**Reborn: New Scripting Commands Documentation**

ScriptThread -> Listener -> Class  
  
netname ( *Entity* player )  
{

Gets player's name and returns it as string.  
Example:

Code:

|  |
| --- |
| local.player\_name = netname $player[1]  or  local.player\_name = netname( $player[1] ) |

}

ScriptThread -> Listener -> Class  
  
getip ( *Entity* player )  
{

Gets player's ip address with port and returns it as string.  
Example:

Code:

|  |
| --- |
| local.player\_ip = getip $player[1]  or  local.player\_ip = getip( $player[1] ) |

Result:

Code:

|  |
| --- |
| 127.0.0.1:12203 |

}

ScriptThread -> Listener -> Class  
  
getping ( *Entity* player )  
{

Gets player's ping and returns it as integer.  
Example:

Code:

|  |
| --- |
| local.player\_ping = getping $player[1]  or  local.player\_ping = getping( $player[1] ) |

}

ScriptThread -> Listener -> Class  
  
getclientnum ( *Entity* player )  
{

Gets player's client number and returns it as integer.  
Example:

Code:

|  |
| --- |
| local.player\_clnum = getclientnum $player[1]  or  local.player\_clnum = getclientnum( $player[1] ) |

}

ScriptThread -> Listener -> Class  
  
getentity ( *Integer* entnum)  
{

Returns entity with given entity number  
Example:

Code:

|  |
| --- |
| local.entity = getentity 0  or  local.entity = getentity( 0 ) |

Entities with entity number between 0 and **sv\_maxclient** are reserved for players and thus getentity( 0 ) is equal to $player[0]

}  
  
  
  
ScriptThread -> Listener -> Class  
  
stuffsrv ( *String* s )  
{

Sends command to server console.  
Example:

Code:

|  |
| --- |
| stuffsrv "restart"  stuffsrv( "restart" )  or  stuffsrv "map dm/mohdm1"  stuffsrv( "map dm/mohdm1" ) |

Result:

Code:

|  |
| --- |
| Server will restart  or  Server will change map to dm/mohdm1 |

}  
  
  
ScriptThread -> Listener -> Class  
  
ihuddraw\_align ( *Entity* player, *Integer* index, *String* h\_align, *String* v\_align )  
{

Sets the alignment of a huddraw element for individual player.  
Where:

Code:

|  |
| --- |
| h\_align = "left", "center", "right"  v\_align = "bottom", "center", "top"  index = index of huddraw element to be set  player = entity of player that will have his huddraw element set |

Example:

Code:

|  |
| --- |
| ihuddraw\_align $player[1] 15 right top  or  ihuddraw\_align( $player[1] 15 "right" "top" ) |

}  
  
  
  
ScriptThread -> Listener -> Class  
  
ihuddraw\_alpha ( *Entity* player, *Integer* index, *Float* alpha )  
{

Sets the alpha of a huddraw element for individual player.  
Where:

Code:

|  |
| --- |
| index = index of huddraw element to be set  player = entity of player that will have his huddraw element set |

Example:

Code:

|  |
| --- |
| ihuddraw\_alpha $player[1] 15 1  or  ihuddraw\_alpha( $player[1] 15 1 ) |

}  
  
  
  
ScriptThread -> Listener -> Class  
  
ihuddraw\_color ( *Entity* player, *Integer* index, *Float* red, *Float* green, *Float* blue )  
{

Sets the color for a huddraw element for individual player.  
Where:

Code:

|  |
| --- |
| index = index of huddraw element to be set  player = entity of player that will have his huddraw element set |

Example:

Code:

|  |
| --- |
| ihuddraw\_color $player[1] 15 1 1 1  or  ihuddraw\_color( $player[1] 15 1 1 1 ) |

}  
  
  
  
ScriptThread -> Listener -> Class  
  
ihuddraw\_font ( *Entity* player, *Integer* index, *String* fontname )  
{

Sets the font to use for a huddraw element, for individual player.  
Where:

Code:

|  |
| --- |
| index = index of huddraw element to be set  player = entity of player that will have his huddraw element set |

Example:

Code:

|  |
| --- |
| ihuddraw\_font $player[1] 15 "verdana-14"  or  ihuddraw\_font( $player[1] 15 "verdana-14" ) |

}

ScriptThread -> Listener -> Class  
  
ihuddraw\_rect ( *Entity* player, *Integer* index, *Integer* x, *Integer* y, *Integer* width, *Integer* height )  
{

Specifies the position of the upper left corner and size of a huddraw element for individual player  
Where:

Code:

|  |
| --- |
| index = index of huddraw element to be set  player = entity of player that will have his huddraw element set |

Example:

Code:

|  |
| --- |
| ihuddraw\_rect $player[1] 15 -140 65 0 0  or  ihuddraw\_rect( $player[1] 15 -140 65 0 0 ) |

}  
  
  
  
ScriptThread -> Listener -> Class  
  
ihuddraw\_shader ( *Entity* player, *Integer* index, *String* shader )  
{

Sets the shader to use for a particular huddraw element for individual player  
Where:

