotes.
```

2.4. How to create and refer to text files

It is very hard for beginners to get the references to external text files correct. Yet, you must learn to do it because each individual piece of text that appears during the course of playing your map must come from a text file that you have added to the map's resources during the creation of the map. For example, you will have to refer to a resource file that contains the text for each message box or a question box you wish the player to see.

To create a resource text file

1. Open the Map Properties Tree.
2. Select Resources and expand its list if necessary.
3. Select SavesFilenames.
4. Right click and select the Add command.
5. Expand the SaveFilenames list if necessary.
   A new file placeholder appears at the end of the list.
6. Expand the new file placeholder if necessary.
7. Click in the value field for the Name property and type descriptive name.
   For example: lavaWarning.
8. Click in the SaveFilenameFileRef value field, but do not type anything.
9. Click the New button.
   The Create New Object dialog box appears.
10. Type a file name for the new text file.
    TIP: Use the same name as you used for the Name property.
11. Click OK.
    A blank text entry window appears.
12. Type your message text.
13. Click OK and then Yes to save.

To refer to a text file you must provide the full path starting from Maps directory. (This full path appears in the SaveFilenameFileRef value field after you've created the file as described earlier.)

Example

Displaying a message to the player:

```
MessageBox("Maps/SingleMissions/MyMap/textfilename.txt");
-- MyMap is the name of your map.
-- textfilename.txt is a map resource text file name you created (lavaWarning.txt, for example).
```

TIP: You can copy the full path in the SaveFilenameFileRef value field and then paste it into the script when you wish to refer to the file.

2.4.1. Displaying variables within message text

CAUTION: The Descriptions given in official HoF editor manuals for `MessageBox()` and `QuestionBox()` for complex messages are misleading and erroneous. **Additionally**, you cannot display variables in any 1.x game version prior to the Hammers Of Fate expansion.

To display the value of a numeric variable within a complex message

1. Create a resource text file for the complex message using <value= variablename> at places where the value of the numeric variable should be displayed.

   Example

   Filename: `Pillage.txt`
   Text in file:
   ```
   The enemy has pillaged <value=goldamount> gold from local villages.
   ```

2. In the script, the `MessageBox()` command for the above example would look like:

   Format

   ```
   MessageBox( { "path/to/textfile"; variablename = value}, "optional_callback_function" );
   ```

   Example

   ```
   MessageBox( { "Maps/SingleMissions/MyMap/Pillage.txt "; goldamount = 526 });
   ```

3. In the game the complex message displayed to the player would look like:

> "The enemy has pillaged 526 gold from local villages."

To display a text variable within a complex message

1. Create a separate resource file for the text variable you wish to display in the complex message.

   Example

```
Filename: JonnyBlackeye.txt
Text in file: Jonny Blackeye
```

2. Create a resource text file for the complex message, using <value=variablename> at places where variables should be displayed.

   Example

```
Filename: Pillage.txt
Text in file: <value=name> has pillaged
              <value=goldamount> gold from
              local villages.
```

3. In the script, the MessageBox() command for the above example would look like:

   Example

```
MessageBox( { "Maps/SingleMissions/
MyMap/Pillage.txt "; name = "Maps/
SingleMissions/MyMap/JonnyBlackeye.txt",
goldamount = 526 });
```

   Notice that commas separate the elements in the variable list inside the {}.

4. In the game the message displayed to the player it would look like:

> "Jonny Blackeye has pillaged 526 gold from local villages."

2.5. How to refer to map objects

To refer to map objects other than heroes, you must give each object a unique scripting name.

To give an object a scripting name

1. Open the Map Properties Tree.
2. Select the object on the map.
   The object's property list appears in the Map Properties Tree.
3. Find the Name property and click in its value field.
   Two buttons, "…" and "New" appear.
4. Type a unique (to this map) name for the object.

TIP: Give the object a descriptive name that is easy to recognize and use. For example, rockBarrier, BlackForest, or firstbridge. Notice that capitals can make long names easier to read, but you do have to remember that all the names you create are case sensitive when you use them in the script.

Once you've completed these steps for an object, you can use its scripting name to refer to the object.

CAUTION: Giving the same, identical name to two or more objects, will not cause an error message right away, but the script will eventually crash when run.

2.6. How to refer to heroes

Referring to heroes must be done by their internal H5 ID. You can give them a scripting name as described earlier, but scripts do not recognize those names, nor do scripts recognize the standard game names.

To find a hero's internal ID

1. Open the Map Properties Tree
2. Select the hero on the map (may be a temporary placement of the hero).
   The hero's property list appears in the Map Properties Tree
3. Find the Shared property and click in its value field.
Two buttons, "…" and "New" appear.
4. Click on the "…" button.
   The Browse Object Link screen appears. A specific hero will be selected in the left window, and list of that hero's Properties will appear in the right window.
5. Find the InternalName property in the right window, and make note of the value, for it is the internal H5 ID for the hero.
6. Close the Browse Object Link screen without making any changes to the values.

If this procedure seems too complicated for you, then simply refer to the "Internal hero ID names needed for scripts" section towards the end of this guide.

2.7. Meaning of some symbols and special commands

The Lua programming language that is used in conjunction with the H5 script functions to create your scripts uses several symbols and provides a few special commands:

Syntax

Symbol or Command	Description
=	**Assignment** Assigns a value to a variable. *Example* m = 2, means that henceforth the variable m has a value of two.
==	**Equality** Compares a value on the left with the value on the right to see if they are equal. (Relational operator, returns true or false.) *Example* Where: x =2 y = 3 x == y would return false.
<	**Less than** Compares a value on the left with the value on the right to see if the left value is less than the right value. (Relational operator, returns true or false.)
>	**Greater than** Compares a value on the left with the value on the right to see if the left value is greater than the right value. (Relational operator, returns true or false.)
<=	**Less than or equal** Compares a value on the left with the value on the right to see if the left value is less than or equal to the right value. (Relational operator, returns true or false.)
>=	**Greater than or equal** Compares a value on the left with the value on the right to see if the left value is greater than or equal to the right value. (Relational operator, returns true or false.)
~=	**Not equal** Compares a value on the left with the value on the right to see if they are not equal. (Relational operator, returns true or false.) *Example* Where: x =2 y = 3 x ~= y would return true.
and	**Logical AND** Between two relational comparisons, means that both tests must be true in order to proceed. *Example* Where: x =2 y = 3 (x == 2 and y == 3) would return true.
or	**Logical OR** Between two relational comparisons, means that either of the tests can be true in order to proceed. *Example* Where: x =2 y = 3 (x == 4 or y == 3) would return true.
nil	A special, unique value generally meaning “none”. A global variable has a **nil** value by default before a first assignment. You can use **nil** as a kind of non-value, to disable a trigger by setting its target to “nothing.” In an equality test, only **nil** is equal to **nil**.

..	**String concatenation operator** (two dots) Joins two strings together. If any of its operands is a number, the string concatenation operator converts that number to a string. *Example* Where x = 8 "Mine"..x results in the string "Mine8"
random(nTop)	**Arithmetic function** Returns a random number from zero to the nTop - 1. *Example* Random(6) Returns a random in the range of 0 to 5.
mod(x, y)	**Arithmetic function** Returns the remainder from dividing the number x by the number y. Both arguments must be numbers (not necessarily whole). The second argument must be not equal to zero. *Example* Mod(42, 7) Returns: 0
+	**Addition** Adds the number on the left to the right *Example* Where: x =2 y = 3 x + y results in 5
-	**Subtraction** *Example* Where: x =2 y = 3 x - y results in -1
*	**Multiplication** *Example* Where: x =2 y = 3 x * y results in 6
/	**Division** *Example* Where: x =3 y = 2 x / y results in 1.5

3. Variables

A variable is a container in your script with a name and a value. It can have different values while the script is run. Do not confuse variable names with function names and names of adventure map objects, scenario objectives, file names, etc. Variable names are used and handled differently from these other names. When in doubt as to whether a name refers to a variable or not, check to see if there is a function with the same name, or an adventure map object or scenario objective with a scripting name that is the same. If there is not, it is most likely a variable name.

3.1. Variable types

There are three types of variables based on their accessibility.

1. **Local** variables are accessible only from inside a particular function.
2. **Global** variables accessible from anywhere in the script.
3. **Game** variables special-use global variables that are for transferring variables from one script file to another (in cases when a map has more than one script file).

You can transfer the values from one type to another.

Example

```
local m = GetGameVar("temp.gamevar1");
```

Assigns the value of a game variable to a local variable.

3.1.1. Local variables

Local variables are defined with the Lua local programming statement. They work only within the function where they are defined, and you can use the same local variable name in several different functions. There are specific cases when it is best to use local variables, but you can usually get by with just global variables.

Example

```
function day2()
local day =2; -- defines a local variable with value 2
   if GetDate(DAY)== day then -- comparison
                              returns true on day two.
      blabla();
   end;
end;

function day3() -- another function in the same script
                   file
   if GetDate(DAY)== day then -- error, since
                         variable "day" does not exist
                         in this function.
      blabla();
   end;
end;
```

3.1.2. Global variables

In actual scripting, global variables are used more than the other types of variables. You do not have to define them prior use. The first time you use a variable name outside of a function and give it a value, you have defined a global variable. From then on, it can be used at any place in the script file. However, for keeping better track of them and avoiding potential errors where you use a variable before giving it a value, you may wish to define them at the start of a script file and give them an initial value of the correct type.

Example

```
bankAccount = 0; -- defines a numerical type global
                    value named bankAccount and
                    assigns it a value of zero.

mainHero = "Aberrar"; -- defines a string type global
                         variable named mainHero and
                         assigns it a value of Aberrar.
```

3.1.3. Game variables

Game variables are required for transporting variable values between script files. For example if you have a separate combat script file along with the general map script, and you want to use the value of a variable in the main script in the combat script, then you would have to define and use a game variable.

Example

Main script:

```
SetGameVar("variablename",value)
```

Combat script:

```
xValue = GetGameVar("variablename")
```

4. Functions

4.1. Defining functions

Functions are the main part of a script file. Each function is essentially a group of H5 script functions and user-defined functions with Lua programming statements to determine the order in which those H5 script functions are executed.

The pre-defined H5 script commands that are available for your use are defined in the HOMM5_A2_Script_Functions.pdf, which is located in: `Program Files\Ubisoft\Heroes of Might and Magic V - Tribes of the East\Editor Documentation`.

You will use these pre-defined commands and Lua programming statements to define your own functions. You will likely have to define many functions if your map is to be customized to any great degree.

Simpler user-defined functions look like this:

```
function function_name()
   commands_of_your_choice;
end;
```

More complicated user-defined functions can have different blocks. For example:

```
function function_name()
   commands_that_run_always_when_the_
   function_is_triggered;
   if boolean_test then
      commands_that_run_only_if_boolean_
      test_returns_true;
   end;
again_commands_that_run_always_when_the_
function_is_triggered;
end;
```

And, you can have blocks within other blocks:

```
function function_name()
   commands_that_run_always_when_the_
   function_is_triggered;
   if boolean_test then
      commands_that_run_only_if_boolean_
      test_returns_true;
      if boolean_test2 then
         commands_that_run_only_if_
         boolean_test_and_boolean_ test2_
         BOTH_return_true;
      end;
   end;
end;
```

NOTE: The map Editor's Check Map and Check commands give you an ERROR "function xxx not defined" for each user-defined function. This error is meant only to confuse and scare you. Ignore it.

4.2. if ... then, while ... do etc. blocks

The basic most common types of code blocks are briefly described below. However, Lua, the language that Heroes 5 uses, is more powerful than this brief description indicates. In order to learn the other possibilities of these and other Lua commands, refer to the Lua manual found at (http://www.lua.org/pil/).

