List of CE specific functions and variables:

Global Variables:
TrainerOrigin : A variable that contains the path of the trainer that launched cheat engine (Only set when launched as a trainer)
process : A variable that contains the main modulename of the currently opened process
MainForm: The main ce gui
AddressList: The address list of the main ce gui

Global Functions:
getCEVersion(): Returns a floating point value specifying the version of cheat engine
getCheatEngineFileVersion(): Returns the full version data of the cheat engine version. A raw integer, and a table containing major, minor, release and build

getOperatingSystem(): Returns 0 if CE is running in Windows, 1 for Mac

darkMode(): Returns true if CE is running in windows Dark Mode. Has no effect on mac

activateProtection(): Prevents basic memory scanners from opening the cheat engine process (Not that useful)
enableDRM(altitude OPTIONAL, secondaryprocessid OPTIONAL) : Prevents normal memory scanners from reading the Cheat Engine process (kernelmode) The secondaryprocessid lets you protect another process. E.g the game itself, so they can't easily see what you change

fullAccess(address,size): Changes the protection of a block of memory to writable and executable
setMemoryProtection(address, size, {r:boolean; w: Boolean; x: Boolean}): Sets the given protection on the address range. Note, some systems do not support X and W to be true at the same time
getMemoryProtection(address): Returns a table {r: Boolean; w: Boolean; x: Boolean)

loadTable(filename, merge OPTIONAL): Loads a .ct or .cetrainer. If merge is provided and set to true it will not clear the old table
loadTable(stream ,merge OPTIONAL, ignoreluascriptdialog BOOLEAN): Loads a table from a stream object
saveTable(filename, protect OPTIONAL, dontDeactivateDesignerForms OPTIONAL): Saves the current table. If protect is provided and set to true and the filename has the .CETRAINER extension, it will protect it from reading normally
saveTable(stream, dontDeactivateDesignerForms OPTIONAL): Saves the current table to a stream object

signTable(filename) : If the current CE install has a valid cheta engine signature, this will sign the specific table with that signature (will pop up the password request)

note: addresses can be strings, they will get interpreted by ce's symbolhandler

copyMemory(sourceAddress: integer, size: integer, destinationAddress:integer SEMIOPTIONAL, Method:integer OPTIONAL):
  Copies memory from the given address to the destination address
  If no destinationAddress is given(or nil), CE will allocate a random address for you

  Method can be:
    nil/0: Copy from target process to target process
    1: Copy from target process to CE Memory
    2: Copy from CE Memory to target process
    3: Copy from CE Memory to CE Memory

  Returns the address of the copy on success, nil on failure

compareMemory(address1: integer; address2: integer; size: integer; method: integer)
  Compares the memory and returns true if the same or false and and index where the first disparity is

  Method can be:
  0: Target to Target
  1: Address1=Target Address2=CE
  2: Address1=CE Address2=CE

readBytes(address,bytecount, ReturnAsTable) : returns the bytes at the given address. If ReturnAsTable is true it will return a table instead of multiple bytes
  Reads the bytes at the given address and returns a table containing the read out bytes

writeBytes(address, x,x,x,x,...) : Write the given bytes to the given address from a table
writeBytes(address, table) : Write the given bytes to the given address from a table

readShortInteger(address) / readByte(address) : Reads a 8-bit integer from the specified address
readSmallInteger(address) : Reads a 16-bit integer from the specified address
readInteger(address) : Reads a 32-bit integer from the specified address
readQword(address): Reads a 64-bit integer from the specified address
readPointer(address): In a 64-bit target this equals readQword, in a 32-bit target readInteger()
readFloat(address) : Reads a single precision floating point value from the specified address
readDouble(address) : Reads a double precision floating point value from the specified address
readString(address, maxlength, widechar OPTIONAL) : Reads a string till it encounters a 0-terminator. Maxlength is just so you won't freeze for too long, set to 6000 if you don't care too much. Set WideChar to true if it is encoded using a widechar formatting
writeShortInteger(address,value) / writeByte(address,value) : Writes a 8-bit integer to the specified address. Returns true on success

writeSmallInteger(address,value) : Writes a 16-bit integer to the specified address. Returns true on success
writeInteger(address,value) : Writes a 32-bit integer to the specified address. Returns true on success
writeQword(address, value): Write a 64-bit integer to the specified address. Returns true on success
writePointer(address,value)
writeFloat(address,value) : Writes a single precision floating point to the specified address. Returns true on success
writeDouble(address,value) : Writes a double precision floating point to the specified address. Returns true on success
writeString(address,text, widechar OPTIONAL) : Write a string to the specified address. Returns true on success

readBytesLocal(address,bytecount, ReturnAsTable) : See readBytes but then it's for Cheat engine's memory
readSmallIntegerLocal(address) : Reads a 16-bit integer from the specified address in CE's memory
readIntegerLocal(address) : Reads a 32-bit integer from the specified address in CE's memory
readQwordLocal(address) : Reads a 64-bit integer from the specified address in CE's memory
readPointerLocal(address) : ReadQwordLocal/ReadIntegerLocal depending on the cheat engine build
readFloatLocal(address) : Reads a single precision floating point value from the specified address in CE's memory
readDoubleLocal(address) : Reads a double precision floating point value from the specified address in CE's memory
readStringLocal(address, maxlength, widechar OPTIONAL)
writeSmallIntegerLocal(address,value) : Writes a 16-bit integer to the specified address in CE's memory. Returns true on success
writeIntegerLocal(address,value) : Writes a 32-bit integer to the specified address in CE's memory. Returns true on success
writeQwordLocal(address,value) : Writes a 64-bit integer to the specified address in CE's memory. Returns true on success
writePointerLocal(address,value)
writeFloatLocal(address,value) : Writes a single precision floating point to the specified address in CE's memory. Returns true on success
writeDoubleLocal(address,value) : Writes a double precision floating point to the specified address in CE's memory. Returns true on success
writeStringLocal(address,string, widechar OPTIONAL)
writeBytesLocal(address, x,x,x,x,...) : See writeBytes but then it's for Cheat Engine's memory
writeBytesLocal(address, table, , count) : See writeBytes but then it's for Cheat Engine's memory

readSmallInteger, readInteger, readSmallIntegerLocal, readIntegerLocal
can also have second boolean parameter. If true, value will be signed.

signExtend(value,mostSignificantBit): integer - Extends the bits so that if it's MSB

bit is set, it will be negative

wordToByteTable(number): {}   - Converts a word to a bytetable
dwordToByteTable(number): {}   - Converts a dword to a bytetable
qwordToByteTable(number): {}   - Converts a qword to a bytetable
floatToByteTable(number): {}   - Converts a float to a bytetable
doubleToByteTable(number): {}   - Converts a double to a bytetable
extendedToByteTable(number): {}   - Converts an extended to a bytetable
stringToByteTable(string): {}   - Converts a string to a bytetable
wideStringToByteTable(string): {}   - Converts a string to a widestring and converts that to a bytetable

byteTableToWord(table, OPTIONAL signed:boolean, OPTIONAL tableindex=1): number - Converts a bytetable to a word
byteTableToDword(table, OPTIONAL signed:boolean, OPTIONAL tableindex=1): number - Converts a bytetable to a dword
byteTableToQword(table, OPTIONAL tableindex=1): number   - Converts a bytetable to a qword
byteTableToFloat(table, OPTIONAL tableindex=1): number   - Converts a bytetable to a float
byteTableToDouble(table, OPTIONAL tableindex=1): number   - Converts a bytetable to a double
byteTableToExtended(table, OPTIONAL tableindex=1): number   - Converts a bytetable to an extended and converts that to a double
byteTableToString(table, OPTIONAL tableindex=1): string   - Converts a bytetable to a string
byteTableToWideString(table, OPTIONAL tableindex=1): string - Converts a bytetable to a widestring and converts that to a string

bOr(int1, int2)   : Binary Or
bXor(int1, int2) : Binary Xor
bAnd(int1, int2) : Binary And
bShl(int, int2)   : Binary shift left
bShr(int, int2)   : Binary shift right
bNot(int)   : Binary not

enumMemoryRegions() : Returns an indexed table containing the memorylayout. Each entry consists out of: BaseAddress, AllocationBase, AllocationProtect, RegionSize, State, Protect, Type

writeRegionToFile(filename, sourceaddress,size) : Writes the given region to a file. Returns the number of bytes written
readRegionFromFile(filename, destinationaddress)

resetLuaState(): This will create a new lua state that will be used. (Does not destroy the old one, so memory leak)
getGlobalVariable(string): Returns the given variable from the main lua state. Only basic types are supported. (Handy for new lua state threads)
setGlobalVariable(string, something): Sets the global variable names string in the main lua state. Only basic types are supported

createRef(...): integer - Returns an integer reference that you can use with getRef. Useful for objects that can only store integers and need to reference lua objects. (Component.Tag...)
getRef(integer): ... - Returns whatever the reference points out
destroyRef(integer) - Removes the reference

encodeFunction(function): string - Converts a given function into an encoded string that you can pass on to decodeFunction
decodeFunction(string): function - Converts an encoded string back into a function. Note that the string must be made on the same architecture as it is currently running.