Code:

|  |
| --- |
| index = index of huddraw element to be set  player = entity of player that will have his huddraw element set |

Example:

Code:

|  |
| --- |
| ihuddraw\_shader $player[1] 15 "textures/hud/axis"  or  ihuddraw\_shader( $player[1] 15 "textures/hud/axis" ) |

}  
  
  
  
ScriptThread -> Listener -> Class  
  
ihuddraw\_string ( *Entity* player, *Integer* index, *String* string )  
{

Sets a string to be displayed. Clears the shader value of huddraw element for individual player  
Where:

Code:

|  |
| --- |
| index = index of huddraw element to be set  player = entity of player that will have his huddraw element set |

Example:

Code:

|  |
| --- |
| ihuddraw\_string $player[1] 15 "I luv Reborn 1.12 Patch!"  or  ihuddraw\_string( $player[1] 15 "I luv Reborn 1.12 Patch!" ) |

}  
  
  
  
ScriptThread -> Listener -> Class  
  
ihuddraw\_virtualsize ( *Entity* player, *Integer* index, *Integer* virtual )  
{

Sets if the huddraw element (for individual player) should use virutal screen resolution for positioning and size.  
Where:

Code:

|  |
| --- |
| index = index of huddraw element to be set  player = entity of player that will have his huddraw element set |

Example:

Code:

|  |
| --- |
| ihuddraw\_virtualsize $player[1] 15 1  or  ihuddraw\_virtualsize( $player[1] 15 1 ) |

}  
  
  
  
ScriptThread -> Listener -> Class  
  
fopen ( *String* filename, *String* accessType )  
{

Opens file.  
Example:

Code:

|  |
| --- |
| local.file = fopen "main/config.txt" "a+"  or  local.file = fopen("main/config.txt" "a+") |

Result:

Code:

|  |
| --- |
| Command returns file handle that is needed for identification and further operations on this file. |

You can open only 32 files at once. This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - fopen](http://www.cplusplus.com/reference/clibrary/cstdio/fopen/)  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
fclose ( *Integer* filehandle )  
{

Closes file.  
Example:

Code:

|  |
| --- |
| local.return = fclose local.file  or  local.return = fclose(local.file) |

Result:

Code:

|  |
| --- |
| If file is successfully closed, a zero value is returned.  On failure, EOF is returned. |

You can open only 32 files at once. This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - fclose](http://www.cplusplus.com/reference/clibrary/cstdio/fclose/)  
}  
ScriptThread -> Listener -> Class  
  
feof ( *Integer* filehandle )  
{

Checks for end of file.  
Example:

Code:

|  |
| --- |
| local.return = feof local.file  or  local.return = feof(local.file) |

Result:

Code:

|  |
| --- |
| A non-zero value is returned in the case that the End-of-File indicator associated with the file is set.  Otherwise, a zero value is returned. |

You can open only 32 files at once. This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - feof](http://www.cplusplus.com/reference/clibrary/cstdio/feof/)  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
fseek ( *Integer* filehandle, *Integer* offset, *Integer* origin )  
{

Sets the position indicator associated with the file to a new position defined by adding *offset* to a reference position specified by *origin*.  
Example:

Code:

|  |
| --- |
| local.return = fseek local.file 154 0  or  local.return = fseek(local.file 154 0) |

Where:

Code:

|  |
| --- |
| origin:   * 0 = SEEK\_SET * 1 = SEEK\_CUR * 2 = SEEK\_END |

Result:

Code:

|  |
| --- |
| If successful, the function returns a zero value.  Otherwise, it returns nonzero value. |

You can open only 32 files at once. This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - fseek](http://www.cplusplus.com/reference/clibrary/cstdio/fseek/)  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
ftell ( *Integer* filehandle )  
{

Returns the current value of the position indicator of the file.  
Example:

Code:

|  |
| --- |
| local.return = ftell local.file  or  local.return = ftell(local.file) |

Result:

Code:

|  |
| --- |
| On success, the current value of the position indicator is returned.  If an error occurs, -1 is returned. |

You can open only 32 files at once. This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - ftell](http://www.cplusplus.com/reference/clibrary/cstdio/ftell/)  
}

ScriptThread -> Listener -> Class  
  
frewind ( *Integer* filehandle )  
{

Sets the position indicator associated with file to the beginning of the file.  
Example:

Code:

|  |
| --- |
| local.return = frewind local.file  or  local.return = frewind(local.file) |

Result:

Code:

|  |
| --- |
| *none* |

You can open only 32 files at once. This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - rewind](http://www.cplusplus.com/reference/clibrary/cstdio/rewind/)  
}  
  
  
ScriptThread -> Listener -> Class  
  
fputc ( *Integer* filehandle, *String* character )  
{

Writes a character to the file and advances the position indicator.  
Example:

Code:

|  |
| --- |
| local.return = fputc local.file "a"  or  local.return = fputc(local.file "a") |