```
if boolean_test then
   commands;
end;
if boolean test then
   commands;
  else
   commands;
end;
while boolean test do
   commands;
end;
for variable name = start_value, end_value do
   commands;
end;
repeat
   commands;
until boolean test
```

4.3. How many "end;"s does there have to be?

An "end;" is required for each "block" that is started with:

- function
- if … then
- if … then … else
- while … do
- for … do

Everything that is between an above listed command and its respective end; statement forms a block, and all the commands in that block are run or none of them are run. See the examples under the general description of functions.

One block that does not require "end" is:

```
repeat
   commands;
until boolean test;
```

4.4. Function fname() or function fname(parameter)

Imagine a situation where you have several objects which trigger similar functions. For example, suppose your map had several towns that are converted to another town type upon capture. You could write a specific function for each town, but you could also write a general function and specify the town that has to be changed in the trigger for that function.

To write such a general function, you define variables in parenthesis after the function name. You can have more than one such variable there, separated by commas, if needed.

Example

The general function looks like this:

```
function transform(townname)
   TransformTown(townname, TOWN_ACADEMY);
end;
```

When calling `transform()` you must specify a value for the "townname" parameter, which is used like a local variable within the `transform()` function. Lets say the town script names are town1 and town2.

When you trigger this function by calling:

```
transform("town1");
```

The townname parameter in the `TransformTown()` function is given value "town1" and the general function `transform()` is understood as:

```
function transform()
   TransformTown("town1", TOWN_ACADEMY);
end;
```

When you trigger this function by calling:

```
transform("town2");
```

The very same general function `transform()` would be understood as:

```
function transform()
   TransformTown("town2", TOWN_ACADEMY);
end;
```

In the following example of bad scripting, the "townname" variable is not used for anything. It is defined, but the function always converts town2. Thus, it is more appropriate to define the function as `transform()` than it is to define the function as `transform(townname)`.

Bad scripting example

```
function transform(townname)
   TransformTown("town2", TOWN_ACADEMY);
end;
```

4.5. Special functionName(heroName) functions

Heroes have a special relationship with `functionName(parameter)` type of functions. Namely, one does not need to specify the specific hero to which a generic function applies. The ID of the currently selected hero who triggered the script is used as the value for what ever parameter name is used inside the parenthesis when the function is defined.

Examples

```
function blockHero(heroName) -- the script will
                use the ID of the current hero as the
                value of the heroName parameter.
   if HasArtefact(heroName, 2 ) then -- tests
         whether current hero (whose ID is in the
         heroName variable) has artefact number 2
      SetRegionBlocked("passage",not nil,
      PLAYER_3);
   end;
end;
```

The parameter can have any name you wish. Just remember that you must use that same name as the name of the variable within the commands in the function.

```
function blockHero(xxx) -- the script will use the
  ID of the current hero as the value of the xxx parameter.
   if HasArtefact(xxx, 2 ) then -- tests wheth-
                er current hero (whose ID is in the
                xxx variable) has artefact number 2.
      SetRegionBlocked("passage",not nil,
      PLAYER_3);
   end;
end;
```

This example function can be called up without specifying the hero name like with this command:

```
blockHero();
```

or

```
Trigger(REGION_ENTER_AND_STOP_TRIGGER,
"area1"," blockHero " );
```

See also the "Triggering fname(parameter) type of functions" section.

4.6. Threads

Threads are similar to continuous events in Heroes IV. They can kick in at any time and they happily use your CPU resources. A thread, once triggered, can run forever and in parallel to other functions. To be used as a successful thread a function must have the following structure:

```
function funcname()
   while boolean test do
      commands;
      sleep(10);
   end;
end;
```

As long as the boolean test (e.g., x == y) returns true, commands are run. Most often you need the opposite: commands are held back until a certain requirement is fulfilled. For that requirement use the following:

```
function funcname()
   while boolean test do
      if boolean test2 then
         commands;
      end;
      sleep(10);
   end;
end;
```

If you have problems defining the first boolean test which determines how long the thread lives, give it eternal life with "while 1 do". Official maps use this statement quite often.

Why is the `sleep(10)` command necessary? During the time specified by its parameter, the `sleep()` command allows the game engine to do other processes such as move heroes, operate eye candy, detect object contacts, display UI screens, run other script functions in parallel to the thread, etc. Consequently, the `sleep()` command must be in the thread loop or the thread will cause the game to slow down and lag, sometimes to the point where it is not playable. The higher the number the more time you give for other processes. A value of 10 is works well most of the time, but any value between 5 and 20 is reasonable in my opinion. However, there may be special cases where other values may be required, and you will have to try a few experiments to see what works best.

Triggering threads goes via `startThread(function name)` command.

CAUTION: Triggering threads in ways other than `startThread()` blocks access to other script functions as long as the (infinite) thread runs.

Example script

You want a script that gives the player an instant win when he accumulates 100.000 gold.

```
function checkGold()
   while 1 do
      if GetPlayerResources(PLAYER_1, 6)
      >= 100000 then
         win();
      end;
      sleep(10);
   end;
end;
startThread(checkGold); -- Note exception to rule:
      It is startThread(fname) NOT startThread("name")
```

5. Triggers

Another very important part of a script is to set triggers so that each function is run when you wish it run. While you can call a function up instantly by using its name as a command in the script or in the definition of another function, it is just as likely that you will wish to define the conditions when the function has to run. For example, you wish to trigger the display of a informational message to the player when a hero enters a certain region on the map. The text that follows discusses the techniques for handling both these situations.

5.1. Instant triggering of functions

A function can be called up within another function as a command within the function definition, or it can be placed in the script outside any functions and be used as a map starting event. In either case, you simply type the function name.

Example

```
function blabla()
   MessageBox("Maps/SingleMissions/test/
   blah.txt");
end;

function day2()
   if GetDate(DAY)== 2 then
      blabla(); -- here function blabla is triggered
                   and the messagebox is shown at day 2.
   end;
end;
```

5.2. Binding functions to map events and objects

Suppose you require a function to run when a certain event occurs on your map. How do you tie the function to the event? That is what you have the `Trigger()` command for. The `Trigger()` command is thoroughly described in the official editor document Script_Functions.pdf, and you should consult that document before you actually use the `Trigger()` command in your script. Briefly, the generalized syntax for the `Trigger()` command looks like this:

```
Trigger(EVENT _TYPE, "object to which the
event is bound", "function name that is to
be triggered");
```

Examples

```
Trigger(OBJECT_TOUCH_TRIGGER,
"festivalArena", "offerFestival");

Trigger(REGION_ENTER_AND_STOP_TRIGGER,
"monolithOneRgn", "openMonoliths");

Trigger(NEW_DAY_TRIGGER, "dawn");
```

When the `Trigger()` command is not a part of a function definition, the function specified in `Trigger()` command's last parameter is bound to the trigger when the map is first opened for playing. When the `Trigger()` command is part of a function definition, the link between the function specified in the `Trigger()` command's last parameter and the trigger is created when the defined function runs.

Only one function can be bound to each object and event combination. For example you cannot trigger two functions simultaneously by defining:

```
Trigger(OBJECT_CAPTURE_TRIGGER, "town1",
"function1");

Trigger(OBJECT_CAPTURE_TRIGGER, "town1",
"function2");
```

or

```
Trigger(NEW_DAY_TRIGGER, "day1");

Trigger(NEW_DAY_TRIGGER, "day2");
```

In both cases only the latter Trigger command would work, since it overwrites the first. Luckily you can do an intermediate function that triggers as many other functions as you wish.

Example

```
Trigger(NEW_DAY_TRIGGER, "days");
function days()
   day1();
   day2();
end;
function day1();
   commands;
end;
function day2();
   commands;
end;
```

Of course different objects can have the same type triggers without problems and the same object can have different type of triggers without problems.

5.3. Triggering fname(parameter) type of functions

What if the function being triggered has a parameter that you need to specify? As explained above, in cases where the parameter refers to currently selected hero, specification is actually not needed and therefore there is no complication. In other cases, you could avoid using parameters within the `Trigger()` command's parameters (long way), or you could use different sets of quotation marks within the `Trigger()` command's parameters.

Example of the long way

```
Trigger(NEW_DAY_TRIGGER, "transform1");

function transform1();
   transform("town1");
end;

function transform(townname)
   TransformTown(townname, 5);
end;
```

Example of the short way

```
Trigger(NEW_DAY_TRIGGER, "transform
('town1')"); -- NOTE: Different types of quota-
                tion marks are used. Check that
                "transform1('town1')" is entirely
                orange colored in your editor!

function transform(townname)
   TransformTown(townname, 5);
end;
```

5.4. Removing a trigger

To remove a trigger use "nil" instead of a function name as the last parameter in the `Trigger()` command.

Example command

```
Trigger(EVENT _TYPE, "object to which the
event is bound", nil);
```

Example script

At day 7 a town called town1 turns into academy and thereafter the trigger is removed:

```
Trigger(NEW_DAY_TRIGGER, "transform");

function transform()
   if GetDate(DAY) == 7 then
      TransformTown("town1", 2);
      Trigger(NEW_DAY_TRIGGER, nil);
   end;
end;
```

6. Hints for debugging

6.1. Enabling the console

It is extremely unlikely that you will be able to successfully write a working script without being able to see what errors you have made in writing it. Unfortunately, the H5 Editor's Check Map command does not actually find pertinent errors and, worse yet, reports errors that do not exist.

However, each time you launch the H5 game and open your map for playing by using the Create command on the main menu, the game compiles your scrip and checks for errors in the process. If it finds an error in your script, it will display an error message on the developer's console. If the error is bad enough, the script will be ignored when the map opens for play. This type of error is the main reason why new mapmakers complain "my script doesn't run" or "nothing happens" when I test my map.

Unfortunately, the developer's console is disabled during the installation process for the game, so you cannot see it while you play the game, nor can you use it to debug your script. But, there is a way to enable it.

For the Tribes of The East expansion, follow these steps:

1. Go into the `Program Files\Ubisoft\Heroes of Might and Magic V - Tribes of the East\profiles` directory.
2. Find the `autoexec_a2.cfg` file and open it for editing with a text editing program such as Notepad or WordPad.
3. Add this line of text `setvar dev_console_password = schwinge-des-todes` as the last line in the file.
4. Go into the `My Documents/My Games/Heroes of Might and Magic V - Tribes of the East/Profiles` directory. You will see some folders, each with the name of a game profile you created to play the game with. If you don't see any folders with profile names, launch the game and create a new profile with a name such as Tester.
5. Decide which profile you are going to use for testing your script and open its folder.
6. Find the `input_a2.cfg` file and open it for editing with a text editing program such as Notepad or WordPad.
7. Add the string `bind show_console '`'` right after the line saying `bind enter_pressed 'NUM_ENTER'`.

If you are using the Hammers of Fate expansion, you need to edit `autoexec_a1.cfg` and `cfg input_a1.cfg`.

If you are using just Heroes V, you need to edit `autoexec.cfg` and `input.cfg`.

Once you've enabled the console, go ahead and launch the game. Make sure you have selected the profile for which you modified the `input_a2.cfg` file and then use the Create command on the main menu to start playing your map. Once the game has compiled your script and opened your map for playing, you should be able to enter the console by pressing ` (the key above TAB).

If the console screen does not appear, go back over the steps and check your work. Even the most trivial typing error, capitalization error, or missing space will prevent enabling the console.

In addition to error messages for faulty scripts, the console shows you many things about the game, most of which are too obscure or cryptic to be of much use.

6.2. Using the print() command

With console activated, you can use the `print()` command to aid you in debugging your script. It prints the specified text or the value of a variable into the console window. You can put it pretty much anywhere between other commands to see if the script runs that far, or to display your own script tracking information.

Example