encodeFunctionEx(string,pathtodll OPTIONAL) - See encodeFunction but uses a script instead of a function, and lets you specify which lua dll to use
encodeFunctionCompatibilityMode(state) - Set to true if you with to let encodeFunction generate code that can also be loaded by versions before 7.6 (Note that older CE versions can not load scripts made by different cpu architectures)

getTranslationFolder(): Returns the path of the current translation files. Empty if there is no translation going on
loadPOFile(path): Loads a .PO file used for translation
translate(string): Returns a translation of the string. Returns the same string if it can't be found
translateID(translationid: string, originalstring: string OPTIONAL): Returns a translation of the string id

convertToUTF8(string/bytetable, originalcodepagenumber): Conversts a given string/bytearray into a utf8 converted from the given codepage number
convertFromUTF8(utf8string, targetcodepage): Converts an UTF8 encoded string into a string of the desired codepage
ansiToUTF8(string): Converts a string in Ansi encoding to UTF8
UTF8ToAnsi(string): Converts a string in UTF8 encoding to Ansi
Note: GUI components mainly show in UTF8, some other functions use Ansi, try to find out which ones...

enumModules(processid OPTIONAL):
  Returns a table containing information about each module in the current process, or the specified processid
  Each entry is a table with fields
    Name : String containing the modulename
    Address: Integer representing the address the module is loaded
    Size: Integer which holds the size of the module (7.6+)
    Is64Bit: Boolean set to true if it's a 64-bit module
    PathToFile: String to the location this module is loaded

md5memory(address, size): Returns a md5 sum calculated from the provided memory.
md5file(pathtofile): Returns a md5 sum calculated from the file.

getFileVersion(pathtofile): returns the 64-bit file version, and a table that has split up the file version into major, minor, release and build

getFileList(Path:string, searchMask:string OPTIONAL, SearchSubDirs: boolean OPTIONAL, DirAttrib: integer OPTIONAL): Returns an indexed table with filenames
getDirectoryList(Path:string, SearchSubDirs: boolean OPTIONAL): Returns an indexed table with directory names
extractFileName(filepath): returns the filename of the path
extractFileExt(filepath): returns the file extension of the path
extractFileNameWithoutExt(filepath): Returns the filename of the path, without the extension
extractFilePath(filepath): removes the filename from the path

getTempFolder() : Returns the path to the temp folder
fileExists(pathtofile): Returns true if a file exists at that path
deleteFile(pathtofile): Returns true if a file existed at that path, and now not anymore

enableWindowsSymbols(): Will download the PDB files of Windows and load them (Takes a long time the first time)
getAddress(string, local OPTIONAL): returns the address of a symbol. Can be a modulename or an export. set Local to true if you wish to querry the symboltable of the ce process
enableKernelSymbols(): Will check the option for kernelmode symbols in memory view (Gets only the exports unless enableWindowsSymbols() is used)
getAddressSafe(string, local OPTIONAL, shallow OPTIONAL): returns the address of a symbol, or nil if not found. Similar to getAddress when errorOnLookup is false, but returns nil instead
getSymbolInfo(symbolname): Returns a table as defined by the SymbolList class object (modulename, searchkey, address, size)
getModuleSize(modulename): Returns the size of a given module (Use getAddress to get the base address)
getRTTIClassName(address): Returns the classname of a given structure based on RTTI information (assuming it can be found, returns nil if not or unknown)

getStructureElementsFromName(name): When PDB's are loaded, this will return an indexed table with elements: {offset, name, vartpe}

loadNewSymbols(): Scans for changes in the modulelist and loads the symbols of new modules
reinitializeSymbolhandler(waittilldone: BOOLEAN OPTIONAL, default=TRUE): reinitializes the symbolhandler. E.g when new modules have been loaded
reinitializeDotNetSymbolhandler(modulename OPTIONAL): Reinitializes only the DotNet part of the symbol list. (E.g After an ILCode has been JITed) (6.4+)
reinitializeSelfSymbolhandler(waittilldone: BOOLEAN OPTIONAL, default=TRUE): reinitializes the selfsymbolhandler. E.g when new modules have been loaded to CE process
waitForSections(): Waits till the sections have been enumerated
waitForExports(): Waits till all DLL Exports are loaded
waitForDotNet(): Waits till all .NET symbols are loaded (this includes DLL Exports)

waitForPDB(): Waits till all PDB symbols are loaded (this includes DLL Exports, and .NET)
symbolsDoneLoading(): Returns true when all symbols have been loaded
searchPDBWhileLoading(state: boolean): Will interrupt symbol enum to query the debughelp symbol handler about a specific symbol. This can take a while. Default is false
symbolHandlerAddModule(pathtofile,baseaddress): Loads the symbols of an arbitrary module (can be a pdb) at the given address

errorOnLookupFailure(state): If set to true (default) address lookups in stringform will raise an error if it can not be looked up. This includes symbolnames that are not defined and pointers that are bad. If set to false it will return 0 in those cases
  (Useful for pointers that don't work 100% of the time)
6.4+:Returns the original state

waitforsymbols(state): If set to true looking up a symbol will wait for the symbol to be loaded(default true)

generateAPIHookScript(address:string, addresstojumpto:string, addresstogetnewcalladdress:string OPT, ext:string OPT, targetself:boolean OPT) : Generates an auto assembler script which will hook the given address when executed
assemble(line, address OPTIONAL, assemblePreference OPTIONAL, skipRangeCheck OPTIONAL): assembles a single line of code and returns a byte array of the generated code.
  address is the address to assemble this code at
  assemblePreference: apNone=0, apShort=1, apLong=2, apFar=3
  skipRangeCheck is a boolean. Which will skip range checks if true and just assembles it, no matter of how wrong the result will be
autoAssemble(text, targetself OPTIONAL, disableInfo OPTIONAL)
autoAssemble(text, disableInfo OPTIONAL)
  Runs the auto assembler with the given text. Returns true on success, with as secondary a table you can use when disabling (if targetself is set it will assemble into Cheat Engine itself). If disableInfo is provided the [Disable] section will be handled
    DisableInfo contains the following fields which might be of interest:
      allocs: name: {address, size, prefered} All the allocations done by the script
      registeredsymbols: indexed array containing the registered symbols
      ccodesymbols: a SymbolList class object that contains all the symbols of this compilation. See the SymbolList class for more info . By default this list is registered to the main symbolhandler. You can use this to unregister the list
      exceptionlist: indexed array of start addresses of exception ranges
      symbols: all the symbols in the script(labels included) and their address. Indexable by the name of the symbols
  When the script uses {$C}/{$CCODE} tags, there will be a 3th result containing warnings during the compilation. Nil if there are none

autoAssembleCheck(text, enable, targetself) : Checks the script for syntax errors. Returns true on succes, false with an error message on failure

compile(text, address OPTIONAL, targetself OPTIONAL, kernelmode OPTIONAL, nodebug OPTIONAL) : Compiles C code and returns a table with the addresses of the symbols on success, or nil with a secondary result containing the errormessage
compile({indexedtable containing scripts}, address OPTIONAL, targetself OPTIONAL)) : ^ but allows multiple scripts to be compiled into one
compileFiles({filelist}, address OPTIONAL, targetself OPTIONAL)): ^ but takes an indexed list of files
    {filelist} is an indexed table of filenames.
      7.6+: The filelist entry can be a table formatted as: {virtualfilename, stream} and .o files are supported

compileTCCLib() - Compiles the TCC library functions some C code may need to function internally

addCIncludePath(path) : Adds an extra default include path for the compile() function
removeCIncludePath(path) : Removes a specific path previously added with addCIncludePath
setCustomTCCParameters(params: string): Sets custom parameters to be used when compiling C code

compileCS(text, {references}, coreAssembly OPTIONAL) - Compiles c# code and returns the autogenerated filename. references is a list of c# assemblies this code may reference. This file will be deleted when CE closes (or next time another CE closes and it's not in use anymore). Note: This requires .NET 4 to be installed, even if the target is mono. Tip: Handy with injectDotNetDLL
dotNetExecuteClassMethod(pathtodll, namespace, classname, methodname, parameters: string): integer - Inside CE, call a method in a .NET class declared as : public static int methodname(string parameters) For the target process version, look into injectDotNetDLL

registerEXETrainerFeature(FeatureName:String, function():table): adds a new feature to the exe trainer generator window, and calls your function when the user builds an .exe trainer. The function should return a table with table entries: PathToFile and RelativePath.
  example output:
    [1]:
      PathToFile=c:\cefolder\autorun\mycode.lua
      RelativePath="autorun\"

    [2]:
      PathToFile=c:\cefolder\autorun\dlls\mycode.lua
      RelativePath="autorun\mylib.dll"

Note: This runs AFTER the table has been packaged already

unregisterEXETrainerFeature(id)

registerAutoAssemblerCommand(command, function(parameters, syntaxcheckonly)): Registers an auto assembler command to call the specified function. The command will be replaced by the string this function returns when executed. The function can be called twice. Once for syntax check and symbol lookup(1), and the second time for actual execution by the assembler(2) if it has not been removed in phase1.
  Note: The callback function can return multiple values
  Nil, <String>: Will raise an error with the given string
  MultilineString: Replaces the line in the script with the given strings.