Result:

Code:

|  |
| --- |
| If there are no errors, the same character that has been written is returned.  If an error occurs, EOF is returned and the error indicator is set.  You have to cast returned value to char.  If you pass longer string, only first character will be written to file. |

You can open only 32 files at once. This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - fputc](http://www.cplusplus.com/reference/clibrary/cstdio/fputc/)  
}  
ScriptThread -> Listener -> Class  
  
fputs ( *Integer* filehandle, *String* text )  
{

Writes text to the file and advances the position indicator.  
Example:

Code:

|  |
| --- |
| local.return = fputs local.file "This is example"  or  local.return = fputs(local.file "This is example") |

Result:

Code:

|  |
| --- |
| On success, a non-negative value is returned.  On error, the function returns EOF. |

You can open only 32 files at once. This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - fputs](http://www.cplusplus.com/reference/clibrary/cstdio/fputs/)  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
fgetc ( *Integer* filehandle )  
{

Reads single character from file.  
Example:

Code:

|  |
| --- |
| local.char = fgetc local.file  or  local.char = fgetc(local.file) |

Result:

Code:

|  |
| --- |
| The character read is returned as an int value.  You need to cast it to char if you want to use it in string. |

You can open only 32 files at once. This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - fgetc](http://www.cplusplus.com/reference/clibrary/cstdio/fgetc/)  
}  
ScriptThread -> Listener -> Class  
  
fgets ( *Integer* filehandle, *Integer* maxbuffsize )  
{

Reads string line from file.  
Where:

Code:

|  |
| --- |
| maxbuffsize - specifies maximum buffer size that will be allocated to store the string in memory. |

Example:

Code:

|  |
| --- |
| local.text = fgets local.file 256  or  local.text = fgets(local.file 256) |

Result:

Code:

|  |
| --- |
| If the End-of-File is encountered and no characters have been read 0 (null) is returned.  If an error occurs 0 (null) is returned.  If a memory allocation error occurs, -1 is returned. |

You can open only 32 files at once. This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - fgets](http://www.cplusplus.com/reference/clibrary/cstdio/fgets/)  
}

ScriptThread -> Listener -> Class  
  
ferror ( *Integer* filehandle )  
{

Checks if the error indicator associated with file is set, returning a value different from zero if it is.  
  
Example:

Code:

|  |
| --- |
| local.ret = ferror local.file  or  local.ret = ferror(local.file) |

Result:

Code:

|  |
| --- |
| If the error indicator associated with the file was set, the function returns a nonzero value.  Otherwise, it returns a zero value. |

You can open only 32 files at once. This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - ferror](http://www.cplusplus.com/reference/clibrary/cstdio/ferror/)  
}

ScriptThread -> Listener -> Class  
  
fflush ( *Integer* filehandle )  
{

If the given file was open for writing and the last i/o operation was an output operation, any unwritten data in the output buffer is written to the file.  
  
The file remains open after this command.  
Example:

Code:

|  |
| --- |
| local.ret = fflush local.file  or  local.ret = fflush(local.file) |

Result:

Code:

|  |
| --- |
| A zero value indicates success.  If an error occurs, EOF is returned and the error indicator is set. |

You can open only 32 files at once. This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - fflush](http://www.cplusplus.com/reference/clibrary/cstdio/fflush/)  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
fexists ( *String* filename )  
{

Checks if file with given filename exists.  
  
Example:

Code:

|  |
| --- |
| local.ret = fexists "folder/folder2/file.txt"  or  local.ret = fexists("folder/folder2/file.txt") |

Result:

Code:

|  |
| --- |
| If file exists, function returns 1. Otherwise it returns 0. |

You can open only 32 files at once.  
}  
ScriptThread -> Listener -> Class  
  
freadall ( *Integer* filehandle )  
{

Reads whole file into a string at once. File has to be opened in binary mode (rb, rb+)  
  
Example:

Code:

|  |
| --- |
| local.content = freadall local.file  or  local.content = freadall(local.file) |

Result:

Code:

|  |
| --- |
| Function returns file content as string. |

Don't read binary files with this function because it may cause memory leaks.  
  
You can open only 32 files at once.  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
fsaveall ( *Integer* filehandle, *String* content )  
{

Writes string content to file at once. File has to be opened in binary mode (wb, wb+, ab). The content will start to be saved from the current file position.  
  
Example:

Code:

|  |
| --- |
| local.ret = fsaveall local.file local.content |

Result:

Code:

|  |
| --- |
| Function returns number of character written to file or -1 if content is NULL. |

You can open only 32 files at once.  
}  
  
  
ScriptThread -> Listener -> Class  
  
fremove ( *String* filename )  
{

Removes the file with given filename.  
  
Example:

Code:

|  |
| --- |
| local.ret = fremove local.filename |

Result:

Code:

|  |
| --- |
| If the file is successfully deleted, a zero value is returned.  On failure, a nonzero value is returned and the errno variable is set to the corresponding error code. |

You can open only 32 files at once. This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - remove](http://www.cplusplus.com/reference/clibrary/cstdio/remove/)  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
frename ( *String* oldname, *String* newname )  
{

Renames the file with given filename.  
  