```
print('works to here'); -- shows works to here in
                                the console window.

print(test1); -- shows the value of the test1 variable
                    in the console window.

print("x1 = ", x1) -- shows x1 = followed by the
            value of the x1 variable in the console window.
```

6.3. My script does not run!

If your script does not run when you try to play your map, follow these steps:

1. Make sure that the map is SingleMission not MultiPlayer (open the map's *.h5m file with zip or rar archiver and see the names of subdirectories).
2. Enable console and read the error messages that are displayed there. Make notes on a piece of scratch paper so you can remember what they are.
3. Get out of the game and open your map with the Editor.
4. Look into your script and see if you can find the problems indicated in the error messages. Typically, an error message will tell you what line the error is on, or identify the last line the compiler worked on, and tell you what it expected to find versus what it did find. Examine your code carefully in light of this information, and decide what changes you wish to make before trying to compile it again. For example, you could do any or all of the following:
   a. Check your spelling and capitalization. Even the most trivial spelling or capitalization error can stop the compiler in its tracks.
   b. Check your use of quotes.
   c. Check that there are enough end;-s. A common problem is that the program does not understand properly where a function ends and therefore ignores all or part of the script file. It is a very good idea to use the TAB key to indent commands within each block of code. That way, for example, you will get a dotted line leading downwards from each if to the corresponding end; for that if. All the scripting examples in this guide use indenting in this manner. You are advised to do likewise.
   d. Check that you didn't use = where you should have used ==.

e. Reread each error message carefully, and try to understand what it is trying to tell you. Compare what you have done in your script to the documentation. Look through the Celestial Heavens' Mapmaking Guild Forum to see if anyone has found an answer to your problem. And, finally, post a description of your problem on the forum to see if someone can help you resolve it.

5. Make changes to your script and repeat these steps until you no longer have any reported errors.

6.4. My script runs but doesn't do what I want

Once your script compiles without errors, it is entirely possible that it still won't do what you wanted or expected. In this case, proceed as follows:

1. Study the results you do see in the game and then go into your script and look at the function that is supposed to be generating those results.
   a. Double-check triggers. Maybe a trigger is accidentally set to run under different conditions than you wish.
   b. Trace through the sequence of commands in the function and examine the logic to verify that it actually does what you desire. Make notes to track changing variable values, arithmetic results, comparison results, etc.
   c. Embed `print()` commands within the function to display interim results on the console, or just to verify that the correct function is being called and is working.
2. Devise a test and set up temporary conditions in the script that will test only that function, then launch the game and play your map while you verify the function's operation. Use embedded `print()` commands as required.
3. Make changes to the function and test it again. Repeat as necessary.
4. If all else fails, discard the function and rewrite it using a different approach.

7. Internal hero ID names needed for scripts

Referring to heroes in your scripts must be done by their internal H5 ID names. You can give them a scripting name as described earlier, but scripts do not recognize scripting names, nor do scripts recognize the standard game names as seen by the player in an tuncustomized map.

Here is a list of the standard game names (in **bold**) versus the internal hero ID names:

Bold Italic indicates names from the Tribes of the East expansion pack.

(The information in this table comes from Curios cheat guide http://www.heroesofmightandmagic.com/heroes5/heroes5_cheats_names.shtml)

Standard Game Name	Internal Hero ID
Agrael	Agrael
Alaron	Ildar
Alastor	Efion
Andreas	RedHeavenHero01
Anwen	Metlirn
Biara	Biara
Brand	Brand
Cyrus	Cyrus
Deirdre	Nemor
Deleb	Deleb
Dirael	Diraya
Dougal	Orrin
Duncan	Duncan
Ebba	Bersy
Ellaine	Nathaniel
Erasial	Erasial
Ergar	Ergar
Erling	Egil
Eruina	Eruina
Faiz	Faiz
Findan	Heam
Freyda	Axel
Freyda	Freyda
Galib	Tan
Garuna	Hero3
Giar	Giar
Gilraen	Gillion
Giovanni	Giovanni
Glen	Glen
Godric	Godric
Gorshak	Hero4
Gotai	Gottai
Grawl	Calid
Grok	Grok
Guarg	Guarg
Haggash	Hero7

Standard Game Name	Internal Hero ID
Havez	Havez
Helmar	Ottar
Inga	Una
Ingvar	Ingvar
Irina	Ving
Isabel	Isabell
Jezebeth	Oddrema
Jhora	Sufi
Karli	Skeggy
Kaspar	Gles
Kilghan	Hero9
King	Tolghar
Klaus	Sarge
Kragh	Hero1
Kujin	Kujin
Kunyak	Hero4
Kythra	Menel
Laszlo	Laszlo
Laszlo	Mardigo
Lethos	Dalom
Lorenzo	RedHeavenHero02
Lucretia	Tamika
Maahir	Maahir
Maeve	Maeve
Marbas	Marder
Markal	Berein
Naadir	Muscip
Narxes	Razzak
Nathir	Nur
Nebiros	Jazaz
Nicolai	Nicolai
Nur	Astral
Nymus	Nymus
Ornella	Ornella
Orson	Straker
Ossir	Ossir

Standard Game Name	Internal Hero ID
Quroq	Quroq
Raelag	Raelag
Raven	Effig
Razzak	Isher
Rolf	Rolf
Rutger	Brem
Segref	Segref
Shadya	Kelodin
Shak'Karukat	Hero6
Sinitar	Inagost
Sorgal	Ferigl
Svea	Vegeyr
Talanar	Nadaur
Telsek	Hero8
Temkhan	Timerkhan

Standard Game Name	Internal Hero ID
Thralsai	Thralsai
Urghat	Hero2
Valeria	RedHeavenHero03
Vayshan	Ohtarig
Vinrael	Elleshar
Vittorio	Christian
Vladimir	Pelt
Wulfstan	Wulfstan
Wyngaal	Linaas
Ylaya	Shadwyn
Ylthin	Itil
Yrbeth	Almegir
Yrwanna	Urunir
Zehir	Zehir
Zoltan	Aberrar

8. Other IDs Needed For Scripts

Many of the H5 script functions require IDs for parameter values to make them work properly. This section covers all the IDs you will need to write your scripts.

NOTE: Each expansion to the original H5 game brought new heroes, artifacts, and other objects. This list has notes indicating which IDs belong to each expansion where required.

NOTE: For some reason, the official documentation for the Editor was not updated for the new creature, town, and artifact IDs added by the Tribes of the East expansion. You must use this section for reference instead of the official documentation.

8.1. Logical constants

```
true = not nil
false = nil
```

8.2. Player IDs

```
PLAYER_NONE = 0
PLAYER_1 = 1
PLAYER_2 = 2
PLAYER_3 = 3
PLAYER_4 = 4
PLAYER_5 = 5
PLAYER_6 = 6
PLAYER_7 = 7
PLAYER_8 = 8
```

8.3. Player states

```
PLAYER_NOT_IN_GAME = 0
PLAYER_ACTIVE = 1
PLAYER_WON = 2
PLAYER_LOST = 3
```

8.4. Floor names

```
GROUND = 0
UNDERGROUND = 1
```

8.5. Kinds of resources

```
WOOD= 0
ORE = 1
MERCURY = 2
CRYSTAL = 3
SULFUR  = 4
GEM = 5
GOLD= 6
```

8.6. Kinds of treasures

```
TREASURE_CRYSTAL    = 0
TREASURE_GEMS   = 1
TREASURE_GOLD   = 2
TREASURE_MERCURY    = 3
TREASURE_ORE    = 4
TREASURE_SULFUR= 5
TREASURE_WOOD   = 6
TREASURE_CAMPFIRE   = 8
TREASURE_CHEST = 9
TREASURE_SEA_CHEST = 10
TREASURE_FLOATSAM   = 11
TREASURE_SHIPWRECK = 13
```

8.7. Date type IDs

```
DAY = 0
WEEK = 1
MONTH = 2
DAY_OF_WEEK = 3
ABSOLUTE_DAY = DAY
```

8.8. Advmap predefided names

```
OBJECT_GRAIL = 'grail'
```

8.9. Advmap object types

```
OBJECT_HERO = 0
```

8.10. Hero stat's IDs

```
STAT_EXPERIENCE = 0
STAT_ATTACK = 1
STAT_DEFENCE = 2
STAT_SPELL_POWER = 3
STAT_KNOWLEDGE = 4
STAT_LUCK = 5
STAT_MORALE = 6
STAT_MOVE_POINTS = 7
STAT_MANA_POINTS = 8
```

8.11. Objective state's IDs

```
OBJECTIVE_SCENARIO_INFO = 0
OBJECTIVE_UNKNOWN = 1
OBJECTIVE_ACTIVE = 2
OBJECTIVE_COMPLETED = 3
OBJECTIVE_FAILED = 4
```

8.12. Artifact sets IDs (for basic artifact sets)

```
ARTIFACT_SET_DRAGONISH = 0
ARTIFACT_SET_DWARVEN = 1
ARTIFACT_SET_LIONS = 2
ARTIFACT_SET_MAGIS = 3
ARTIFACT_SET_NECROMANCERS = 4
ARTIFACT_SET_EDUCATIONAL = 5
ARTIFACT_SET_HUNTERS = 6
ARTIFACT_SET_OGRES = 7
ARTIFACT_SET_RUNIC = 8
ARTIFACT_SET_DEMONIC = 9
```

8.13. Artifact type IDs