 If the function returns nil, and as secondary parameter a string, this will make the auto assembler fail with that error

unregisterAutoAssemblerCommand(command)

registerLuaFunctionHighlight(functionname): Makes the lua highlighter show the functionname as a functionkeyword
unregisterLuaFunctionHighlight(functionname): Removes the given name from showing up as a functionkeyword

registerSymbolLookupCallback(function(string):integer, location): ID 6.4+
  Registers a function to be called when a a symbol is parsed
  Location determines at what part of the symbol lookup the function is called
    slStart: The very start of a symbol lookup. Before tokenization
    slNotInt: Called when it has been determined it's not a hexadecimal only string. Before tokenization
    --The following locations can be called multiple times for one string as they are called for each token and appended token
    slNotModule: Called when it has been determined the current token is not a modulename
    slNotUserdefinedSymbol: Called when it has been determined it's not a userdefined symbol
    slNotSymbol: Called when it has been determined it's not a symbol in the symbollist
    slFailure: Called when it has no clue what the given string is

    Note: slNotSymbol and slFailure are similar, but failure comes only if there's no token after the current token that can be concatenated. Else slNotSymbol will loop several times till all tokens make up the full string

  Return an Integer with the corresponding address if you found it. Nil or 0 if you didn't.

unregisterSymbolLookupCallback(ID): Removes the callback

registerAddressLookupCallback(function(integer):string, first: boolean): ID
  Registers a function to be called when the name of an address is requested. First will make it get put at the front of the list. Also, if first is true it will go even before CE's internal lookup

unregisterAddressLookupCallback(ID): Removes the callback

registerGlobalStructureListUpdateNotification(function(sender)): ID - Calls the given function each time the list is updated
unregisterGlobalStructureListUpdateNotification(id) - removes the callback with the given ID

registerStructureAndElementListCallback(function StructureListCallback(), function elementlistcallback(id1,id2)) : Registers a function to be called when a structure needs to be dissected
  function StructureListCallback() will be a function that returns an array of list of structures in table format
  the entries are build up as:
    name: string - name of the structure
    id1: integer - id you can use for whatever(e.g moduleid). It will be passed on to elementlistcallback when this structure is picked
    id2: integer - id you can use for whatever(e.g structureid inside the module). It will be passed on to elementlistcallback when this structure is picked

  function elementlistcallback(id1,id2) will be a function that returns an array of structure elements in table format
  the entries are build up as:
    name: string
    offset: integer
    vartype: variabletype (look up vtByte, vtWord, etc..)

  tip: If you return an empty table the structure will not be created. You can use this to create the structure layout yourself and register that instead

unregisterStructureAndElementListCallback(ID)

registerStructureDissectOverride(function(structure, baseaddress): table):
  same as onAutoGuess, but is called by the structure dissect window when the user chooses to let cheat engine guess the structure for him.
  Use the structure object to fill it in
  Return true if you have filled it in, or false or nil if you did not

  Tip: Use inputQuery to ask the user the size if your function doesn't do that automatically

unregisterStructureDissectOverride(ID)

registerStructureNameLookup((function(address): name, address OPTIONAL), first OPTIONAL):
  Registers a function to be called when dissect data asks the user for the name of a new structure define. If you have code that can look up the name of a structure, and perhaps also the real starting point, you can use this to improve the data dissection.
  If first is true it will be called first, even when CE already has an idea what it is

unregisterStructureNameLookup(ID)

registerAssembler(function(address, instruction):bytetable)
  Registers a function to be called when the single line assembler is invoked to convert an instruction to a list of bytes
  Return a bytetable with the specific bytes, or nil if you wish to let another function, or the original x86 assembler to assemble it

unregisterAssembler(ID): Unregisters the registered assembler

registerAutoAssemblerPrologue(function(script, syntaxcheck), postAOB:boolean=false)
  Registers a function to be called when the auto assembler is about to parse an auto assembler script. The script you get is after the [ENABLE] and [DISABLE] tags have been used to strip the script to the according one, but before comment stripping and trimming has occured

  script is a Strings object which when changed has direct effect to the script

unregisterAutoAssemblerPrologue(ID)

registerAutoAssemblerTemplate(name, function(script: TStrings; sender: TFrmAutoInject), shortcut OPTIONAL): id - Registers an template for the auto assembler. The script parameter is a TStrings object that has a direct connection to the current script. (All script parsing is up to you...). Returns an ID
unregisterAutoAssemblerTemplate(ID)

registerProcessListCallback(function(): list) : Registers a processlist override.
  list is an indexed array with it's value field a table formatted as:
    Pid: integer - the processid
    Name: string - The processname
    Image: Graphic - The image/icon for the process. Can be any graphic inherited object like bmp, png, etc... can be nil. The user owns the image so make sure to free the image yourself when not needed anymore

  Return nil if you do not wish to override the list

registerModuleListCallback(function(): list) - Registers a modulelist override

```
  list is an indexed array with it's value field a table formatted as:
    Name : String containing the modulename
    Address: Integer representing the address the module is loaded
    Size: Integer which holds the size of the module
    Is64Bit: Boolean set to true if it's a 64-bit module
    PathToFile: String to the location this module is loaded
    Sections: { } Optional table describing the sections of the module.  If nil, CE
will try to figure it out by reading the path
      Sections is an indexed array with it's value field a table formatted as:
        Name: string - Section name
        Size: integer - Size of the section
        Address: integer - Address of the section
        FileAddress: integer - Location of this section in the file

generateCodeInjectionScript(script: Tstrings, address: string, farjmp: boolean) -
Adds a default codeinjection script to the given script
generateAOBInjectionScript(script: Tstrings, symbolname: string, address: string,
commentradius(default 10), farjmp: boolean) - Adds an AOB injection script to the
given script
generateFullInjectionScript(script: Tstrings, address: string, commentradius(default
10), farjmp: boolean) - Adds a Full Injection script to the given script

getNextAllocNumber(script: TStrings): integer - scans the given script for
alloc(newmem## and returns the next unused newmem number)
addSnapshotAsComment(script: TStrings, address: integer, radius(Default 10)) -
creates a comment section for AA scripts that contains a snapshot of the original
code
getUniqueAOB(address): AOBString,Offset - Scans for a unique AOB for the given
address and returns the AOB as a string, and an offset applied in case the aob
returned doesn't start at the given address



showMessage(text) : shows a messagebox with the given text
inputQuery(caption, prompt, initialstring): Shows a dialog where the user can input
a string. This function returns the given string, or nil on cancel
showSelectionList(title, caption, stringlist, allowCustomInput OPTIONAL, formname
OPTIONAL): integer,string - Shows a menu with the given list. It returns the
linenumber (starting at 0) and the selected string.  Linenumber is -1 if the user
was allowed to enter custom input

messageDialog(text, type, buttons...) : pops up a messagebox with a specific
icon/sound with the specified buttons (mbok, mbyes, ....)
messageDialog(title, text, type, buttons...): ^ but adds a custom title
messageDialog(text) : shows an information dialog with the text

sleep(milliseconds): pauses for the number of specified milliseconds (1000= 1
sec...)

getProcesslist(Strings): Fills a Strings inherited object with the processlist of
```

the system. Format: %x-pidname
getProcesslist(): Returns a table with the processlist (pid - name)
getWindowlist(Strings): Fills a Strings inherited object with the top-window list of the system. Format: %x-windowcaption
getWindowlist(): Returns a table with the windowlist (pid - window caption). The table is formatted as : {pid,{id,caption}}

getThreadlist(List): fills a List object with the threadlist of the currently opened process. Format: %x
getHandleList(filter OPTIONAL): returns a table with all the handles in the system(Filter 0=everything, 1=target process handles only, 2 handles to target process, 3 handles to ce process). Each handle entry has fields: ProcessID, ObjectTypeIndex, HandleAttributes, HandleValue, Object and GrantedAccess. Note: Object will be invalid if you use the 32-bit CE on a 64-bit windows

closeRemoteHandle(handle, processid OPTIONAL): Closes the handle of a process.

duplicateHandle(handle): Duplicates the provided CE based handle into the target process (You still need tell the target about this handle, like an injected dll data block)
duplicateHandle(handle, Mode (0/1)): If mode is 0 then it's the same as just duplicateHandle(handle), but if it's 1 it duplicates the target process handle into CE's process
duplicateHandle(handle, fromPID, toPID): Copies the handle from the given process to the target process.

function onOpenProcess(processid):
  If this function is defined it will be called whenever cheat engine opens a process.
  Note: The the same process might be opened multiple times in a row internally
  Note 2: This function is called before attachment is fully done. You can call reinitializeSymbolhandler() to force the open to complete, but it will slow down process opens. Alternatively, you could launch a timer which will run when the opening has finished

MainForm.OnProcessOpened: function(processid, processhandle, caption) - Define this if you want to be notified when a new process has been opened. Called only once from the main thread. It is recommended to use this instead of onOpenProcess

function onTableLoad(before): If defined this function will be called twice when a table gets loaded. Once before the loading, and once after.