Example:

Code:

|  |
| --- |
| local.ret = frename local.oldname local.newname |

Result:

Code:

|  |
| --- |
| If the file is successfully renamed, a zero value is returned.  On failure, a nonzero value is returned and the errno variable is set to the corresponding error code. |

You can open only 32 files at once. This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - rename](http://www.cplusplus.com/reference/clibrary/cstdio/rename/)  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
fcopy ( *String* filename, *String* copyname )  
{

Creates a copy of file.  
  
Example:

Code:

|  |
| --- |
| local.ret = fcopy local.filename local.copyname |

Result:

Code:

|  |
| --- |
| If the file is successfully copied, a zero value is returned.  When function fails to open original file, a -1 value is returned.  When function fails to create a second file, a -2 value is returned.  When function fails during data copy process, a -3 value is returned. |

You can open only 32 files at once.  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
freadpak ( *String* filename )  
{

Reads file located inside .pk3 file in text mode and returns it's content as string.  
  
Example:

Code:

|  |
| --- |
| local.content = freadpak local.filename |

Result:

Code:

|  |
| --- |
| If the file is successfully read, function returns a string with it's content.  When function fails to find, open or read a file from .pk3, a -1 value is returned. |

You can open only 32 files at once.  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
flist ( *String* path, *String* extension, *Integer* scanSubDirectories )  
{

Returns a list (array) of files with given extension located in given path. Function handles .pk3 folder structure and normal system directories. When scanSubDirectories equals 1, function will include subdirectories located under directory path.  
  
Extension needs to have "." (dot) included. Otherwise it will act as filter.  
  
Example:

Code:

|  |
| --- |
| local.list = flist local.path local.extension local.scanSubDirectories |

Result:

Code:

|  |
| --- |
| List with filenames and their paths found. |

You can open only 32 files at once.  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
cos ( *Float* x )  
{

Returns the cosine of an angle of x radians.  
  
Example:

Code:

|  |
| --- |
| local.result = cos local.x |

Result:

Code:

|  |
| --- |
| Cosine of x. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - cos](http://www.cplusplus.com/reference/clibrary/cmath/cos/)  
}

ScriptThread -> Listener -> Class  
  
sin ( *Float* x )  
{

Returns the sine of an angle of x radians.  
  
Example:

Code:

|  |
| --- |
| local.result = cos local.x |

Result:

Code:

|  |
| --- |
| Cosine of x. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - cos](http://www.cplusplus.com/reference/clibrary/cmath/cos/)  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
tan ( *Float* x )  
{

Returns the tangent of an angle of x radians.  
  
Example:

Code:

|  |
| --- |
| local.result = tan local.x |

Result:

Code:

|  |
| --- |
| Tangent of x. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - tan](http://www.cplusplus.com/reference/clibrary/cmath/tan/)  
}

ScriptThread -> Listener -> Class  
  
acos ( *Float* x )  
{

Returns the principal value of the arc cosine of x, expressed in radians.  
  
Example:

Code:

|  |
| --- |
| local.result = acos local.x |

Result:

Code:

|  |
| --- |
| Floating point value in the interval [-1,+1]. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - acos](http://www.cplusplus.com/reference/clibrary/cmath/acos/)  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
asin ( *Float* x )  
{

Returns the principal value of the arc sine of x, expressed in radians.  
  
Example:

Code:

|  |
| --- |
| local.result = asin local.x |

Result:

Code:

|  |
| --- |
| Floating point value in the interval [-1,+1]. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - asin](http://www.cplusplus.com/reference/clibrary/cmath/asin/)  
}

ScriptThread -> Listener -> Class  
  
atan ( *Float* x )  
{

Returns the principal value of the arc tangent of x, expressed in radians.  
  
Notice that because of the sign ambiguity, a function cannot determine with certainty in which quadrant the angle falls only by its tangent value. You can use atan2 if you need to determine the quadrant.  
  
Example:

Code:

|  |
| --- |
| local.result = atan local.x |

Result:

Code:

|  |
| --- |
| Principal arc tangent of x, in the interval [-pi/2,+pi/2] radians. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - atan](http://www.cplusplus.com/reference/clibrary/cmath/atan/)  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
atan2 ( *Float* y, [I]Float[I] x )  
{

Returns the principal value of the arc tangent of y/x, expressed in radians.  
  
To compute the value, the function uses the sign of both arguments to determine the quadrant.  
  
Example:

Code:

|  |
| --- |
| local.result = atan2 local.y local.x |

Result:

Code:

|  |
| --- |
| Principal arc tangent of y/x, in the interval [-pi,+pi] radians. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - atan2](http://www.cplusplus.com/reference/clibrary/cmath/atan2/)  
}  
ScriptThread -> Listener -> Class  
  
cosh ( *Float* x )  
{

Returns the hyperbolic cosine of x.  
  