```
ARTIFACT_SWORD_OF_RUINS = 1
ARTIFACT_GREAT_AXE_OF_GIANT_SLAYING = 2
ARTIFACT_WAND_OF_X = 3
ARTIFACT_UNICORN_HORN_BOW = 4
ARTIFACT_TITANS_TRIDENT = 5
ARTIFACT_STAFF_OF_VEXINGS = 6
ARTIFACT_SHACKLES_OF_WAR = 7
ARTIFACT_FOUR_LEAF_CLOVER = 8
ARTIFACT_ICEBERG_SHIELD = 9
ARTIFACT_GOLDEN_SEXTANT = 10
ARTIFACT_CROWN_OF_COURAGE = 11
ARTIFACT_CROWN_OF_MANY_EYES = 12
ARTIFACT_PLATE_MAIL_OF_STABILITY = 13
ARTIFACT_BREASTPLATE_OF_PETRIFIED_WOOD = 14
ARTIFACT_PEDANT_OF_MASTERY = 15
ARTIFACT_NECKLACE_OF_BRAVERY = 16
ARTIFACT_WEREWOLF_CLAW_NECKLACE = 17
ARTIFACT_EVERCOLD_ICICLE = 18
ARTIFACT_NECKLACE_OF_POWER = 19
ARTIFACT_RING_OF_LIGHTING_PROTECTION = 20
ARTIFACT_RING_OF_LIFE = 21
ARTIFACT_RING_OF_HASTE = 22
ARTIFACT_NIGHTMARISH_RING = 23
ARTIFACT_BOOTS_OF_SPEED = 24
ARTIFACT_GOLDEN_HORSESHOE = 25
ARTIFACT_WAYFARER_BOOTS = 26
ARTIFACT_BOOTS_OF_INTERFERENCE = 27
ARTIFACT_ENDLESS_SACK_OF_GOLD = 28
ARTIFACT_ENDLESS_BAG_OF_GOLD = 29
ARTIFACT_ANGEL_WINGS = 30
ARTIFACT_LION_HIDE_CAPE = 31
ARTIFACT_PHOENIX_FEATHER_CAPE = 32
ARTIFACT_CLOAK_OF_MOURNING = 33
ARTIFACT_HELM_OF_ENLIGHTMENT = 34
ARTIFACT_CHAIN_MAIL_OF_ENLIGHTMENT = 35
ARTIFACT_DRAGON_SCALE_ARMOR = 36
ARTIFACT_DRAGON_SCALE_SHIELD = 37
ARTIFACT_DRAGON_BONE_GRAVES = 38
ARTIFACT_DRAGON_WING_MANTLE = 39
ARTIFACT_DRAGON_TEETH_NECKLACE = 40
ARTIFACT_DRAGON_TALON_CROWN = 41
ARTIFACT_DRAGON_EYE_RING = 42
ARTIFACT_DRAGON_FLAME_TONGUE = 43
ARTIFACT_ROBE_OF_MAGI = 44
ARTIFACT_STAFF_OF_MAGI = 45
ARTIFACT_CROWN_OF_MAGI = 46
ARTIFACT_RING_OF_MAGI = 47
ARTIFACT_DWARVEN_MITHRAL_CUIRASS = 48
ARTIFACT_DWARVEN_MITHRAL_GREAVES = 49
ARTIFACT_DWARVEN_MITHRAL_HELMET = 50
ARTIFACT_DWARVEN_MITHRAL_SHIELD = 51
ARTIFACT_SCROLL_OF_SPELL_X = 52
ARTIFACT_GRAAL = 53
ARTIFACT_BOOTS_OF_LEVITATION = 54
ARTIFACT_SKULL_HELMET = 55
ARTIFACT_VALORIOUS_ARMOR = 56
ARTIFACT_BOOTS_OF_SWIFTNESS = 57
ARTIFACT_MOONBLADE = 58
ARTIFACT_RING_OF_CELERITY = 59
ARTIFACT_BAND_OF_CONJURER = 60
ARTIFACT_EARTHSLIDERS = 61
ARTIFACT_RIGID_MANTLE = 62
ARTIFACT_JINXING_BAND = 63
ARTIFACT_BONESTUDDED_LEATHER = 64
ARTIFACT_WISPERING_RING = 65
ARTIFACT_HELM_OF_CHAOS = 66
ARTIFACT_TWISTING_NEITHER = 67
ARTIFACT_SANDALS_OF_THE_SAINT = 68
ARTIFACT_SHAWL_OF_GREAT_LICH = 69
ARTIFACT_RING_OF_DEATH = 70
ARTIFACT_NECROMANCER_PENDANT = 71
ARTIFACT_FREIDA = 72
ARTIFACT_RING_OF_THE_SHADOWBRAND = 73
ARTIFACT_OGRE_CLUB = 74
ARTIFACT_OGRE_SHIELD = 75
ARTIFACT_TOME_OF_DESTRUCTION = 76
ARTIFACT_TOME_OF_LIGHT_MAGIC = 77
ARTIFACT_TOME_OF_DARK_MAGIC = 78
ARTIFACT_TOME_OF_SUMMONING_MAGIC = 79
ARTIFACT_BEGINNER_MAGIC_STICK = 80
ARTIFACT_RUNIC_WAR_AXE = 81
ARTIFACT_RUNIC_WAR_HARNESS = 82
ARTIFACT_SKULL_OF_MARKAL = 83
ARTIFACT_BEARHIDE_WRAPS = 84
ARTIFACT_DWARVEN_SMITHY_HUMMER = 85
ARTIFACT_RUNE_OF_FLAME = 86
ARTIFACT_TAROT_DECK = 87
ARTIFACT_CROWN_OF_LEADER = 88
ARTIFACT_MASK_OF_DOPPELGANGER = 89
ARTIFACT_EDGE_OF_BALANCE = 90
ARTIFACT_RING_OF_MACHINE_AFFINITY = 91
ARTIFACT_HORN_OF_PLENTY = 92
ARTIFACT_RING_OF_UNSUMMONING = 93
ARTIFACT_BOOK_OF_POWER = 94
ARTIFACT_TREEBORN_QUIVER = 95
ARTIFACT_PRINCESS = 96
ARTIFACT_ARTIFACT_EFFECT_COUNT = 97
```

8.14. Skill type IDs

8.14.1. Basic Skills

```
SKILL_LOGISTICS = 1
SKILL_WAR_MACHINES = 2
SKILL_LEARNING = 3
SKILL_LEADERSHIP = 4
SKILL_LUCK = 5
SKILL_OFFENCE = 6
SKILL_DEFENCE = 7
SKILL_SORCERY = 8
SKILL_DESTRUCTIVE_MAGIC = 9
SKILL_DARK_MAGIC = 10
SKILL_LIGHT_MAGIC = 11
SKILL_SUMMONING_MAGIC = 12
```

8.14.2. Class skills

```
SKILL_TRAINING   = 13
SKILL_GATING = 14
SKILL_NECROMANCY = 15
SKILL_AVENGER = 16
SKILL_ARTIFICIER = 17
SKILL_INVOCATION = 18
```

8.15. Perks

```
PERK_PATHFINDING = 19
PERK_SCOUTING = 20
PERK_NAVIGATION = 21
PERK_FIRST_AID = 22
PERK_BALLISTA = 23
PERK_CATAPULT = 24
PERK_INTELLIGENCE = 25
PERK_SCHOLAR = 26
PERK_EAGLE_EYE = 27
PERK_RECRUITMENT = 28
PERK_ESTATES = 29
PERK_DIPLOMACY = 30
PERK_RESISTANCE = 31
PERK_LUCKY_STRIKE = 32
PERK_FORTUNATE_ADVENTURER = 33
PERK_TACTICS = 34
PERK_ARCHERY = 35
PERK_FRENZY = 36
PERK_PROTECTION = 37
PERK_EVASION = 38
PERK_TOUGHNESS = 39
PERK_MYSTICISM = 40
PERK_WISDOM = 41
PERK_ARCANE_TRAINING = 42
PERK_MASTER_OF_ICE = 43
PERK_MASTER_OF_FIRE = 44
PERK_MASTER_OF_LIGHTNINGS = 45
PERK_MASTER_OF_CURSES = 46
PERK_MASTER_OF_MIND = 47
PERK_MASTER_OF_SICKNESS = 48
PERK_MASTER_OF_BLESSING = 49
PERK_MASTER_OF_ABJURATION = 50
PERK_MASTER_OF_WRATH = 51
PERK_MASTER_OF_QUAKES = 52
PERK_MASTER_OF_CREATURES = 53
PERK_MASTER_OF_ANIMATION = 54
```

8.15.1. Knight perks

```
PERK_HOLY_CHARGE = 55
PERK_PRAYER = 56
PERK_EXPERT_TRAINER = 57
```

8.15.2. Demonlord perks

```
PERK_CONSUME_CORPSE = 58
PERK_DEMONIC_FIRE = 59
PERK_DEMONIC_STRIKE = 60
```

8.15.3. Necromancer perks

```
PERK_RAISE_ARCHERS  = 61
PERK_NO_REST_FOR_THE_WICKED = 62
PERK_DEATH_SCREAM   = 63
```

8.15.4. Ranger perks

```
PERK_MULTISHOT = 64
PERK_SNIPE_DEAD = 65
PERK_IMBUE_ARROW = 66
```

8.15.5. Wizard perks

```
PERK_MAGIC_BOND = 67
PERK_MELT_ARTIFACT = 68
PERK_MAGIC_MIRROR = 69
```

8.15.6. Warlock perks

```
PERK_EMPOWERED_SPELLS = 70
PERK_DARK_RITUAL = 71
PERK_ELEMENTAL_VISION = 72
```

8.16. Feats

8.16.1. Knight

```
KNIGHT_FEAT_ROAD_HOME = 73
KNIGHT_FEAT_TRIPLE_BALLISTA = 74
KNIGHT_FEAT_ENCOURAGE = 75
KNIGHT_FEAT_RETRIBUTION = 76
KNIGHT_FEAT_HOLD_GROUND = 77
KNIGHT_FEAT_GUARDIAN_ANGEL = 78
KNIGHT_FEAT_STUDENT_AWARD = 79
KNIGHT_FEAT_GRAIL_VISION = 80
KNIGHT_FEAT_CASTER_CERTIFICATE = 81
KNIGHT_FEAT_ANCIENT_SMITHY = 82
KNIGHT_FEAT_PARIAH = 83
KNIGHT_FEAT_ELEMENTAL_BALANCE = 84
KNIGHT_FEAT_ABSOLUTE_CHARGE = 85
```

8.16.2. Demon Lord

```
DEMON_FEAT_QUICK_GATING = 86
DEMON_FEAT_MASTER_OF_SECRETS = 87
DEMON_FEAT_TRIPLE_CATAPULT = 88
DEMON_FEAT_GATING_MASTERY = 89
DEMON_FEAT_CRITICAL_GATING = 90
DEMON_FEAT_CRITICAL_STRIKE = 91
DEMON_FEAT_DEMONIC_RETALIATION = 92
DEMON_FEAT_EXPLODING_CORPSES = 93
```

```
DEMON_FEAT_DEMONIC_FLAME = 94
DEMON_FEAT_WEAKENING_STRIKE = 95
DEMON_FEAT_FIRE_PROTECTION = 96
DEMON_FEAT_FIRE_AFFINITY = 97
DEMON_FEAT_ABSOLUTE_GATING = 98
```

8.16.3. Necromancer

```
NECROMANCER_FEAT_DEATH_TREAD = 99
NECROMANCER_FEAT_LAST_AID = 100
NECROMANCER_FEAT_LORD_OF_UNDEAD = 101
NECROMANCER_FEAT_HERALD_OF_DEATH = 102
NECROMANCER_FEAT_DEAD_LUCK = 103
NECROMANCER_FEAT_CHILLING_STEEL = 104
NECROMANCER_FEAT_CHILLING_BONES = 105
NECROMANCER_FEAT_SPELLPROOF_BONES = 106
NECROMANCER_FEAT_DEADLY_COLD = 107
NECROMANCER_FEAT_SPIRIT_LINK = 108
NECROMANCER_FEAT_TWILIGHT = 109
NECROMANCER_FEAT_HAUNT_MINE = 110
NECROMANCER_FEAT_ABSOLUTE_FEAR = 111
```

8.16.4. Ranger

```
RANGER_FEAT_DISGUISE_AND_RECKON = 112
RANGER_FEAT_IMBUE_BALLISTA = 113
RANGER_FEAT_CUNNING_OF_THE_WOODS = 114
RANGER_FEAT_FOREST_GUARD_EMBLEM = 115
RANGER_FEAT_ELVEN_LUCK = 116
RANGER_FEAT_FOREST_RAGE = 117
RANGER_FEAT_LAST_STAND = 118
RANGER_FEAT_INSIGHTS = 119
RANGER_FEAT_SUN_FIRE = 120
RANGER_FEAT_SOIL_BURN = 121
RANGER_FEAT_STORM_WIND = 122
RANGER_FEAT_FOG_VEIL = 123
RANGER_FEAT_ABSOLUTE_LUCK = 124
```

8.16.5. Wizard