function onLuaError(errorstring): if defined this function will be called when a lua error happens. The string returned will be the error shown

function onPointerMapGenerationStart(): Called when a pointermap is about to be generated. The specific thread that calls tyhis will do so. (Use this to adjust the

readProcessMemory output)
function onPointerMapGenerationFinish(): Called when a pointermap has finished being generated

getOpenedProcessID() : Returns the currently opened process. If none is open, returns 0
getOpenedProcessHandle(): Returns the handle of the currently opened process
getProcessIDFromProcessName(name) : returns a processid
openProcess(processid) : causes cheat engine to open the given processid
openProcess(processname): causes cheat engine to find and open the given process
openFileAsProcess(filename,is64bit OPTIONAL,startaddress OPTIONAL): causes cheat engine to open the file with memory access as if it's a process
getOpenedFileSize(): Returns the file of the opened file
saveOpenedFile(filename OPTIONAL): Saves the changes of the opened file, set filename if you want a different file
setPointerSize(size): Sets the size cheat engine will deal with pointers in bytes. (Some 64-bit processes can only use 32-bit addresses)
getPointerSize() : Gets the current pointersize
setAssemblerMode(int): 0=32-bit, 1=64-bit
pause() : pauses the current opened process
unpause(): resumes the current opened process

getCPUCount(): Returns the number of CPU's
cpuid(EAX,ECX): returns a table with CPUID info (EAX, EBX, ECX, EDX)

gc_setPassive(state: boolean): enables/disables the passive garbage collector
gc_setActive(state: boolean, interval: integer, minsize: integer): enables/disables the active garbage collector and lets you configure the interval and minimim size

getSystemMetrics(index): Retrieves the specified system metric or system configuration setting
(https://msdn.microsoft.com/en-us/library/windows/desktop/ms724385.aspx)

getScreenDPI(): Returns the Dots/Pixels Per Inch of the screen
getScreenHeight(): Returns the screen height
getScreenWidth(): Returns the screen width

getWorkAreaHeight(): Returns the work area height
getWorkAreaWidth(): Returns the work area width

invertColor(color): Returns the inverted color
getScreenCanvas(): Returns a Canvas object you can use to write to the screen (Note: Not as useful as you may think)

getPixel(x,y) : returns the rgb value of the pixel at the specific screen coordinate
getMousePos: returns the x,y coordinates of the mouse
setMousePos(x,y): sets the mouse position

isKeyPressed(key) : returns true if the specified key is currently pressed
keyDown(key) : causes the key to go into down state
keyUp(key) :causes the key to go up
doKeyPress(key) : simulates a key press

mouse_event(flags, x OPTIONAL, y OPTIONAL, data OPTIONAL, extra OPTIONAL) - The mouse_event windows API. Check MSDN for information on how to use

shortCutToText(shortcut): Returns the textual representation of the given shortut value (integer) (6.4+)
textToShortCut(shortcutstring): Returns an shortcut integer that the given string represents. (6.4+)

convertKeyComboToString(key1,...): Returns a string representation of the given keys like the hotkey handler does
convertKeyComboToString({key1,...}): ^

outputDebugString(text): Outputs a message using the windows OutputDebugString message. You can use tools like dbgview to read this. Useful for testing situations where the GUI freezes

shellExecute(command, parameters OPTIONAL, folder OPTIONAL, showcommand OPTIONAL): Executes a given command
runCommand(exepath, parameters or {parameter1, parameter2}, pathtoexecutein OPTIONAL): Executes the given command and returns the output of the command as a string and the exitcode as an integer (Should not open a console window) If pathtoexecutein is not provided the path will be the current working directory of CE

getTickCount() : Returns the current tickcount since windows was started. Each tick is one millisecond
getTimeStamp() : Returns a string containing the current time (H:M:S.ms)
processMessages() : Lets the main eventhandler process the new messages (allows for new button clicks)
processMessagesPaintOnly() : Handles some basic paint and null messages so windows won't mark it as unresponsive during long runs. The main difference from processMessages is that it does not handle mouse or keyboard events
inMainThread(): Returns true if the current code is running inside the main thread (6.4+)
integerToUserData(int): Converts a given integer to a userdata variable
userDataToInteger(UserDataVar): Converts a given userdata variable to an integer

synchronize(function(...), ...): Calls the given function from the main thread. Returns the return value of the given function
queue(function(...),...): calls the given function from the main thread. Does not wait for the result. Note: Be sure to synchronize and call checkSynchronize() before freeing the calling thread. Note2: The queue will be emptied and NOT executed if the thread is freed. So it's not recommended without setting freeOnTerminate to

false
checkSynchronize(timeout OPTIONAL): Call this from an infinite loop in the main thread when using threading and synchronize calls. This will execute any queued synchronize calls

writeToClipboard(text): Writes the given text to the clipboard
readFromClipboard(): Reads the text from the clipboard

speedhack_setSpeed(speed) : Enables the speedhack if needed and sets the specific speed
speedhack_getSpeed(): Returns the last set speed

registerSpeedhackCallbacks(
  OnActivate()   : Handled, result, errormsg - Called when the speedhack gets activated.   Return true if you handled it. False if you wish to let CE's default handler handle it. If true, the 2nd result is success or failure. And on failure, the 3th result is the error message shown to the user
  OnSetSpeed(speed): Handled, result, errormsg - Called when the speedhack speed gets set.   ' ' '
): CallbackID - Registers callbacks for use in the speedhack

unregisterSpeedhackCallbacks(CallbackID) - Unregisters callbacks for the speedhack

injectDLL(filename, skipsymbolreloadwait OPTIONAL): Injects a dll or dylib, and returns true on success
injectLibrary(filepath, skipsymbolreloadwait OPTIONAL): Same as injectDLL, just sounds better

injectDotNetDLL(dllpath, FullClassName, MethodName,parameterstring, timeout optional)

executeCode(address, parameter OPTIONAL, timeout OPTIONAL) : address - Executes a stdcall function with 1 parameter at the given address in the target process and wait for it to return. The return value is the result of the function that was called
executeCodeLocal(addres, parameter OPTIONAL): address - Executes a stdcall function with 1 parameter at the given address in the target process. The return value is the result of the function that was called

executeCodeEx(callmethod, timeout, address, {type=x,value=param1} or param1,{type=x,value=param2} or param2,...)
callmethod: 0=stdcall, 1=cdecl
  timeout: Number of milliseconds to wait for a result. nil or -1, infitely. 0 is no wait (will not free the call memory, so beware of it's memory leak)
  address: Address to execute
  {type,value} : Table containing the value type, and the value

```
    {
    type: 0=integer (32/64bit) can also be a pointer
          1=float (32-bit float)
          2=double (64-bit float)
          3=ascii string (will get converted to a pointer to that string)
          4=wide string (will get converted to a pointer to that string)

    value: anything base type that lua can interpret
    }
  if just param is provided CE will guess the type based on the provided type

executeMethod(callmethod, timeout, address, {regnr=0..15,classinstance=xxxxxxxx} or
classinstance, {type=x,value=param1} or param1, {type=x,value=param2} or param2,...)
- Executes a method.
  regnr can be:
    0: R/EAX
    1: R/ECX
    2: R/EDX
    3: R/EBX
    4: R/ESP
    5: R/EBP
    6: R/ESI
    7: R/EDI
    8: R8
    9: R9
    10: R10
    11: R11
    12: R12
    13: R13
    14: R14
    15: R15


  If no register number is provided then ECX(1) is picked
  If instance is nil it is the same as executeCodeEx

executeCodeLocalEx(address, {type=x,value=param1} or param1,{type=x,value=param2} or
param2,...)
  Calls a function using the given callmethod and parameters


  If a direct parameter is given instead of a table entry describing the type, CE
will 'guess' the type it is

  Returns the E/RAX value returned by the called function (if no timeout or other
interruption)
```

loadPlugin(dllnameorpath): Loads the given plugin. Returns nil on failure. On success returns a value of 0 or greater

loadFontFromStream(memorystream) : Loads a font from a memory stream and returns an id (handle) to the font for use with unloadLoadedFont
unloadLoadedFont(id)

autoGuess(address, nostring OPTIONAL, nodouble OPTIONAL) : returns a valuetype of what CE `guesses` it is. (keep in mind: it is a guess. Don't rely on it)

onAutoGuess(function) :
  Registers a function to be called whenever autoguess is used to predict a variable type
  function override (address, ceguess): Return the variable type you want it to be. If no change, just return ceguess

closeCE() : just closes ce
hideAllCEWindows() : makes all normal ce windows invisible (e.g trainer table)
unhideMainCEwindow() : shows the main cheat engine window

getAutoAttachList(): returns the AutoAttach StringList object. It can be controlled with the stringlist_ routines (it's not recommended to destroy this list object)

AOBScan(x,x,x,x,...):
scans the currently opened process and returns a StringList object containing all the results. don't forget to free this list when done
Bytevalue of higher than 255 or anything not an integer will be seen as a wildcard
AOBScan(aobstring, OPTIONAL protectionflags, OPTIONAL alignmenttype, OPTIONAL alignmentparam): see above but here you just input one string
AOBScanUnique(aobstring, OPTIONAL protectionflags, OPTIONAL alignmenttype, OPTIONAL alignmentparam)- Integer: scans for the aobstring and returns the first result it finds and nil if nothing is found. Make sure it is unique as it will return the first result found as it will return any random match
AOBScanModuleUnique(modulename, aobstring, OPTIONAL protectionflags, OPTIONAL alignmenttype, OPTIONAL alignmentparam)- Integer : scans for the aobstring in the designated module