Example:

Code:

|  |
| --- |
| local.result = cosh local.x |

Result:

Code:

|  |
| --- |
| Hyperbolic cosine of x. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - cosh](http://www.cplusplus.com/reference/clibrary/cmath/cosh/)  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
sinh ( *Float* x )  
{

Returns the hyperbolic sine of x.  
  
Example:

Code:

|  |
| --- |
| local.result = sinh local.x |

Result:

Code:

|  |
| --- |
| Hyperbolic sine of x. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - sinh](http://www.cplusplus.com/reference/clibrary/cmath/sinh/)  
}

ScriptThread -> Listener -> Class  
  
tanh ( *Float* x )  
{

Returns the hyperbolic tangent of x.  
  
Example:

Code:

|  |
| --- |
| local.result = tanh local.x |

Result:

Code:

|  |
| --- |
| Hyperbolic tangent of x. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - tanh](http://www.cplusplus.com/reference/clibrary/cmath/tanh/)  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
exp ( *Float* x )  
{

Returns the base-e exponential function of x, which is the e number raised to the power x.  
  
Example:

Code:

|  |
| --- |
| local.result = exp local.x |

Result:

Code:

|  |
| --- |
| Exponential value of x. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - exp](http://www.cplusplus.com/reference/clibrary/cmath/exp/)  
}

ScriptThread -> Listener -> Class  
  
frexp ( *Float* x )  
{

Breaks the floating point number x into its binary significand (a floating point value between 0.5(included) and 1.0(excluded)) and an integral exponent for 2, such that:  
  
x = significand \* 2exponent  
If x is zero, both parts (significand and exponent) are zero.  
  
Example:

Code:

|  |
| --- |
| local.result = frexp local.x |

Result:

Code:

|  |
| --- |
| local.result["significand"] - significand part  local.result["exponent"] - exponent part |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - frexp](http://www.cplusplus.com/reference/clibrary/cmath/frexp/)  
}  
  
  
ScriptThread -> Listener -> Class  
  
ldexp ( *Float* x, *Integer* exponent )  
{

Returns the resulting floating point value from multiplying x (the significand) by 2 raised to the power of exp (the exponent).  
  
Example:

Code:

|  |
| --- |
| local.result = ldexp local.x local.exponent |

Result:

Code:

|  |
| --- |
| The function returns float number:  x \* 2exp |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - ldexp](http://www.cplusplus.com/reference/clibrary/cmath/ldexp/)  
}  
ScriptThread -> Listener -> Class  
  
log ( *Float* x )  
{

Returns the natural logarithm of x.  
  
The natural logarithm is the base-e logarithm, the inverse of the natural exponential function (exp). For base-10 logarithms, a specific function log10 exists.  
  
Example:

Code:

|  |
| --- |
| local.result = log local.x |

Result:

Code:

|  |
| --- |
| Natural logarithm of x. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - log](http://www.cplusplus.com/reference/clibrary/cmath/log/)  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
log10 ( *Float* x )  
{

Returns the common (base-10) logarithm of x.  
  
Example:

Code:

|  |
| --- |
| local.result = log10 local.x |

Result:

Code:

|  |
| --- |
| Common logarithm of x, for values of x greater than zero. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - log10](http://www.cplusplus.com/reference/clibrary/cmath/log10/)  
}

ScriptThread -> Listener -> Class  
  
modf ( *Float* x )  
{

Breaks x into two parts: the integer part and the fractional part.  
  
Example:

Code:

|  |
| --- |
| local.result = modf local.x |

Result:

Code:

|  |
| --- |
| local.result["intpart"] - integer part  local.result["fractional"] - fractional part |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - modf](http://www.cplusplus.com/reference/clibrary/cmath/modf/)  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
pow ( *Float* x, *Integer* exponent )  
{

Returns base raised to the power exponent:  
  
baseexponent  
  
Example:

Code:

|  |
| --- |
| local.result = pow local.x local.exponent |

Result:

Code:

|  |
| --- |
| The result of raising base to the power exponent. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - pow](http://www.cplusplus.com/reference/clibrary/cmath/pow/)  
}

ScriptThread -> Listener -> Class  
  
sqrt ( *Float* x )  
{

Returns the square root of x.  
  
Example:

Code:

|  |
| --- |
| local.result = sqrt local.x |

Result:

Code:

|  |
| --- |
| Square root of x. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - sqrt](http://www.cplusplus.com/reference/clibrary/cmath/sqrt/)  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
ceil ( *Float* x )  
{

Returns the smallest integral value that is not less than x.  
  
Example:

Code:

|  |
| --- |
| local.result = ceil local.x |

Result:

Code:

|  |
| --- |
| The smallest integral value not less than x. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - ceil](http://www.cplusplus.com/reference/clibrary/cmath/ceil/)  
}

ScriptThread -> Listener -> Class  
  
floor ( *Float* x )  
{

Returns the largest integral value that is not greater than x.  
  