```
WIZARD_FEAT_MARCH_OF_THE_MACHINES = 125
WIZARD_FEAT_REMOTE_CONTROL = 126
WIZARD_FEAT_ACADEMY_AWARD = 127
WIZARD_FEAT_ARTIFICIAL_GLORY = 128
 WIZARD_FEAT_SPOILS_OF_WAR = 129
WIZARD_FEAT_WILDFIRE = 130
WIZARD_FEAT_SEAL_OF_PROTECTION = 131
WIZARD_FEAT_COUNTERSPELL = 132
WIZARD_FEAT_MAGIC_CUSHION = 133
WIZARD_FEAT_SUPRESS_DARK  = 134
WIZARD_FEAT_SUPRESS_LIGHT  = 135
WIZARD_FEAT_UNSUMMON  = 136
WIZARD_FEAT_ABSOLUTE_WIZARDY = 137
```

8.16.6. Warlock

```
WARLOCK_FEAT_TELEPORT_ASSAULT = 138
WARLOCK_FEAT_SHAKE_GROUND = 139
WARLOCK_FEAT_DARK_REVELATION = 140
WARLOCK_FEAT_FAST_AND_FURIOUS = 141
WARLOCK_FEAT_LUCKY_SPELLS = 142
WARLOCK_FEAT_POWER_OF_HASTE = 143
WARLOCK_FEAT_POWER_OF_STONE = 144
WARLOCK_FEAT_CHAOTIC_SPELLS = 145
WARLOCK_FEAT_SECRETS_OF_DESTRUCTION = 146
WARLOCK_FEAT_PAYBACK = 147
WARLOCK_FEAT_ELITE_CASTERS = 148
WARLOCK_FEAT_ELEMENTAL_OVERKILL = 149
WARLOCK_FEAT_ABSOLUTE_CHAINS = 150
```

8.16.7. Runemage

```
HERO_SKILL_RUNELORE = 151
HERO_SKILL_REFRESH_RUNE = 152
HERO_SKILL_STRONG_RUNE = 153
HERO_SKILL_FINE_RUNE = 154
HERO_SKILL_QUICKNESS_OF_MIND = 155
HERO_SKILL_RUNIC_MACHINES = 156
HERO_SKILL_TAP_RUNES = 157
HERO_SKILL_RUNIC_ATTUNEMENT = 158
HERO_SKILL_DWARVEN_LUCK = 159
HERO_SKILL_OFFENSIVE_FORMATION = 160
HERO_SKILL_DEFENSIVE_FORMATION = 161
HERO_SKILL_DISTRACT = 162
HERO_SKILL_SET_AFIRE = 163
HERO_SKILL_SHRUG_DARKNESS = 164
HERO_SKILL_ETERNAL_LIGHT = 165
HERO_SKILL_RUNIC_ARMOR = 166
HERO_SKILL_ABSOLUTE_PROTECTION = 167
```

8.16.8. Runmage & Barbarian cross-class

```
HERO_SKILL_SNATCH = 168
HERO_SKILL_MENTORING = 169
HERO_SKILL_EMPATHY = 170
HERO_SKILL_PREPARATION = 171
```

8.16.9. Barbarian

```
HERO_SKILL_DEMONIC_RAGE = 172
HERO_SKILL_MIGHT_OVER_MAGIC = 173
HERO_SKILL_MEMORY_OF_OUR_BLOOD = 174
HERO_SKILL_POWERFULL_BLOW = 175
HERO_SKILL_ABSOLUTE_RAGE = 176
HERO_SKILL_PATH_OF_WAR = 177
HERO_SKILL_BATTLE_ELATION = 178
HERO_SKILL_LUCK_OF_THE_BARBARIAN = 179
HERO_SKILL_STUNNING_BLOW = 180
HERO_SKILL_DEFEND_US_ALL = 181
HERO_SKILL_GOBLIN_SUPPORT = 182
HERO_SKILL_BARBARIAN_LEARNING = 183
HERO_SKILL_POWER_OF_BLOOD = 184
HERO_SKILL_WARCRY_LEARNING = 185
HERO_SKILL_BODYBUILDING = 186
HERO_SKILL_VOICE = 187
HERO_SKILL_VOICE_TRAINING = 188
HERO_SKILL_MIGHTY_VOICE = 189
HERO_SKILL_VOICE_OF_RAGE = 190
HERO_SKILL_SHATTER_DESTRUCTIVE_MAGIC = 191
HERO_SKILL_CORRUPT_DESTRUCTIVE = 192
HERO_SKILL_WEAKEN_DESTRUCTIVE = 193
HERO_SKILL_DETAIN_DESTRUCTIVE = 194
HERO_SKILL_SHATTER_DARK_MAGIC = 195
HERO_SKILL_CORRUPT_DARK = 196
HERO_SKILL_WEAKEN_DARK = 197
HERO_SKILL_DETAIN_DARK = 198
HERO_SKILL_SHATTER_LIGHT_MAGIC = 199
```

```
HERO_SKILL_CORRUPT_LIGHT = 200
HERO_SKILL_WEAKEN_LIGHT = 201
HERO_SKILL_DETAIN_LIGHT = 202
HERO_SKILL_SHATTER_SUMMONING_MAGIC = 203
HERO_SKILL_CORRUPT_SUMMONING = 204
HERO_SKILL_WEAKEN_SUMMONING = 205
HERO_SKILL_DETAIN_SUMMONING = 206
HERO_SKILL_DEATH_TO_NONEXISTENT = 207
HERO_SKILL_BARBARIAN_ANCIENT_SMITHY = 208
HERO_SKILL_BARBARIAN_WEAKENING_STRIKE = 209
HERO_SKILL_BARBARIAN_SOIL_BURN = 210
HERO_SKILL_BARBARIAN_FOG_VEIL = 211
HERO_SKILL_BARBARIAN_INTELLIGENCE = 212
HERO_SKILL_BARBARIAN_MYSTICISM = 213
HERO_SKILL_BARBARIAN_ELITE_CASTERS = 214
HERO_SKILL_BARBARIAN_STORM_WIND = 215
```

8.17. Town type IDs

CAUTION: These values are for the TotE expansion only. If you are using H5 or HOF, you should check the documentation that came with each of those games.

Tribes of the East values

```
TOWN_HEAVEN = 0
TOWN_PRESERVE = 1
TOWN_ACADEMY = 2
TOWN_DUNGEON = 3
TOWN_NECROMANCY = 4
TOWN_INFERNO = 5
TOWN_FORTRESS = 6
TOWN_STRONGHOLD = 7
```

8.18. Town buildings IDs

```
TOWN_BUILDING_TOWN_HALL = 0
TOWN_BUILDING_FORT = 1
TOWN_BUILDING_MARKETPLACE = 2
TOWN_BUILDING_SHIPYARD = 3
TOWN_BUILDING_TAVERN = 4
TOWN_BUILDING_BLACKSMITH = 5
TOWN_BUILDING_MAGIC_GUILD = 6
TOWN_BUILDING_DWELLING_1 = 7
TOWN_BUILDING_DWELLING_2 = 8
TOWN_BUILDING_DWELLING_3 = 9
TOWN_BUILDING_DWELLING_4 = 10
TOWN_BUILDING_DWELLING_5 = 11
TOWN_BUILDING_DWELLING_6 = 12
TOWN_BUILDING_DWELLING_7 = 13
TOWN_BUILDING_GRAIL = 14
TOWN_BUILDING_WONDER = 15
TOWN_BUILDING_SPECIAL_0 = 16
TOWN_BUILDING_SPECIAL_1 = 17
TOWN_BUILDING_SPECIAL_2 = 18
TOWN_BUILDING_SPECIAL_3 = 19
TOWN_BUILDING_SPECIAL_4 = 20
TOWN_BUILDING_SPECIAL_5 = 21
TOWN_BUILDING_SPECIAL_6 = 22
TOWN_BUILDING_SPECIAL_7 = 23
TOWN_BUILDING_SPECIAL_8 = 24
TOWN_BUILDING_SPECIAL_9 = 25

TOWN_BUILDING_HAVEN_TRAINING_GROUNDS =
    TOWN_BUILDING_SPECIAL_1
TOWN_BUILDING_HAVEN_MONUMENT_TO_FALLEN_
    HEROES = TOWN_BUILDING_SPECIAL_2
TOWN_BUILDING_HAVEN_HOSPITAL = TOWN_
    BUILDING_SPECIAL_3
TOWN_BUILDING_HAVEN_STABLE = TOWN_
    BUILDING_SPECIAL_4
TOWN_BUILDING_HAVEN_FARMS = TOWN_
    BUILDING_SPECIAL_5
TOWN_BUILDING_INFERNO_INFERNAL_LOOM =
    TOWN_BUILDING_SPECIAL_1
TOWN_BUILDING_INFERNO_ORDER_OF_FIRE =
    TOWN_BUILDING_SPECIAL_3
TOWN_BUILDING_INFERNO_HALLS_OF_HORROR =
    TOWN_BUILDING_SPECIAL_4
TOWN_BUILDING_INFERNO_SACRIFICIAL_PIT =
    TOWN_BUILDING_SPECIAL_5
TOWN_BUILDING_DUNGEON_ALTAR_OF_ELEMENTS =
    TOWN_BUILDING_SPECIAL_1
TOWN_BUILDING_DUNGEON_RITUAL_PIT = TOWN_
    BUILDING_SPECIAL_3
TOWN_BUILDING_DUNGEON_TRADE_GUILD = TOWN_
    BUILDING_SPECIAL_4
TOWN_BUILDING_DUNGEON_TREASURE_DIG_SITE =
    TOWN_BUILDING_SPECIAL_5
TOWN_BUILDING_DUNGEON_HALL_OF_INTRIGUE =
    TOWN_BUILDING_SPECIAL_6
TOWN_BUILDING_ACADEMY_LIBRARY = TOWN_
    BUILDING_SPECIAL_1
TOWN_BUILDING_ACADEMY_ARCANE_FORGE = TOWN_
    BUILDING_SPECIAL_2
TOWN_BUILDING_ACADEMY_ARTIFACT_MERCHANT =
    TOWN_BUILDING_SPECIAL_3
TOWN_BUILDING_ACADEMY_TREASURE_CAVE =
    TOWN_BUILDING_SPECIAL_4
TOWN_BUILDING_ACADEMY_ELEMENTAL_ENCLAVE =
    TOWN_BUILDING_SPECIAL_5
TOWN_BUILDING_PRESERVE_AVENGERS_
    BROTHERHOOD = TOWN_BUILDING_SPECIAL_0
TOWN_BUILDING_PRESERVE_MYSTIC_POND = TOWN_
    BUILDING_SPECIAL_2
TOWN_BUILDING_PRESERVE_SPARKLING_FONTAINS
    = TOWN_BUILDING_SPECIAL_3
TOWN_BUILDING_PRESERVE_BLOOMING_GROVE =
    TOWN_BUILDING_SPECIAL_4
TOWN_BUILDING_PRESERVE_TREANT_SAMPLING =
    TOWN_BUILDING_SPECIAL_5
TOWN_BUILDING_NECROMANCY_AMPLIFIER = TOWN_
    BUILDING_SPECIAL_1
TOWN_BUILDING_NECROMANCY_UNHOLY_TEMPLE =
    TOWN_BUILDING_SPECIAL_2
TOWN_BUILDING_NECROMANCY_UNEARHED_GRAVES =
    TOWN_BUILDING_SPECIAL_3
TOWN_BUILDING_NECROMANCY_DRAGON_TOMBSTONE
    = TOWN_BUILDING_SPECIAL_4
```