Regarding eventhandlers. You can initialize them using both a string of a functionname or the function itself.
If initialized using a function itself it won't be able to get saved in the table

allocateMemory(size, BaseAddress OPTIONAL, Protection OPTIONAL): Allocates some memory into the target process
deAlloc(address, size OPTIONAL): Frees allocated memory

allocateSharedMemory(name, size):
  Creates a shared memory object in the attached process of the given size if it doesn't exist yet. If size is not given and there is no shared region with this name then the default size of 4096 is used
  It then maps this shared memory block into the currently targeted process. It returns the address of this mapped region in the target process. Keep in mind that a process can map the same block multiple times, so keep track of your assignments
allocateSharedMemoryLocal(name, size): Same as allocateSharedMemory but then for the Cheat Engine process itself

createSection(size) : Creates a 'section' in memory
mapViewOfSection(section, preferedBaseAddress OPTIONAL): Maps the section to memory
unMapViewOfSection(baseaddress): Unmaps a section from memory

getForegroundProcess() : Returns the processID of the process that is currently on top

findWindow(classname OPTIONAL, caption OPTIONAL): windowhandle - Finds a window with the given classname and/or windowname
getWindow(windowhandle, command) : windowhandle - Gets a specific window based on the given window (Check MSDN getWindow for the command description)
getWindowCaption(windowhandle) : string - Returns the caption of the window
getWindowClassName(windowhandle): string - Returns the classname of the window
getWindowProcessID(windowhandle): processid - Returns the processid of the process this window belongs to
getForegroundWindow() - windowhandle : Returns the windowhandle of the topmost window

sendMessage(hwnd, msg, wparam, lparam): result - Sends a message to a window. Those that wish to use it, should know how to use it (and fill in the msg id's yourself)
hookWndProc(hwnd, function(hwnd, msg, wparam, lparam), ASYNC: BOOL) - Hooks a window's wndproc procedure. The given function will receive all functions. Return 0 to say you handled it. 1 to let the default windows handler deal with it. Or anything else, to let the original handler deal with it. Besides the return value, you can also return hWnd, Msg, lParam and wParam, modified, or nil for the original value. Set ASYNC to true if you don't want to run this in the CE GUI. (faster, but you can't touch gui objects)
unhookWndProc(hwnd) - call this when done with the hook. Not calling this will result in the process window behaving badly when you exit CE

cheatEngineIs64Bit(): Returns true if CE is 64-bit, false if 32-bit
targetIs64Bit(): Returns true if the target process is 64-bit, false if 32-bit
targetIsX86(): Returns true if the target process is x86 based
targetIsArm(): Returns true if the target process is arm based
targetIsAndroid(): Returns true if the target process is running on the Android OS
targetIsRosetta(): Returns true if the target process is running inside the rosetta emulator (macos only)

getABI(): Returns 0 for windows calling convention. Returns 1 for unix/linux calling convention

getCheatEngineDir(): Returns the folder Cheat Engine is located at
getCheatEngineProcessID(): Returns the processid of cheat engine

getAutorunPath() : Returns the autorun path

disassemble(address): Disassembles the given address and returns a string in the format of "address - bytes - opcode : extra"
splitDisassembledString(disassembledstring): Returns 4 strings. The address, bytes, opcode and extra field

getInstructionSize(address): Returns the size of an instruction (basically it disassembles the instruction and returns the number of bytes for you)
getPreviousOpcode(address): Returns the address of the previous opcode (this is just an estimated guess)

disassembleBytes(hexadecimalbytestring or {bytetable},address OPTIONAL) : Disassembles the given bytes and returns the result.

beep() : Plays the fabulous beep/ping sound!
playSound(stream, waittilldone OPTIONAL): Plays the given memorystream containing a .WAV formatted memory object. If waittilldone is true the script will stop executing till the sound has stopped
playSound(tablefile, waittilldone OPTIONAL) : Takes the memorystream from the tablefile and plays it.
  There are two tablefiles predeclared inside cheat engine "Activate" and "Deactivate" . You are free to use or override them

speak(text, waittilldone OPTIONAL): Speaks a given text. If waitTillDone is true the thread it's in will be frozen till it is done
speak(text, flags): Speaks a given text using the given flags. https://msdn.microsoft.com/en-us/library/speechplatform_speakflags.aspx
speakEnglish(text, waittilldone OPTIONAL) - will try the English voice by wrapping the given text into an XML statement specifying the english voice. Will not say anything if no Egnlish language is present. Do not use SPF_IS_NOT_XML flag and SPF_PARSE_SSML won't work in this situation

printf(...) : Same as print(string.format(...))
setProgressState(state): Sets the state of the cheatengine task in the taskbar (windows only) values: tbpsNone, tbpsIndeterminate, tbpsNormal, tbpsError, tbpsPaused
setProgressValue(current, max): Sets the state of the cheatengine task progress status

getUserRegistryEnvironmentVariable(name): string - Returns the environment variable stored in the user registry environment
setUserRegistryEnvironmentVariable(name, string) - Sets the environment variable stored in the user registry environment
is when you've changed the environment variables in the registry. This will cause at least the shell to update so you don't have to reboot. (It's always recommended to reboot though)

stringToMD5String(string): Returns an md5 hash string from the provided string

getFormCount() : Returns the total number of forms assigned to the main CE application
getForm(index): Returns the form at the specific index

registerFormAddNotification(function(form)): Registers a function to be called when a form is attached to ce's form list. This is useful for extentions that add new functionality to certain existing forms. It returns an object you can use with unregisterFormAddNotification. Note: This gets called before the form is properly initialized. It's recommended to use form.registerFirstShowCallback so your code gets called after initialization
unregisterFormAddNotification(Object)

getSettingsForm(): Returns the main settings form
getMemoryViewForm() : Returns the main memoryview form class object which can be accessed using the Form_ class methods and the methods of the classes it inherits from. There can be multiple memory views, but this will only find the original/base
getMainForm() : Returns the main form class object which can be accessed using the Form class methods and the methods of the classes it inherits from

getApplication() : Returns the application object. (the titlebar)
getAddressList() : Returns the cheat table addresslist object
getFreezeTimer() : Returns the freeze timer object
getUpdateTimer() : Returns the update timer object

setGlobalKeyPollInterval(integer): Sets the global keypoll interval. The interval determines the speed of how often CE checks if a key has been pressed or not. Lower is more accurate, but eats more cpu power

setGlobalDelayBetweenHotkeyActivation(integer): Sets the minimum delay between the activation of the same hotey in milliseconds. Affects all hotkeys that do not set their own minimum delay

getXBox360ControllerState(ControllerID OPTIONAL) : table - Fetches the state of the connected xbox controller. Returns a table containing the following fields on success:

ControllerID : The id of the controller (between 0 and 3)
PacketNumber : The packet id of the state you see. (use to detect changes)
GAMEPAD_DPAD_UP : D-PAD Up (boolean)
GAMEPAD_DPAD_DOWN: D-PAD Down (boolean)
GAMEPAD_DPAD_LEFT: D-PAD Left (boolean)
GAMEPAD_DPAD_RIGHT: D-PAD Right (boolean)
GAMEPAD_START: Start button (boolean)
GAMEPAD_BACK: Back button (boolean)
GAMEPAD_LEFT_THUMB: Left thumb stick down (boolean)
GAMEPAD_RIGHT_THUMB: Right thumb stick down (boolean)

GAMEPAD_LEFT_SHOULDER: Left shoulder button (boolean)
GAMEPAD_RIGHT_SHOULDER: Right shoulder button (boolean)

GAMEPAD_A: A button (boolean)
GAMEPAD_B: B button (boolean)
GAMEPAD_X: X button (boolean)
GAMEPAD_Y: Y button (boolean)

LeftTrigger: Left trigger (integer ranging from 0 to 255)
RightTrigger: Right trigger (integer ranging from 0 to 255)

ThumbLeftX: Horizontal position of the left thumbstick (-32768 to 32767)
ThumbLeftY: Verital position of the left thumbstick (-32768 to 32767)
ThumbRightX: Horizontal position of the right thumbstick (-32768 to 32767)
ThumbRightY: Vertical position of the right thumbstick (-32768 to 32767)

setXBox360ControllerVibration(ControllerID, leftMotor, rightMotor) - Sets the speed of the left and right vibrating motor inside the controller. Range (0 to 65535 where 0 is off)

undefined property functions. Not all properties of all classes have been explicitly exposed to lua, but if you know the name of a property of a specific class you can still access them (assuming they are declared as published in the pascal class declaration)
getPropertyList(class) : Returns a stringlist object containing all the published properties of the specified class (free the list when done) (Note, not all classed with properties have 'published' properties. E.g: stringlist)
setProperty(class, propertyname, propertyvalue) : Sets the value of a published property of a class (Won't work for method properties)

getProperty(class, propertyname) : Gets the value of a published property of a class (Won't work for method properties)
setMethodProperty(class, propertyname, function): Sets the method property to the specific function
getMethodProperty(Class, propertyname): Returns a function you can use to call the original function

registerSymbol(symbolname, address, OPTIONAL donotsave): Registers a userdefined symbol. If donotsave is true this symbol will not get saved when the table is saved
unregisterSymbol(symbolname)

enumRegisteredSymbols(): Returns a table with elements containing {symbolname, address, OPTIONAL {allocsize, processid, donotsave}}

deleteAllRegisteredSymbols() : Deletes all symbols registered with registerSymbols, both in AA and Lua scripts (Does not remove registered symbolLists)

getNameFromAddress(address,ModuleNames OPTIONAL=true, Symbols OPTIONAL=true, Sections OPTIONAL=false): Returns the given address as a string. Registered symbolname, modulename+offset, or just a hexadecimal string depending on what address
inModule(address) : returns true if the given address is inside a module
inSystemModule(address) : returns true if the given address is inside a system module
getCommonModuleList: Returns the commonModuleList stringlist. (Do not free this one)