Example:

Code:

|  |
| --- |
| local.result = floor local.x |

Result:

Code:

|  |
| --- |
| The largest integral value not greater than x. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - floor](http://www.cplusplus.com/reference/clibrary/cmath/floor/)  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
fmod ( *Float* numerator, *Float* denominator )  
{

Returns the floating-point remainder of numerator/denominator.  
  
The remainder of a division operation is the result of subtracting the integral quotient multiplied by the denominator from the numerator:  
  
remainder = numerator - quotient \* denominator  
  
Example:

Code:

|  |
| --- |
| local.result = fmod local.numerator local.denominator |

Result:

Code:

|  |
| --- |
| The remainder of dividing the arguments. |

This command works exactly like ANSI C function. For further documentation, please visit: [ANSI C - fmod](http://www.cplusplus.com/reference/clibrary/cmath/fmod/)  
}  
  
  
ScriptThread -> Listener -> Class  
  
gettime ( *Integer* zero )  
{

Gets current time in format: H:M:S  
Example:

Code:

|  |
| --- |
| local.time = gettime 0  or  local.time = gettime(0) |

Result:

Code:

|  |
| --- |
| String with current time. |

}  
  
  
  
ScriptThread -> Listener -> Class  
  
gettimezone ( *Integer* zero )  
{

Gets current time zone.  
Example:

Code:

|  |
| --- |
| local.timezone = gettimezone 0  or  local.timezone = gettimezone(0) |

Result:

Code:

|  |
| --- |
| Integer value that represents current time zone.  eg. 2 = GMT +2 |

}

ScriptThread -> Listener -> Class  
  
getdate ( *String* format )  
{

Gets current date in format given as parameter.  
Example:

Code:

|  |
| --- |
| local.date = getdate "%D"  or  local.date = getdate("%D") |

Result:

Code:

|  |
| --- |
| String with current date.  04/06/18 |

Formatting options:  
<http://www.cplusplus.com/reference/ctime/strftime/>  
  
Max format length is 512 characters.

}

ScriptThread -> Listener -> Class  
  
registerev ( *String* eventname, *String* scriptname )  
{

Registers script callback handler for given event type.  
Example:

Code:

|  |
| --- |
| local.result = registerev "connected" global/eventhandlers.scr::connected  or  local.result = registerev("connected" global/eventhandlers.scr::connected) |

Result:

Code:

|  |
| --- |
| When given even type will occur, EventSystem engine will execute given script.  local.result can have one of the following values:  0 = Registering event callback handler was successful  1 = Event callback handler is already registered for given event  2 = Memory allocation error |

Please see EventSystem documentation for further informations  
  
}

ScriptThread -> Listener -> Class  
  
unregisterev ( *String* eventname )  
{

Unregisters script callback handler for given event type.  
Example:

Code:

|  |
| --- |
| local.result = unregisterev "connected"  or  local.result = unregisterev("connected") |

Result:

Code:

|  |
| --- |
| EventSystem engine will unregister events of given type and won't execute their script callback handlers.  local.result can have one of the following values:  0 = Unregistering event callback handler was successful  1 = Event callback handler is already unregistered |

Please see EventSystem documentation for further informations  
  
}  
  
  
  
ScriptThread -> Listener -> Class  
  
conprintf ( *String* text )  
{

Prints text to a console.  
Example:

Code:

|  |
| --- |
| conprintf "This can be a custom error message from the script"  or  conprintf( "This can be a custom error message from the script" ) |

Result:

Code:

|  |
| --- |
| Text will be printed to the console. |

}  
ScriptThread -> Listener -> Class  
  
md5string ( *String* text )  
{

Generates MD5 checksum of text  
Example:

Code:

|  |
| --- |
| local.checksum = md5string local.text |

Result:

Code:

|  |
| --- |
| MD5 checksum as string. |

}  
  
  
  
ScriptThread -> Listener -> Class  
  
md5file ( *String* filename )  
{

Generates MD5 checksum of file with given filename  
Example:

Code:

|  |
| --- |
| local.checksum = md5file local.filename |

Result:

Code:

|  |
| --- |
| MD5 checksum as string. |

}

ScriptThread -> Listener -> Class  
  
typeof ( *Variable* var )  
{

Gets the type of variable.  
Example:

Code:

|  |
| --- |
| local.type = typeof local.var |

Result:

Code:

|  |
| --- |
| The type of variable returned as string (array, string, vector, listener, ...) |

}  
  
  
  
ScriptThread -> Listener -> Class  
  
traced ( *Vector* start, *Vector* end, [*Integer* pass\_entities], [*Vecotr* mins], [*Vector* maxs], [*Integer* mask])  
{

Performs a ray trace from *start* origin to *end* origin. It takes optional arguments such as entity number to be ignored/skipped by the trace, mins and maxs of trace box and trace mask.  
  