```
TOWN_BUILDING_NECROMANCY_SHROUD_OF_
    DARKNESS = TOWN_BUILDING_SPECIAL_5
TOWN_BUILDING_FORTRESS_RUNIC_SHRINE =
    TOWN_BUILDING_SPECIAL_1;
TOWN_BUILDING_FORTRESS_ARENA = TOWN_
    BUILDING_SPECIAL_2;
TOWN_BUILDING_FORTRESS_GUARDPOST = TOWN_
    BUILDING_SPECIAL_3;
TOWN_BUILDING_FORTRESS_RUNIC_STONEWORKS =
    TOWN_BUILDING_SPECIAL_4;
TOWN_BUILDING_FORTRESS_RUNIC_ACADEMY =
    TOWN_BUILDING_SPECIAL_5;
TOWN_BUILDING_STRONGHOLD_HALL_OF_TRIAL =
    TOWN_BUILDING_SPECIAL_1
TOWN_BUILDING_STRONGHOLD_GARBAGE_PILE =
    TOWN_BUILDING_SPECIAL_2
TOWN_BUILDING_STRONGHOLD_TRAVELLERS_
    SHELTER = TOWN_BUILDING_SPECIAL_3
TOWN_BUILDING_STRONGHOLD_PILE_OF_OUR_FOES
    = TOWN_BUILDING_SPECIAL_4
TOWN_BUILDING_STRONGHOLD_SLAVE_MARKET =
    TOWN_BUILDING_SPECIAL_5
```

8.19. Monster mood IDs

```
MONSTER_MOOD_FRIENDLY = 0
MONSTER_MOOD_AGGRESSIVE = 1
MONSTER_MOOD_HOSTILE = 2
MONSTER_MOOD_WILD = 3
```

8.20. Monster courage IDs

```
MONSTER_COURAGE_ALWAYS_JOIN = 0
MONSTER_COURAGE_ALWAYS_FIGHT = 1
MONSTER_COURAGE_CAN_FLEE_JOIN = 2
```

8.21. Borderguard key colors

```
RED_KEY = 1
BLUE_KEY = 2
GREEN_KEY = 3
YELLOW_KEY = 4
ORANGE_KEY = 5
TEAL_KEY = 6
PURPLE_KEY = 7
TAN_KEY = 8
```

8.22. Trigger type IDs

```
NEW_DAY_TRIGGER = 0
PLAYER_ADD_HERO_TRIGGER = 1
PLAYER_REMOVE_HERO_TRIGGER = 2
OBJECTIVE_STATE_CHANGE_TRIGGER = 3
OBJECT_TOUCH_TRIGGER = 4
OBJECT_CAPTURE_TRIGGER = 5
REGION_ENTER_AND_STOP_TRIGGER = 6
REGION_ENTER_WITHOUT_STOP_TRIGGER = 7
HERO_LEVELUP_TRIGGER = 8
WAR_FOG_ENTER_TRIGGER = 9
COMBAT_RESULTS_TRIGGER = 11
CUSTOM_ABILITY_TRIGGER = 12
REGION_EXIT_TRIGGER = 13
```

8.23. Saved combat results types

```
COMBAT_RESULT_NONE = 0
COMBAT_RESULT_WIN = 1
COMBAT_RESULT_RETREAT = 2
COMBAT_RESULT_SURRENDER = 3
```

8.24. Custom abilities IDs

```
CUSTOM_ABILITY_1 = 1
CUSTOM_ABILITY_2 = 2
CUSTOM_ABILITY_3 = 3
CUSTOM_ABILITY_4 = 4
```

8.25. Custom abilities modes

```
CUSTOM_ABILITY_NOT_PRESENT = -1
CUSTOM_ABILITY_DISABLED = 0
CUSTOM_ABILITY_ENABLED = 1
```

8.26. Moon weeks (for GetCurrentMoon Week adventure script function)

```
WEEK_OF_NOTHING = 0
WEEK_OF_TOAD = 1
WEEK_OF_SALAMANDER = 2
WEEK_OF_BEETLE = 3
WEEK_OF_WYVERN = 4
WEEK_OF_DRAGONFLY = 5
WEEK_OF_FOX = 6
WEEK_OF_SHAMAN = 7
WEEK_OF_RABBIT = 8
WEEK_OF_SQUIRREL = 9
WEEK_OF_CATERPILLAR = 10
WEEK_OF_HAMSTER = 11
WEEK_OF_PIGEON = 12
WEEK_OF_RAGE = 13
WEEK_OF_EAGLE = 14
WEEK_OF_BEE = 15
WEEK_OF_WASP = 16
WEEK_OF_SWAN = 17
WEEK_OF_BUTTERFLY = 18
WEEK_OF_THANE = 19
WEEK_OF_ANTILOPE = 20
WEEK_OF_ORC = 21
WEEK_OF_RAVEN = 22
WEEK_OF_BADGER = 23
WEEK_OF_FLAMINGO = 24
WEEK_OF_TORTOISE = 25
WEEK_OF_LYNX = 26
```

```
WEEK_OF_PENGUIN = 27
WEEK_OF_FALCON = 28
WEEK_OF_HEDGEHOG = 29
WEEK_OF_SPARROW = 30
WEEK_OF_SWALLOW = 31
WEEK_OF_LION = 32
WEEK_OF_SPEARS = 33
WEEK_OF_BEAR = 34
WEEK_OF_CHIEFTAIN = 35
WEEK_OF_CENTAUR = 36
WEEK_OF_GOBLIN = 37
WEEK_OF_DEER = 38
WEEK_OF_OWL = 39
WEEK_OF_DEFENDER = 40
WEEK_OF_WYRM = 41
WEEK_OF_TIGER = 42
WEEK_OF_PLAGUE = 43
WEEK_OF_DISEASE = 44
WEEK_OF_FEVER = 45
WEEK_OF_FLAME = 46
WEEK_OF_WINDS = 47
WEEK_OF_FOLLY = 48
WEEK_OF_HONOR = 49
WEEK_OF_DIPLOMACY = 50
WEEK_OF_FORGERY = 51
WEEK_OF_TRADE = 52
WEEK_OF_MEDITATION = 53
WEEK_OF_LIFE = 54
WEEK_OF_CONJUNCTION = 55
WEEK_OF_JEWELS = 56
WEEK_OF_ALCHEMY = 57
WEEK_OF_GOLD = 58
WEEK_OF_FESTIVALS = 59
WEEK_OF_HARVEST = 60
WEEK_OF_IDLENESS = 61
WEEK_OF_MAGIC = 62
WEEK_OF_FEEBLENESS = 63
WEEK_OF_SORROW = 64
WEEK_OF_CHAOS = 65
WEEK_OF_CALM = 66
WEEK_OF_HOPE = 67
WEEK_OF_WATER = 68
WEEK_OF_FIRE = 69
WEEK_OF_EARTH = 70
WEEK_OF_AIR = 71
WEEK_OF_FIRENICE = 72
WEEK_OF_MIGHT = 73
WEEK_OF_BALANCE = 74
WEEK_OF_MIGHTNMAGIC = 75
WEEK_OF_INFIRMITY = 76
WEEK_OF_LIGHT = 77
WEEK_OF_EVOCATION = 78
WEEK_OF_ABJURATION = 79
WEEK_OF_ALTERATION = 80
WEEK_OF_CONJURATION = 81
WEEK_OF_ETHER = 82
WEEK_OF_TOUGHNESS = 83
WEEK_OF_PEASANT = 84
WEEK_OF_ARCHER = 85
WEEK_OF_FOOTMAN = 86
WEEK_OF_GRIFFIN = 87
WEEK_OF_PRIEST = 88
WEEK_OF_CAVALIER = 89
WEEK_OF_ANGEL = 90
WEEK_OF_GREMLIN = 91
WEEK_OF_GARGOYLE = 92
WEEK_OF_GOLEM = 93
WEEK_OF_MAGI = 94
WEEK_OF_GENIE = 95
WEEK_OF_RAKSHASA = 96
WEEK_OF_GIANT = 97
WEEK_OF_PIXIE = 98
WEEK_OF_WARDANCER = 99
WEEK_OF_WOODELF = 100
WEEK_OF_DRUID = 101
WEEK_OF_UNICORN = 102
WEEK_OF_TREANT = 103
WEEK_OF_GREENDRAGON = 104
WEEK_OF_IMP = 105
WEEK_OF_DEMON = 106
WEEK_OF_HELLHOUND = 107
WEEK_OF_SUCCUBUS = 108
WEEK_OF_NIGHTMARE = 109
WEEK_OF_BALOR = 110
WEEK_OF_DEVIL = 111
WEEK_OF_ASSASSIN = 112
WEEK_OF_WITCH = 113
WEEK_OF_MINOTAUR = 114
WEEK_OF_RIDER = 115
WEEK_OF_HYDRA = 116
WEEK_OF_MATRON = 117
WEEK_OF_DRAGON = 118
WEEK_OF_SKELETON = 119
WEEK_OF_WALKINGDEAD = 120
WEEK_OF_WIGHT = 121
WEEK_OF_VAMPIRE = 122
WEEK_OF_LICH = 123
WEEK_OF_GHOST = 124
WEEK_OF_BONEDRAGON = 125
```

8.27. Animation Action Types

```
INVISIBLE = 0
IDLE = 1
ONESHOT_STILL = 2
ONESHOT = 3
MOVE = 4
NON_ESSENTIAL = 5
```

8.28. Disabled interactive objects modes

```
DISABLED_DEFAULT = 0
DISABLED_ATTACK = 1
DISABLED_INTERACT = 2
DISABLED_BLOCKED = 3
```

8.29. Region Auto Action modes

```
REGION_AUTOACTION_ON_ENTER = 1
REGION_AUTOACTION_ON_EXIT = 2
```

8.30. Region Auto Action enable variants

```
REGION_AUTOACTION_OBJECT_NO_ACTION = -1
REGION_AUTOACTION_OBJECT_DISABLE = 0
REGION_AUTOACTION_OBJECT_ENABLE = 1
```

8.31. Heroes Roles Modes

NOTE: for SetHeroRoleMode(heroName,roleMode) script commmand

```
HERO_ROLE_MODE_REGULAR = 0
HERO_ROLE_MODE_FREEMAN = 1 -- always
    Freelancer
HERO_ROLE_MODE_HERMIT  = 2 -- always
    Freelancer, never interact with
    other heroes
```

8.32. Players Teams constants

```
PLAYERS_TEAM_1 = 1
PLAYERS_TEAM_2 = 2
PLAYERS_TEAM_3 = 3
PLAYERS_TEAM_4 = 4
PLAYERS_TEAM_5 = 5
PLAYERS_TEAM_6 = 6
PLAYERS_TEAM_7 = 7
PLAYERS_TEAM_8 = 8
```

8.33. Game difficulty IDs