AOBScan("aobstring", protectionflags OPTIONAL, alignmenttype OPTIONAL, alignmentparam HALFOPTIONAL):
protectionflags is a string.
  X=Executable W=Writable memory C=Copy On Write. Add a + to indicate that flag MUST be set and a - to indicate that that flag MUST NOT be set. (* sets it to don't care)
  Examples:
    +W-C = Writable memory exluding copy on write and doesn't care about the Executable flag
    +X-C-W = Find readonly executable memory
    +W = Finds all writable memory and don't care about copy on write or execute
    "" = Find everything (is the same as "*X*C*W")

alignmenttype is an integer:
  0=No alignment check
  1=Address must be dividable by alignmentparam
  2=Address must end with alignmentparam
alignmentparam is a string which either holds the value the addresses must be dividable by or what the last digits of the address must be

-debugging

debug variables
EFLAGS
32/64-bit: EAX, EBX, ECX, EDX, EDI, ESI, EBP, ESP, EIP
64-bit only: RAX, RBX, RCX, RDX, RDI, RSI, RBP, RSP, RIP, R8, R9, R10, R11, R12, R13, R14, R15 : The value of the register

Debug related routines:
function debugger_onBreakpoint():
When a breaking breakpoint hits (that includes single stepping) and the lua function debugger_onBreakpoint() is defined it will be called and the global variables EAX, EBX, ... will be filled in
Return 0 if you want the userinterface to be updated and anything else if not (e.g: You continued from the breakpoint in your script)

createProcess(path, parameters OPTIONAL, debug OPTIONAL, breakonentrypoint OPTIONAL) : Creates a process. If debug is true it will be created using the windows debugger and if breakonentry is true it will cause a breakpoint to occur on entrypoint

debugProcess(interface OPT): starts the debugger for the currently opened process (won't ask the user) Optional interface: 0=default, 1=windows debug, 2=VEHDebug, 3=Kerneldebug

debug_isDebugging(): Returns true if the debugger has been started
debug_getCurrentDebuggerInterface() : Returns the current debuggerinterface used (1=windows, 2=VEH 3=Kernel, 4=mac native, 5=gdb, nil=no debugging active)
debug_canBreak(): Returns true if there is a possibility the target can stop on a breakpoint. 6.4+
debug_isBroken(): Returns true if the debugger is currently halted on a thread
debug_isStepping(): Returns true if the debugger was single stepping an instruction earlier
debug_getBreakpointList(): Returns a lua table containing all the breakpoint addresses
debug_breakThread(threadid): Breaks the thread with the specific threadID (Note: The thread may not break instantly and may have to be awakened first)

debug_addThreadToNoBreakList(threadid): This will cause breakpoints on the provided thread to be ignored
debug_removeThreadFromNoBreakList(threadid): removed the threadid from the list

debug_setBreakpointForThread(threadid, address, size OPTIONAL, trigger OPTIONAL, breakpointmethod OPTIONAL, functiontocall() OPTIONAL) : sets a breakpoint of a

specific size at the given address for the specified thread. if trigger is bptExecute then size is ignored. If trigger is ignored then it will be of type bptExecute, which obviously also ignores the size then as well. (Other triggers are bptAccess and bptWrite)
debug_setBreakpoint(address, size OPTIONAL, trigger OPTIONAL, breakpointmethod OPTIONAL, functiontocall() OPTIONAL)
debug_setBreakpoint(address, size OPTIONAL, trigger OPTIONAL, functiontocall() OPTIONAL)
debug_setBreakpoint(address, functiontocall() OPTIONAL)
debug_removeBreakpoint(address) : if the given address is a part of a breakpoint it will be removed
debug_continueFromBreakpoint(continueMethod) : if the debugger is currently waiting to continue you can continue with this. Valid parameters are :co_run (just continue), co_stepinto(when on top of a call, follow it), co_stepover (when on top of a call run till after the call)
debug_getXMMPointer(xmmregnr) :
  Returns the address of the specified xmm register of the thread that is currently broken
  This is a LOCAL Cheat Engine address. Use Local memory access functions to read and modify
  xmmregnr can be 0 to 15 (0 to 7 on 32-bit)

The following routines describe last branch recording. These functions only work when kernelmode debugging is used and using windows XP (vista and later work less effective or not at all because the operating system interferes. Might also be intel specific. A dbvm upgrade in the future might make this work for windows vista and later)
debug_setLastBranchRecording(boolean): When set the Kernel debugger will try to record the last branch(es) taken before a breakpoint happens
debug_getMaxLastBranchRecord() : Returns the maximum branch record your cpu can store (-1 if none)
debug_getLastBranchRecord(index): Returns the value of the Last Branch Record at the given index (when handling a breakpoint)

function debugger_onModuleLoad(modulename, baseaddress) :
this routine is called when a module is loaded. Only works for the windows debugger
return 1 if you want to cause the debugger to break

Changing registers:
When the debugger is waiting to continue you can change the register variables. When you continue those register values will be set in the thread's context

If the target is currently stopped on a breakpoint, but not done through an onBreakpoint function. The context won't be set.
You can get and set the context back with these functions before execution continues"

debug_getContext(BOOL extraregs) - Fills the global variables for the regular registers. If extraregs is true, it will also set FP0 to FP7 and XMM0 to XMM15
debug_setContext(BOOL extraregs)
debug_updateGUI() - Will refresh the userinterface to reflect the new context if the debugger was broken

detachIfPossible() : Detaches the debugger from the target process (if it was attached)

getComment(address) : Gets the userdefined comment at the specified address
setComment(address, text) : Sets a userdefined comment at the specifried address. %s is used to display the autoguess value if there is one
getHeader(address) : Gets the userdefined header at the specified address
setHeader(address) : Sets the userdefined header at the specified address

registerBinUtil(config) Registers a binutils toolset with CE (for assembling and disassembling in other cpu instruction sets)
config is a table containing several fields that describe the tools, and lets you specify extra parameters

Name : The displayed name in the binutils menu in memview
Description: The description for this toolset
Architecture: used by the objdump -m<architecture> (required)
ASParam : extra parameters to pass on to AS (optional)
LDParam : extra parameters to pass on to LD
OBJDUMPParam: extra parameters to pass on to OBJDUMP
OnDisassemble: a lua function that gets called each time an address is disassembled. The return value will be passed on to OBJDUMP
Path: filepath to the binutils set
Prefix: prefix (e.g: "arm-linux-androideabi-")
DisassemblerCommentChar: Depending on which target you're disassembling, the comment character can be different. (ARM=";" x86='#')

class helper functions
inheritsFromObject(object): Returns true if given any class
inheritsFromComponent(object): Returns true if the given object inherits from the Component class
inheritsFromControl(object): Returns true if the given object inherits from the Control class
inheritsFromWinControl(object): Returns true if the given object inherits from the WinControl class

createClass(classname): Creates an object of the specified class (Assuming it's a registered class and has a default constructor)

createComponentClass(classname, owner): Creates an object of the specified component inherited class

Class definitions
Object class: (Inheritance:)
Properties:
  ClassName: String - The name of class (Read only)
Methods:
  getClassName(): Returns the classname
  fieldAddress(fieldname: string): Returns the address of the specific field
  methodAddress(methodname: string)
  methodName(address: integer)
  destroy(): Destroys the object

Component Class: (Inheritance: Object)
properties
  ComponentCount: Integer - Number of child components . Readonly
  Component[int]: Component - Array containing the child components. Starts at 0. Readonly
  ComponentByName[string]: Component - Returns a component based on the name. Readonly
  Name: string - The name of the component
  Tag: integer - Free to use storage space. (Useful for id's)
  Owner: Component - Returns the owner of this object. Nil if it has none

methods
  getComponentCount() : Returns the number of components attached to his component
  getComponent(index) : Returns the specific component
  findComponentByName(name) : Returns the component with this name
  getName() : Return the name
  setName(newname) : Changes the name
  getTag() : Sets an integer value. You can use this for ID's
  setTag(tagvalue) : Get the tag value
  getOwner() : Returns the owner of this component

Control Class: (Inheritance: Component->Object)
properties:
  Caption: string - The text of a control
  Top : integer - The x position
  Left : integer - The y position
  Width : integer - The width of the control
  Height : integer - The height of the control
  ClientWidth: integer - The usable width inside the control (minus the borders)
  ClientHeight: integer - The usable height the control (minus the borders)
  Align: AlignmentOption - Alignment of the control