Example:

Code:

|  |
| --- |
| local.trace = traced local.start local.end  or  local.trace = traced local.start local.end local.pass\_entities local.mins  or  local.trace = traced local.start local.end local.pass\_entities local.mins local.maxs local.mask |

Result:

Code:

|  |
| --- |
| Array holding detailed information about trace:  local.trace["allSolid"] - *Integer* : it tells wheter trace was inside of a solid object  local.trace["startSolid"] - *Integer* : it tells wheter trace started in solid object  local.trace["fraction"] - *Float*  local.trace["endPos"] - *Vector* : position where trace finished because it may finish before it reaches end point specified by caller when it hits object with specified mask before it reaches end point  local.trace["surfaceFlags"] - *Integer*  local.trace["shaderNum"] - *Integer*  local.trace["contents"] - *Integer*  local.trace["entityNum"] - *Integer* : entity number that was hit  local.trace["location"] - *Integer*  local.trace["entity"] - *Entity* : entity that was hit by the trace |

Surface Flags:

Code:

|  |
| --- |
| SURF\_NODAMAGE 1  SURF\_SLICK 2  SURF\_SKY 4  SURF\_LADDER 8  SURF\_NOIMPACT 16  SURF\_NOMARKS 32  SURF\_CASTSHADOW 64  SURF\_PAPER 8192  SURF\_WOOD 16384  SURF\_METAL 32768  SURF\_STONE 65536  SURF\_DIRT 131072  SURF\_METALGRILL 262144  SURF\_GRASS 524288  SURF\_MUD 1048576  SURF\_PUDDLE 2097152  SURF\_GLASS 4194304  SURF\_GRAVEL 8388608  SURF\_SAND 16777216  SURF\_FOLIAGE 33554432  SURF\_SNOW 67108864  SURF\_CARPET 134217728  SURF\_BACKSIDE 268435456  SURF\_NODLIGHT 536870912  SURF\_HINT 1073741824 |

Masks:

Code:

|  |
| --- |
| MASK\_SOLID 1  MASK\_COLLISION 637537057  MASK\_PERMANENTMARK 1073741825  MASK\_AUTOCALCLIFE 1073750049  MASK\_EXPLOSION 1074003969  MASK\_TREADMARK 1107372801  MASK\_THIRDPERSON 1107372857  MASK\_FOOTSTEP 1107437825  MASK\_BEAM 1107569409  MASK\_VISIBLE 1107569409  MASK\_VEHICLE 1107569409  MASK\_BULLET 1107569441  MASK\_SHOT 1107569569  MASK\_CROSSHAIRSHADER 1107897089  MASK\_TRACER 1108618017 |

Contents:

Code:

|  |
| --- |
| CONTENTS\_SOLID 1  CONTENTS\_LAVA 8  CONTENTS\_SLIME 16  CONTENTS\_WATER 32  CONTENTS\_FOG 64  CONTENTS\_AREAPORTAL 32768  CONTENTS\_PLAYERCLIP 65536  CONTENTS\_MONSTERCLIP 131072  CONTENTS\_WEAPONCLIP 262144  CONTENTS\_SHOOTABLEONLY 1048576  CONTENTS\_ORIGIN 16777216  CONTENTS\_BODY 33554432  CONTENTS\_CORPSE 67108864  CONTENTS\_DETAIL 134217728  CONTENTS\_STRUCTURAL 268435456  CONTENTS\_TRANSLUCENT 536870912  CONTENTS\_TRIGGER 1073741824  CONTENTS\_NODROP 2147483648 |

}  
  
  
  
ScriptThread -> Listener -> Class  
  
setproperty ( *String* key, *String* value )  
{

Sets property in local storage to given value.  
  
Key and value can't be NULL. If you want to clear the value, you have to set it to empty string.  
Example:

Code:

|  |
| --- |
| local.res = setproperty "my\_mod\_settings" "abcdefgh" |

Result:

Code:

|  |
| --- |
| Returns integer value:  0 - Success  < 0 - Error |

}

ScriptThread -> Listener -> Class  
  
getproperty ( *String* key )  
{

Gets property saved in local storage for given key.  
  
Key can't be NULL.  
Example:

Code:

|  |
| --- |
| local.value = getproperty "my\_mod\_settings" |

Result:

Code:

|  |
| --- |
| Returns value as string or error code as integer. |

}  
  
  
  
Player (player) -> Sentient -> Animate -> Entity -> SimpleEntity -> Listener -> Class  
  
addkills ( *Integer* kills )  
{

Adds number of kills to player. (Can be also negative)  
Example:

Code:

|  |
| --- |
| $player[1] addkills 5  or  $player[1] addkills -5 |

Result:

Code:

|  |
| --- |
| If player had 8 kills, he will have 13 kills  or  If player had 8 kills, he will have 3 kills |

}  
  
Player (player) -> Sentient -> Animate -> Entity -> SimpleEntity -> Listener -> Class  
  
adddeaths ( *Integer* deaths )  
{

Adds number of deaths to player. (Can be also negative)  
Example:

Code:

|  |
| --- |
| $player[1] adddeaths 5  or  $player[1] adddeaths -5 |

Result:

Code:

|  |
| --- |
| If player had 8 deaths, he will have 13 deaths  or  If player had 8 deaths, he will have 3 deaths |

}  
  
  
  
Player (player) -> Sentient -> Animate -> Entity -> SimpleEntity -> Listener -> Class  
  
getkills ( *void* )  
{

Gets number of player's kills and returns it as integer  
Example:

Code:

|  |
| --- |
| local.player\_kills = $player[1] getkills  or  local.player\_kills = $player[1] getkills( ) |

Result:

Code:

|  |
| --- |
| Number of players that this player has killed. |

}  
Player (player) -> Sentient -> Animate -> Entity -> SimpleEntity -> Listener -> Class  
  
getdeaths ( *void* )  
{

Gets number of player's deaths and returns it as integer.  
Example:

Code:

|  |
| --- |
| local.player\_deaths = $player[1] getdeaths  or  local.player\_deaths = $player[1] getdeaths( ) |

Result:

Code:

|  |
| --- |
| Number of player's deaths. |

Before using this function, check game type that is currently running. When g\_gametype is 3 or 4, deaths aren't counted and result of this function will equal to kills amount that player has  
}  
  
  
  
Player (player) -> Sentient -> Animate -> Entity -> SimpleEntity -> Listener -> Class  
  
isadmin ( *void* )  
{

Checks if player is currently logged in as server administrator.  
Example:

Code:

|  |
| --- |
| local.admin = $player[1] isadmin  or  local.admin = $player[1] isadmin( ) |

Result:

Code:

|  |
| --- |
| Returns 1 if player is logged in as administrator, otherwise it returns 0. |

}  
Player (player) -> Sentient -> Animate -> Entity -> SimpleEntity -> Listener -> Class  
  
getconnstate ( *void* )  
{

Gets state of player's connection.  
Example:

Code:

|  |
| --- |
| local.connection\_state = $player[1] getconnstate  or  local.connection\_state = $player[1] getconnstate( ) |

Result:

Code:

|  |
| --- |
| Returns integer value:   * 0 = CS\_FREE - given player slot is free * 1 = CS\_ZOMBIE - given player slot is in zombie state (his data is still kept after he disconnected or lost connection) * 2 = CS\_CONNECTED - player has connected to server, but he's not yet in the game * 3 = CS\_PRIMED - player has passed through authorization checks and finished downloading any missing files * 4 = CS\_ACTIVE - player is in game and can start playing |

}

Player (player) -> Sentient -> Animate -> Entity -> SimpleEntity -> Listener -> Class  
  
getactiveweap ( *Integer* weaponhand )  
{

Gets currently active weapon from player's hand of given index  
Example:

Code:

|  |
| --- |
| local.weapon = $player[1] getactiveweap 0  or  local.weapon = $player[1] getactiveweap(0) |

Result:

Code:

|  |
| --- |
| Weapon entity. |

You can use weaponhand index from 0-2 range, but it's preffered to use only 0 index, because other indexes my return false values that might crash server  
  
}  
  
  
  
Player (player) -> Sentient -> Animate -> Entity -> SimpleEntity -> Listener -> Class  
  
.secfireheld ( *void* )  
{

Returns 1 if player is holding secondary fire button.  
Example:

Code:

|  |
| --- |
| if( $player[1].secfireheld == 1 )  ... |

Result:

Code:

|  |
| --- |
| 1 = player is holding secondary fire button  0 = opposite |

}  
  
Player (player) -> Sentient -> Animate -> Entity -> SimpleEntity -> Listener -> Class  
  
.userinfo ( *void* )  
{

Returns player's userinfo  
Example:

Code:

|  |
| --- |
| local.userinfo = $player[1].userinfo |

Result:

Code:

|  |
| --- |
| String with player's userinfo |

}  
  
  
  
Player (player) -> Sentient -> Animate -> Entity -> SimpleEntity -> Listener -> Class  
  
.inventory( *void* )  
{

Returns player's inventory  
Example:

Code:

|  |
| --- |
| local.inventory= $player[1].inventory  local.inventorySize = $player[1].inventory.size  local.item1 = $player[1].inventory[0] |

Result:

Code:

|  |
| --- |
| Array with entities in player's inventory. You can assign it to a variable or access directly. |

}

Player (player) -> Sentient -> Animate -> Entity -> SimpleEntity -> Listener -> Class  
  
bindweap ( *Entity* weapon )  
{

Binds weapon to player. Sets him as weapon owner.  
2nd use of the command will unbind the weapon from player.  
Example:

Code:

|  |
| --- |
| $player[1] bindweap local.weapon  local.weapon anim fire  $player[1] bindweap local.weapon |

Result:

Code:

|  |
| --- |
| Sets player as weapon owner. |

This is sort of a hack&trick scripting command. It should only be used by experienced users and only like shown in the example - just before firing the weapon and just after, to unbind it from the player. Otherwise you can have errors, weapon model glued to player, or server crashes. It should be used only for some kind of remote turrets etc.  
  
}