```
DIFFICULTY_EASY = 0
DIFFICULTY_NORMAL = 1
DIFFICULTY_HARD = 2
DIFFICULTY_HEROIC = 3
```

8.34. Creature IDs

```
CREATURE_UNKNOWN = 0
CREATURE_PEASANT = 1
CREATURE_MILITIAMAN = 2
CREATURE_ARCHER = 3
CREATURE_MARKSMAN = 4
CREATURE_FOOTMAN = 5
CREATURE_SWORDSMAN = 6
CREATURE_GRIFFIN = 7
CREATURE_ROYAL_GRIFFIN = 8
CREATURE_PRIEST = 9
CREATURE_CLERIC = 10
CREATURE_CAVALIER = 11
CREATURE_PALADIN = 12
CREATURE_ANGEL = 13
CREATURE_ARCHANGEL = 14
CREATURE_FAMILIAR = 15
CREATURE_IMP = 16
CREATURE_DEMON = 17
CREATURE_HORNED_DEMON = 18
CREATURE_HELL_HOUND = 19
CREATURE_CERBERI = 20
CREATURE_SUCCUBUS = 21
CREATURE_INFERNAL_SUCCUBUS = 22
CREATURE_NIGHTMARE = 23
CREATURE_FRIGHTFUL_NIGHTMARE = 24
CREATURE_PIT_FIEND = 25
CREATURE_BALOR = 26
CREATURE_DEVIL = 27
CREATURE_ARCHDEVIL = 28
CREATURE_SKELETON = 29
CREATURE_SKELETON_ARCHER = 30
CREATURE_WALKING_DEAD = 31
CREATURE_ZOMBIE = 32
CREATURE_MANES = 33
CREATURE_GHOST = 34
CREATURE_VAMPIRE = 35
CREATURE_VAMPIRE_LORD = 36
CREATURE_LICH = 37
CREATURE_DEMILICH = 38
CREATURE_WIGHT = 39
CREATURE_WRAITH = 40
CREATURE_BONE_DRAGON = 41
CREATURE_SHADOW_DRAGON = 42
CREATURE_PIXIE = 43
CREATURE_SPRITE = 44
CREATURE_BLADE_JUGGLER = 45
CREATURE_WAR_DANCER = 46
CREATURE_WOOD_ELF = 47
CREATURE_GRAND_ELF = 48
CREATURE_DRUID = 49
CREATURE_DRUID_ELDER = 50
CREATURE_UNICORN = 51
CREATURE_WAR_UNICORN = 52
CREATURE_TREANT = 53
CREATURE_TREANT_GUARDIAN = 54
CREATURE_GREEN_DRAGON = 55
CREATURE_GOLD_DRAGON = 56
CREATURE_GREMLIN = 57
CREATURE_MASTER_GREMLIN = 58
CREATURE_STONE_GARGOYLE = 59
CREATURE_OBSIDIAN_GARGOYLE = 60
CREATURE_IRON_GOLEM = 61
CREATURE_STEEL_GOLEM = 62
CREATURE_MAGI = 63
CREATURE_ARCH_MAGI = 64
CREATURE_GENIE = 65
CREATURE_MASTER_GENIE = 66
CREATURE_RAKSHASA = 67
CREATURE_RAKSHASA_RUKH = 68
CREATURE_GIANT = 69
CREATURE_TITAN = 70
CREATURE_SCOUT = 71
CREATURE_ASSASSIN = 72
CREATURE_WITCH = 73
```

```
CREATURE_BLOOD_WITCH = 74
CREATURE_MINOTAUR = 75
CREATURE_MINOTAUR_KING = 76
CREATURE_RIDER = 77
CREATURE_RAVAGER = 78
CREATURE_HYDRA = 79
CREATURE_CHAOS_HYDRA = 80
CREATURE_MATRON = 81
CREATURE_MATRIARCH = 82
CREATURE_DEEP_DRAGON = 83
CREATURE_BLACK_DRAGON = 84
CREATURE_FIRE_ELEMENTAL = 85
CREATURE_WATER_ELEMENTAL = 86
CREATURE_EARTH_ELEMENTAL = 87
CREATURE_AIR_ELEMENTAL = 88
CREATURE_DEATH_KNIGHT = 90
CREATURE_PHOENIX = 91
CREATURE_DEFENDER = 92
CREATURE_STOUT_DEFENDER = 93
CREATURE_AXE_FIGHTER = 94
CREATURE_AXE_THROWER = 95
CREATURE_BEAR_RIDER = 96
CREATURE_BLACKBEAR_RIDER = 97
CREATURE_BROWLER = 98
CREATURE_BERSERKER = 99
CREATURE_RUNE_MAGE = 100
CREATURE_FLAME_MAGE = 101
CREATURE_THANE = 102
CREATURE_WARLORD = 103
CREATURE_FIRE_DRAGON = 104
CREATURE_MAGMA_DRAGON = 105
CREATURE_LANDLORD = 106
CREATURE_LONGBOWMAN = 107
CREATURE_VINDICATOR = 108
CREATURE_BATTLE_GRIFFIN = 109
CREATURE_ZEALOT = 110
CREATURE_CHAMPION = 111
CREATURE_SERAPH = 112
CREATURE_WOLF = 113
CREATURE_SNOW_APE = 114
CREATURE_MANTICORE = 115
CREATURE_MUMMY = 116
```

-- Creatures added by ToTE --

```
CREATURE_GOBLIN = 117
CREATURE_GOBLIN_TRAPPER = 118
CREATURE_CENTAUR = 119
CREATURE_CENTAUR_NOMAD = 120
CREATURE_ORC_WARRIOR = 121
CREATURE_ORC_SLAYER = 122
CREATURE_SHAMAN = 123
CREATURE_SHAMAN_WITCH = 124
CREATURE_ORCCHIEF_BUTCHER = 125
CREATURE_ORCCHIEF_EXECUTIONER = 126
CREATURE_WYVERN = 127
CREATURE_WYVERN_POISONOUS = 128
CREATURE_CYCLOP = 129
CREATURE_CYCLOP_UNTAMED = 130
CREATURE_QUASIT = 131
CREATURE_HORNED_LEAPER = 132
CREATURE_FIREBREATHER_HOUND = 133
CREATURE_SUCCUBUS_SEDUCER = 134
CREATURE_HELLMARE = 135
CREATURE_PIT_SPAWN = 136
CREATURE_ARCH_DEMON = 137
CREATURE_STALKER = 138
CREATURE_BLOOD_WITCH_2 = 139
CREATURE_MINOTAUR_CAPTAIN = 140
CREATURE_BLACK_RIDER = 141
CREATURE_ACIDIC_HYDRA = 142
CREATURE_SHADOW_MISTRESS = 143
CREATURE_RED_DRAGON = 144
CREATURE_DRYAD = 145
CREATURE_BLADE_SINGER = 146
CREATURE_SHARP_SHOOTER = 147
CREATURE_HIGH_DRUID = 148
CREATURE_WHITE_UNICORN = 149
CREATURE_ANGER_TREANT = 150
CREATURE_RAINBOW_DRAGON = 151
CREATURE_SKELETON_WARRIOR = 152
CREATURE_DISEASE_ZOMBIE = 153
CREATURE_POLTERGEIST = 154
CREATURE_NOSFERATU = 155
CREATURE_LICH_MASTER = 156
CREATURE_BANSHEE = 157
CREATURE_HORROR_DRAGON = 158
CREATURE_GREMLIN_SABOTEUR = 159
CREATURE_MARBLE_GARGOYLE = 160
CREATURE_OBSIDIAN_GOLEM = 161
CREATURE_COMBAT_MAGE = 162
CREATURE_DJINN_VIZIER = 163
CREATURE_RAKSHASA_KSHATRI = 164
CREATURE_STORM_LORD = 165
CREATURE_STONE_DEFENDER = 166
CREATURE_HARPOONER = 167
CREATURE_WHITE_BEAR_RIDER = 168
CREATURE_BATTLE_RAGER = 169
CREATURE_FLAME_KEEPER = 170
CREATURE_THUNDER_THANE = 171
CREATURE_LAVA_DRAGON = 172
CREATURE_GOBLIN_DEFILER = 173
CREATURE_CENTAUR_MARADEUR = 174
CREATURE_ORC_WARMONGER = 175
CREATURE_SHAMAN_HAG = 176
CREATURE_ORCCHIEF_CHIEFTAIN = 177
CREATURE_WYVERN_PAOKAI = 178
CREATURE_CYCLOP_BLOODEYED = 179
CREATURES_COUNT = 180
```

8.35. War machines IDs

```
WAR_MACHINE_BALLISTA = 1
WAR_MACHINE_CATAPULT = 2
WAR_MACHINE_FIRST_AID_TENT = 3
WAR_MACHINE_AMMO_CART = 4
```

8.36. Spell IDs