  Enabled: boolean - Determines if the object is usable or greyed out
  Visible: boolean - Determines if the object is visible or not
  Color: ColorDefinition/RGBInteger - The color of the object. Does not affect the caption
  RGBColor: RGBInteger - The color of the object in RGB formatting
  Parent: WinControl - The owner of this control
  PopupMenu: PopupMenu - The popup menu that shows when rightclicking the control
  Font: Font - The font class associated with the control
  OnClick: function(sender) - The function to call when a button is pressed
  OnChangeBounds:function(sender) - Called when the size or position of the control changes

methods:
  getLeft()
  setLeft(integer)
  getTop()
  setTop(integer)
  getWidth()
  setWidth(integer)
  getHeight()
  setHeight()
  setCaption(caption) : sets the text on a control. All the GUI objects fall in this category
  getCaption() : Returns the text of the control
  setPosition(x,y): sets the x and y position of the object base don the top left position (relative to the client array of the owner object)
  getPosition(): returns the x and y position of the object (relative to the client array of the owner object)
  setSize(width,height) : Sets the width and height of the control
  getSize() : Gets the size of the control
  setAlign(alignmentoption): sets the alignment of the control
  getAlign(alignmentoption): gets the alignment of the control
  getEnabled() : gets the enabled state of the control
  setEnabled(boolean) : Sets the enabled state of the control
  getVisible() : gets the visible state of the control
  setVisible(boolean) : sets the visible state of the control
  getColor() : gets the color
  setColor(rgb) : Sets the color
  getParent() : Returns nil or an object that inherits from the Wincontrol class
  setParent(wincontrol) : Sets the parent for this control
  getPopupMenu()
  setPopupMenu()
  getFont(): Returns the Font object of this object
  setFont(): Assigns a new font object. (Not recommended to use. Change the font object that's already there if you wish to change fonts)
  repaint(): Invalidates the graphical area of the control and forces and update
  refresh() : updates the control (usually a repaint)
  update() : Only updates the invalidated areas
  setOnClick(functionnameorstring) : Sets the onclick routine
  getOnClick(): Gets the onclick function

  doClick(): Executes the current function under onClick
  bringToFront(): Changes the z-order of the control so it'd at the top
  sendToBack(): Changes the z-order of the control so it'd at the back
  screenToClient(x,y): Converts screen x,y coordinates to x,y coordinates on the control
  clientToScreen(x,y): Converts control x,y coordinates to screen coordinates

GraphicsObject : (GraphicsObject->Object)

Region Class : (Region->GraphicsObject->Object)
createRegion(): Created an empty region

properties
-
methods
  addRectangle(x1, y1, x2, y2): Adds a rectangle to the region
  addPolygon(tablewithcoordinates): Adds an array of 2D locations. (example : {{0,0},{100,100}, {0,100}} for a triangle)

WinControl Class: (Inheritance: Control->Component->Object)
properties
  Handle: Integer - The internal windows handle
  DoubleBuffered: boolean - Graphical updates will go to a offscreen bitmap which will then be shown on the screen instead of directly to the screen. May reduce flickering
  ControlCount : integer - The number of child controls of this wincontrol
  Control[] : Control - Array to access a child control
  OnEnter : function - Function to be called when the WinControl gains focus
  OnExit : function - Function to be called when the WinControl loses focus

methods
  getControlCount() Returns the number of Controls attached to this class
  getControl(index) : Returns a WinControl class object
  getControlAtPos(x,y): Gets the control at the given x,y position relative to the wincontrol's position
  canFocus(): returns true if the object can be focused
  focused(): returns boolean true when focused
  setFocus(): tries to set keyboard focus the object
  setShape(Region): Sets the region object as the new shape for this wincontrol
  setShape(Bitmap):
  setOnEnter(function) : Sets an onEnter event. (Triggered on focus enter)
  getOnEnter()
  setOnExit(function) : Sets an onExit event. (Triggered on lost focus)
  getOnExit()
  setLayeredAttributes(Key, Alpha, Flags) : Sets the layered state for the control

```
if possible (Only Forms are supported in in windows 7 and earlier)
      flags can be a combination of LWA_ALPHA and/or LWA_COLORKEY
      See msdn SetLayeredWindowAttributes for more information



MenuItem class(Inheritance: Component->Object)
createMenuItem(ownermenu) : Creates a menu item that gets added to the owner menu

properties
  Caption : String - Text of the menu item
  Shortcut : string - Shortcut in textform to trigger the menuitem
  Count : integer - Number of children attached to this menuitem
  Menu: Menu - The menu this item resides in
  Parent: MenuItem - The menuitem this item hangs under
  MenuIndex: integer - The position this menu item is in it's parent
  ImageList: ImageList
  ImageIndex: integer - Which image of the attached ImageList to show
  Item[] : Array to access each child menuitem
  [] : Item[]
  OnClick: Function to call when the menu item is activated
  FontColor: Color of the font. (Only works when in dark mode)

methods
  getCaption() : Gets the caption of the menu item
  setCaption(caption) : Sets the caption of the menu item
  getShortcut(): Returns the shortcut for this menu item
  setShortcut(shortcut): Sets the shortcut for this menuitem. A shortcut is a string
in the form of ("ctrl+x")
  getCount()
  getItem(index) : Returns the menuitem object at the given index
  add(menuitem) : Adds a menuItem as a submenu item
  insert(index, menuitem): Adds a menuItem as a submenu item at the given index
  delete(index)
  clear() - Deletes all children under this menuitem (frees the menu item, so it's
gone)
  setOnClick(function) : Sets an onClick event
  getOnClick()
  doClick(): Executes the onClick method if one is assigned



Menu Class: (Inheritance: Component->Object)
properties
  Items : MenuItem - The base MenuItem class of this menu (readonly)
methods
  getItems() : Returns the main MenuItem of this Menu

MainMenu Class: (Inheritance: Menu->Component->Object)
```

createMainMenu(form)
  The mainmenu is the menu at the top of a window

PopupMenu Class: (Inheritance: Menu->Component->Object)
createPopupMenu(owner)
  The popup menu is the menu that pops up when showing the (rightclick) context of an control

Strings Class: (Inheritance : Object) (Mostly an abstract class)
properties
  LineBreak: String - the character(s) to count as a linebreak
  Text : String - All the strings in one string
  Count: Integer - The number of strings in this list
  String[]: String - Array to access one specific string in the list
  Data[]: Integer - Array to access the data of a specific string in the list
  [] = String[]

methods
  clear() : Deletes all strings in the list
  add(string, data:integer OPTIONAL) : adds a string to the list. Returns the index
  addText([[strings]]) : adds multiple strings at once
  delete(index) : Deletes a string from the list
  getText() : Returns all the strings as one big string
  setText(string) : Sets the strings of the given strings object to the given text (can be multiline)
  indexOf(string): Returns the index of the specified string. Returns -1 if not found
  insert(index, string): Inserts a string at a specific spot moving the items after it

  getCount(): Returns the number is strings in the list
  remove(string); Removes the given string from the list
  loadFromFile(filename, ignoreencoding OPTIONAL default true) : Load the strings from a textfile. If ignoreEncoding is false then the file will be loaded with the best encoding the loader can guess
  saveToFile(filename) : Save the strings to a textfile

  getString(index) : gets the string at the given index
  setString(index, string) : Replaces the string at the given index
  getData(index) : Returns the integer value stored in the string
  setData(index, integer): Sets the integer value stored in the string

  beginUpdate() : Stops updates from triggering other events (prevents flashing)
  endUpdate(): call after beginUpdate

ImageList Class: (Inheritance:)
List containing images. Used by several components for images

createImageList(owner OPTIONAL): creates an imagelist object

properties
  Count: integer - Number of images in the list
  DrawingStyle: 'dsFocus', 'dsSelected', 'dsNormal', 'dsTransparent'
  Height: integer
  Width: integer
  Masked: boolean
  Scaled: boolean
  OnChange: function(sender)

methods
  add(bitmap, bitmapmask OPTIONAL):integer - Adds a new bitmap the list and returns the index of the newly added entry
  draw(canvas, x,y, index) - Draws the image at the index to the specific x,y coordinates on the canvas
  getBitmap(index, bitmap, OPTIONAL effect) - Draw the specified image to the provided bitmap. Effect can be : 0=gdeNormal, 1=gdeDisabled, 2=gdeHighlighted, 3=gdeShadowed, 4=gde1Bit

Stringlist Class: (Inheritance : Strings->Object)
createStringlist() : Creates a stringlist class object (for whatever reason, lua strings are probably easier to use)

properties
  Sorted : boolean - Determines if the list should be sorted
  Duplicates : DuplicatesType - Determines how duplicates should be handled when the list is sorted
  CaseSensitive: boolean - Determines if the list is case sensitive or not.

methods
  getDuplicates() : returns the duplicates property
  setDuplicates(Duplicates) : Sets the duplicates property (dupIgnore, dupAccept, dupError)
  getSorted() : returns true if the list has the sorted property
  setSorted(boolean) : Sets the sorted property
  getCaseSensitive() : Returns true if the case sensitive property is set
  setCaseSensitive(boolean): Sets the case sensitive property

OrderedList Class: (Inheritance: Object)
properties
  Count: integer - The number of items in the list

methods
  push(integer)   - Use this to add an item to the end of the list
  pop(): integer - Use this to take an item from the end of the list
  peek(): integer - Use this to to see what the last item of the list is

Application Class: (Inheritance: CustomApplication->Component->Object)

properties
  Title: The title of cheat engine in the bar
  Icon: The icon of Cheat Engine inn the bar

methods
  bringToFront(): Shows the cheat engine app
  processMessages()
  terminate()
  minimize()

ControlScrollBar Class (Inheritance: Object)
  Increment: Word - The amount the position moves when using the scrollbar arrows
  Page: Word - slider size in pixels
  Smooth: Boolean
  Position: Integer - (limited to 0 to range-page)
  Range: Integer
  Tracking: Boolean - Gives feedback when the slider is moved
  Visible: Boolean