```
SPELL_NONE = 0;
SPELL_MAGIC_ARROW = 1;
SPELL_MAGIC_FIST = 2;
SPELL_LIGHTNING_BOLT = 3;
SPELL_ICE_BOLT = 4;
SPELL_FIREBALL = 5;
SPELL_FROST_RING = 6;
SPELL_CHAIN_LIGHTNING = 7;
SPELL_METEOR_SHOWER = 8;
SPELL_IMPLOSION = 9;
SPELL_ARMAGEDDON = 10;
SPELL_CURSE = 11;
SPELL_SLOW = 12;
SPELL_DISRUPTING_RAY = 13;
SPELL_PLAGUE = 14;
SPELL_WEAKNESS = 15;
SPELL_ABILITY_WEAKNING_STRIKE = 16;
SPELL_FORGETFULNESS = 17;
SPELL_BERSERK = 18;
SPELL_BLIND = 19;
SPELL_HYPNOTIZE = 20;
SPELL_UNHOLY_WORD = 21;
SPELL_UNHOLY_WORD_HIT_EFFECT = 22;
SPELL_BLESS = 23;
SPELL_HASTE = 24;
SPELL_STONESKIN = 25;
SPELL_DISPEL = 26;
SPELL_DISPEL_FAIL = 27;
SPELL_BLOODLUST = 28;
SPELL_DEFLECT_ARROWS = 29;
SPELL_DEFLECT_ARROWS_HIT_EFFECT = 30;
SPELL_ANTI_MAGIC = 31;
SPELL_TELEPORT = 32;
SPELL_TELEPORT_FINISH_EFFECT = 33;
SPELL_CELESTIAL_SHIELD = 34;
SPELL_HOLY_WORD = 35;
SPELL_HOLY_WORD_HIT_EFFECT = 36;
SPELL_ARMAGEDDON_HIT_EFFECT = 37;
SPELL_LAND_MINE = 38;
SPELL_WASP_SWARM = 39;
SPELL_PHANTOM = 40;
SPELL_EARTHQUAKE = 41;
SPELL_ANIMATE_DEAD = 42;
SPELL_SUMMON_ELEMENTALS = 43;
SPELL_EFFECT_SUMMON_ELEMENTALS_AIR = 44;
SPELL_EFFECT_SUMMON_ELEMENTALS_EARTH = 45;
SPELL_EFFECT_SUMMON_ELEMENTALS_FIRE = 46;
SPELL_EFFECT_SUMMON_ELEMENTALS_WATER = 47;
SPELL_RESURRECT = 48;
SPELL_SUMMON_BOAT = 49;
SPELL_DIMENSION_DOOR = 50;
SPELL_TOWN_PORTAL = 51;
SPELL_ENCOURAGE = 52;
SPELL_HOLY_CHARGE = 53;
SPELL_PRAYER = 54;
SPELL_DEMONIC_STRIKE_CAST = 55;
SPELL_DEMONIC_STRIKE = 56;
SPELL_CONSUME_CORPSE = 57;
SPELL_SPIRIT_LINK = 58;
SPELL_DEATH_SCREAM = 59;
SPELL_SNIPE_DEAD = 60;
SPELL_MULTISHOT = 61;
SPELL_IMBUE_ARROW = 62;
SPELL_ABILITY_MAGIC_BOND = 63;
SPELL_ABILITY_MELT_ARTIFACT = 64;
SPELL_ABILITY_COUNTERSPELL = 65;
SPELL_ABILITY_UNSUMMON = 66;
SPELL_ABILITY_DARK_RITUAL = 67;
SPELL_SKILL_OFFENCE1 = 68;
SPELL_SKILL_OFFENCE2 = 69;
SPELL_SKILL_OFFENCE3 = 70;
SPELL_SKILL_ARCHERY = 71;
SPELL_SKILL_FRENZY = 72;
SPELL_SKILL_DEFENCE1 = 73;
SPELL_SKILL_DEFENCE2 = 74;
SPELL_SKILL_DEFENCE3 = 75;
SPELL_SKILL_PROTECTION = 76;
SPELL_SKILL_EVASION = 77;
SPELL_SKILL_TOUGHNESS = 78;
SPELL_SKILL_LUCK1 = 79;
SPELL_SKILL_LUCK2 = 80;
SPELL_SKILL_LUCK3 = 81;
SPELL_SKILL_RESISTANCE = 82;
SPELL_SKILL_LUCKY_STRIKE = 83;
SPELL_SKILL_LEADERSHIP1 = 84;
SPELL_SKILL_LEADERSHIP2 = 85;
SPELL_SKILL_LEADERSHIP3 = 86;
SPELL_SKILL_WAR_MACHINES1 = 87;
SPELL_SKILL_WAR_MACHINES2 = 88;
SPELL_SKILL_WAR_MACHINES3 = 89;
SPELL_SKILL_FIRST_AID = 90;
SPELL_SKILL_BALLISTA = 91;
SPELL_SKILL_CATAPULT = 92;
SPELL_SKILL_DEMONIC_FIRE = 93;
SPELL_SKILL_ELVEN_VOLLEY = 94;
SPELL_SKILL_MATRON_SALVO = 95;
SPELL_SKILL_ANCIENT_SMITHY = 96;
SPELL_SKILL_FIRE_PROTECTION = 97;
SPELL_SPEC_JOUSTER = 98;
SPELL_SPEC_PEASANTS = 99;
SPELL_SPEC_ARCHERS = 100;
SPELL_SPEC_FOOTMEN = 101;
SPELL_SPEC_GRIFFINS = 102;
SPELL_SPEC_ARTILLERYMAN = 103;
SPELL_SPEC_FURIOUS = 104;
SPELL_SPEC_BOMBARDIER = 105;
SPELL_SPEC_IMPREGNABLE = 106;
SPELL_SPEC_FLAGBEARER_OF_DARKNESS = 107;
SPELL_SPEC_HOUNDS = 108;
SPELL_SPEC_SUCCUBUSES = 109;
SPELL_SPEC_BLADE_MASTER = 110;
SPELL_SPEC_ELVES = 111;
SPELL_SPEC_UNICORNS = 112;
SPELL_SPEC_ELVEN_FURY = 113;
SPELL_SPEC_FOREST_GUARDIAN = 114;
SPELL_SPEC_ZOMBIES = 115;
SPELL_SPEC_VAMPIRES = 116;
SPELL_SPEC_EMPIRIC = 117;
SPELL_SPEC_SOULHUNTER = 118;
```

Other IDs

```
SPELL_SPEC_MASTER_OF_ELEMENTS = 119;
SPELL_SPEC_GREMLINS = 120;
SPELL_SPEC_GOLEMS = 121;
SPELL_SPEC_MAGES = 122;
SPELL_SPEC_PRUDENT = 123;
SPELL_SPEC_EVASIVE = 124;
SPELL_SPEC_RIDERS = 125;
SPELL_SPEC_MATRON_SALVO = 126;
SPELL_SPEC_SAVAGE = 127;
SPELL_SPEC_WITCHES = 128;
SPELL_SPEC_MINOTAURS = 129;
SPELL_TOWN_OFFENCE_P1 = 130;
SPELL_TOWN_OFFENCE_M1 = 131;
SPELL_TOWN_DEFENCE_P1 = 132;
SPELL_TOWN_DEFENCE_M1 = 133;
SPELL_TOWN_OFFENCE_DEFENCE_P1 = 134;
SPELL_TOWN_OFFENCE_DEFENCE_M1 = 135;
SPELL_TOWN_OFFENCE_P2 = 136;
SPELL_TOWN_OFFENCE_M2 = 137;
SPELL_TOWN_DEFENCE_P2 = 138;
SPELL_TOWN_DEFENCE_M2 = 139;
SPELL_TOWN_OFFENCE_DEFENCE_P2 = 140;
SPELL_TOWN_OFFENCE_DEFENCE_M2 = 141;
SPELL_TOWN_LUCK_P1 = 142;
SPELL_TOWN_LUCK_M1 = 143;
SPELL_TOWN_MORALE_P1 = 144;
SPELL_TOWN_MORALE_M1 = 145;
SPELL_TOWN_ELVEN_CAPITAL = 146;
SPELL_TOWN_WALLS = 147;
SPELL_TOWN_TOWERS = 148;
SPELL_TOWN_UNHOLY_TEMPLE = 149;
SPELL_TOWN_DARK_GUARDIAN = 150;
SPELL_TOWN_SPARKLING_FOUNTAIN = 151;
SPELL_TOWN_DIETY_OF_FIRE = 152;
SPELL_TOWN_INFERNAL_LOOM = 153;
SPELL_ABILITY_BATTLE_DIVE = 154;
SPELL_ABILITY_BATTLE_DIVE_FINISH = 155;
SPELL_ABILITY_LAY_HANDS = 156;
SPELL_ABILITY_RESURRECT_ALLIES = 157;
SPELL_ABILITY_SCATTER_SHOT = 158;
SPELL_ABILITY_GATING = 159;
SPELL_ABILITY_FEAR = 160;
SPELL_ABILITY_SUMMON_BALOR = 161;
SPELL_ABILITY_EXPLOSION = 162;
SPELL_ABILITY_EXPLOSION_EFFECT = 163;
SPELL_ABILITY_CHAIN_SHOT_END_EFFECT = 164;
SPELL_ABILITY_MANA_DESTROY = 165;
SPELL_ABILITY_MANA_STEAL = 166;
SPELL_ABILITY_LIFE_DRAIN = 167;
SPELL_ABILITY_MANA_DRAIN = 168;
SPELL_ABILITY_DEATH_CLOUD = 169;
SPELL_ABILITY_HARM_TOUCH = 170;
SPELL_ABILITY_MANA_FEED = 171;
SPELL_ABILITY_ENTANGLING_ROOTS = 172;
SPELL_ABILITY_REPAIR = 173;
SPELL_ABILITY_RANDOM_CAST_DARK = 174;
SPELL_ABILITY_RANDOM_CAST_DARK_LIGHT = 175;
SPELL_ABILITY_DASH = 176;
SPELL_ABILITY_DASH_EFFECT = 177;
SPELL_REMOTE_CONTROL = 178;
SPELL_EFFECT_ARMOR_CRUSHING = 179;
SPELL_ABILITY_POISONOUS_ATTACK = 180;
SPELL_LIZARD_BITE_HIT = 181;
SPELL_EFFECT_REGENRATION = 182;
SPELL_EFFECT_REBIRTH = 183;
SPELL_ABILITY_FROST_BREATH = 184;
SPELL_EFFECT_BAD_LUCK = 185;
SPELL_EFFECT_GOOD_LUCK = 186;
SPELL_EFFECT_BAD_MORALE = 187;
SPELL_EFFECT_GOOD_MORALE = 188;
SPELL_EFFECT_FIRST_AID_TENT_HEAL = 189;
SPELL_EFFECT_CLERIC_HIT = 190;
SPELL_EFFECT_COMBAT_HIT_00 = 191;
SPELL_EFFECT_COMBAT_HIT_01 = 192;
SPELL_EFFECT_COMBAT_HIT_02 = 193;
SPELL_EFFECT_WAR_MACHINE_HIT = 194;
SPELL_EFFECT_FIRST_AID_HIT = 195;
SPELL_EFFECT_FIRE_HIT = 196;
SPELL_EFFECT_BASH_HIT = 197;
SPELL_EFFECT_SUN_FIRE = 198;
SPELL_EFFECT_SOIL_BURN = 199;
SPELL_EFFECT_CATAPULT_CHARGE_EXPLOSION = 200;
SPELL_EFFECT_FROZEN = 201;
SPELL_EFFECT_FIRE_DAMAGE = 202;
SPELL_EFFECT_LAND_MINE_EXPLOSION = 203;
SPELL_EFFECT_PHANTOM_OUT = 204;
SPELL_EFFECT_FIRE_SHIELD = 205;
SPELL_EFFECT_DIMENSION_DOOR_END = 206;
SPELL_SKILL_CHILLING_BONES = 207;
SPELL_DEBUG_TELEPORT = 208;
SPELL_ABILITY_ENRAGED = 209;
SPELL_MASS_CURSE = 210;
SPELL_MASS_DISRUPTING_RAY = 211;
SPELL_MASS_SLOW = 212;
SPELL_MASS_FORGETFULNESS = 213;
SPELL_MASS_PLAGUE = 214;
SPELL_MASS_WEAKNESS = 215;
SPELL_MASS_BLESS = 216;
SPELL_MASS_DISPEL = 217;
SPELL_MASS_STONESKIN = 218;
SPELL_MASS_DEFLECT_ARROWS = 219;
SPELL_MASS_BLOODLUST = 220;
SPELL_MASS_HASTE = 221;
SPELL_ABILITY_CALL_LIGHTNING = 222;
SPELL_EMPOWERED_MAGIC_ARROW = 223;
SPELL_EMPOWERED_MAGIC_FIST = 224;
SPELL_EMPOWERED_LIGHTNING_BOLT = 225;
SPELL_EMPOWERED_ICE_BOLT = 226;
SPELL_EMPOWERED_FIREBALL = 227;
SPELL_EMPOWERED_FROST_RING = 228;
SPELL_EMPOWERED_CHAIN_LIGHTNING = 229;
SPELL_EMPOWERED_METEOR_SHOWER = 230;
SPELL_EMPOWERED_IMPLOSION = 231;
SPELL_EMPOWERED_ARMAGEDDON = 232;
SPELL_EMPOWERED_STONE_SPIKES = 233;
SPELL_SUMMON_CREATURES = 234;
SPELL_CONJURE_PHOENIX = 235;
SPELL_FIREWALL = 236;
SPELL_STONE_SPIKES = 237;
SPELL_UBER_CHAIN_LIGHTNING = 238;
```

```
SPELL_DEMON_SOVEREIGN_FX = 239;
SPELL_SORROW = 277;
SPELL_VAMPIRISM = 278;
SPELL_DEEP_FREEZE = 279;
SPELL_REGENERATION = 280;
SPELL_DIVINE_VENGEANCE = 281;
SPELL_ARCANE_CRYSTAL = 282;
SPELL_SUMMON_HIVE = 283;
SPELL_BLADE_BARRIER = 284;
SPELL_WARCRY_RALLING_CRY = 290;
SPELL_WARCRY_CALL_OF_BLOOD = 291;
SPELL_WARCRY_WORD_OF_THE_CHIEF = 292;
SPELL_WARCRY_FEAR_MY_ROAR = 293;
SPELL_WARCRY_BATTLECRY = 294;
SPELL_WARCRY_SHOUT_OF_MANY = 295;
```

Visit Celestial Heavens online

www.celestialheavens.com