ScrollingWinControl Class (Inheritance:
CustomControl->WinControl->Control->Component->Object)
properties
  HorzScrollBar: ControlScrollBar
  VertScrollBar: ControlScrollBar

Form Class: (Inheritance:
ScrollingWinControl->CustomControl->WinControl->Control->Component->Object)
properties
  DesignTimePPI: integer - the PPI/DPI at the time the form was designed
  AllowDropFiles: boolean - Allows files to be dragged into the form
  ModalResult: integer - The current ModalResult value of the form. Note: When this
value gets set the modal form will close
  Menu: MainMenu - The main menu of the form

  OnClose: function(sender) - The function to call when the form gets closed
  OnDropFiles: function(sender, {filenames}) - Called when files are dragged on top
of the form. Filenames is an arraytable with the files
  FormState: FormState string ReadOnly - The current state of the form. Possible
values: fsCreating, fsVisible, fsShowing, fsModal, fsCreatedMDIChild,
fsBorderStyleChanged, fsFormStyleChanged, fsFirstShow, fsDisableAutoSize

methods
  fixDPI() : Resizes controls and fonts based on the current DPI and the DPI used to
create the form. Only use this on forms that are not designed with variable DPI in
mind
  centerScreen(); : Places the form at the center of the screen
  hide() : Hide the form
  show() : show the form

close(): Closes the form. Without an onClose this will be the same as hide
bringToFront(): Brings the form to the foreground
showModal() : show the form and wait for it to close and get the close result
isForegroundWindow(): returns true if the specified form has focus
setOnClose(function) : function (sender) : Return a CloseAction to determine how to close the window
getOnClose() : Returns the function
getMenu() : Returns the mainmenu object of this form
setMenu(mainmenu)

setBorderStyle(borderstyle): Sets the borderstyle of the window
getBorderStyle()

printToRasterImage(rasterimage): Draws the contents of the form to a rasterimage class object

registerCreateCallback(function(f)): userdata - Registers a function to be called when the form has finished being created
unregisterCreateCallback(userdata) - removes the specific callback
registerFirstShowCallback(function(f)): userdata - Registers a function to be called when the form is show the first time
unregisterFirstShowCallback(userdata) - removes the specific callback
registerCloseCallback(function(f)): userdata - Registers a function to be called when the form has been closed
unregisterCloseCallback(userdata) - removes the specific callback

dragNow() - Call this on mousedown on any object if you wish that the mousemove will drag the whole form arround. Useful for borderless windows (Dragging will stop when the mouse button is released)
saveFormPosition({IntegerTable OPTIONAL}) - Saves the current form position and dimensions and an optional list of integer. The name of the form must have been set to a unique name
loadFormPosition(): boolean, {IntegerTable OPTIONAL, can be nil} - Restores the form position and dimensions. On success returns true and a integer table if that was provided with the save. The name of the form must have been set to a unique name

CEForm Class: (Inheritance: Form->ScrollingWinControl->CustomControl->WinControl->Control->Component->Object)
createForm(visible OPT): creates a CEForm class object(window) and returns the pointer for it. Visible is default true but can be changed
createFormFromFile(filename): Returns the generated CEform
createFormFromStream(stream): Returns the generated CEform

properties
DoNotSaveInTable: boolean - Set this if you do not wish to save the forms in the table
methods
saveToFile(filename): Saves a userdefined form

  saveToStream(s): Saves the userdefined form to the given stream
  getDoNotSaveInTable(): Returns the DoNotSaveInTable property
  setDoNotSaveInTable(boolean): Sets the DoNotSaveInTable property
  saveCurrentStateAsDesign() : Sets the current state of the form as the state that will be saved when the table is saved

TfrmLuaEngine class: (Inheritance: Form->ScrollingWinControl->CustomControl->WinControl->Control->Component->Object)
getLuaEngine() : Returns the main lua engine form object (Creates it if needed)
createLuaEngine() : Creates a new lua engine form object. If there is no main luaengine window, this will become it.

properties
  mOutput: Memo - Output of the luaengine window
  mScript: SynEdit - Editor for the script

methods

TfrmAutoInject class: (Inheritance: Form->ScrollingWinControl->CustomControl->WinControl->Control->Component->Object)
createAutoAssemblerForm(script: string OPTIONAL): TFrmAutoInject - Spawns an autoassembler window with the optionally provided script

properties
  Assemblescreen: SynEdit - Editor for the script
  TabCount: integer
  TabScript[index]: string

methods
  addTab(): integer
  deleteTab(index)

GraphicControl Class: (Inheritance: Control->Component->Object)
properties
  Canvas: Canvas - The canvas for rendering this control

methods
  getCanvas() : Returns the Canvas object for the given object that has inherited from customControl

PaintBox class: (Inheritance: GraphicControl->Control->Component->Object)
createPaintBox(owner): Creates a Paintbox class object

Label Class: (Inheritance: GraphicControl->Control->Component->Object)
createLabel(owner): Creates a Label class object which belongs to the given owner. Owner can be any object inherited from WinControl

Splitter Class: (Inheritance: CustomControl->WinControl->Control->Component->Object)
createSplitter(owner): Creates a Splitter class object which belongs to the given owner. Owner can be any object inherited from WinControl

Panel Class: (Inheritance: CustomControl->WinControl->Control->Component->Object)
createPanel(owner): Creates a Panel class object which belongs to the given owner. Owner can be any object inherited from WinControl

properties
  Alignment: alignment
  BevelInner: panelBevel
  BevelOuter: panelBevel
  BevelWidth: Integer
  FullRepaint: boolean
methods
  getAlignment() : gets the alignment property
  setAlignment(alignment) : sets the alignment property
  getBevelInner()
  setBevelInner(PanelBevel)
  getBevelOuter()
  setBevelOuter(PanelBevel)
  getBevelWidth()
  setBevelWidth(BevelWidth)
  getFullRepaint()
  setFullRepaint(boolean)

Image Class: (Inheritance: GraphicControl->Control->Component->Object)
createImage(owner): Creates an Image class object which belongs to the given owner. Owner can be any object inherited from WinControl

properties
  Canvas: Canvas - The canvas object to access the picture of the image
  Transparent: boolean - Determines if some parts of the picture are see through (usually based on the bottomleft corner)
  Stretch: boolean - Determines if the picture gets stretched when rendered in the image component
  Picture: Picture - The picture to render

methods
  loadImageFromFile(filename)
  getStretch()
  setStretch(boolean)
  getTransparent()
  setTransparent(boolean)
  getCanvas()

```
  setPicture(picture)
  getPicture() : Returns the Picture object of this image


Edit Class: (Inheritance: WinControl->Control->Component->Object)
createEdit(owner): Creates an Edit class object which belongs to the given owner.
Owner can be any object inherited from WinControl

properties
  Text: string - The current contents of the editfield
  CaretPos: Point - The posaition of the caret
  PasswordChar: string[1] - When not set to char0 this will make the edit field show
the character instead of the given text
  SelText: string - The current selected contents of the edit field
  SelStart: number - The starting index of the current selection (zero-indexed)
  SelLength: number - The length of the current selection.
  OnChange: function - The function to call when the editfield is changed
  OnKeyPress: function - The function to call for the KeyPress event.
  OnKeyUp: function - The function to call for the KeyUp event.
  OnKeyDown: function - The function to call for the KeyDown event.

methods
  clear()
  copyToClipboard()
  cutToClipboard()
  pasteFromClipboard()
  selectAll()
  select(start, length OPTIONAL)
  selectText(start, length OPTIONAL) : Set the control's current selection. If no
length is specified, selects everything after start.
  clearSelection()
  getSelText()
  getSelStart()
  getSelLength()
  getOnChange()
  setOnChange(function)
  getOnKeyPress()
  setOnKeyPress(function)
  getOnKeyUp()
  setOnKeyUp(function)
  getOnKeyDown()
  setOnKeyDown(function)


Memo Class: (Inheritance: Edit->WinControl->Control->Component->Object)
createMemo(owner): Creates a Memo class object which belongs to the given owner.
Owner can be any object inherited from WinControl

properties
  Lines: Strings - Strings object for this memo
```

  WordWrap: boolean - Set if words at the end of the control should go to the next line
  WantTabs: Boolean - Set if tabs will add a tab to the memo. False if tab will go to the next control
  WantReturns: Boolean - Set if returns will send a event or not
  Scrollbars: Scrollstyle - Set the type of ascrollbars to show (ssNone, ssHorizontal, ssVertical, ssBoth,
    ssAutoHorizontal, ssAutoVertical, ssAutoBoth)

methods
  append(string)
  getLines() : returns a Strings class
  getWordWrap()
  setWordWrap(boolean)
  getWantTabs()
  setWantTabs(boolean)
  getWantReturns()
  setWantReturns(boolean)
  getScrollbars()
  setScrollbars(scrollbarenumtype) :
  Sets the scrollbars. Horizontal only takes affect when wordwrap is disabled
  valid enum types:
    ssNone : No scrollbars
    ssHorizontal: Has a horizontal scrollbar
    ssVertical: Has a vertical scrollbar
    ssBoth: Has both scrollbars
    ssAutoHorizontal: Same as above but only shows when there actually is something to scroll for
    ssAutoVertical: " " " " ...
    ssAutoBoth: " " " " ...

SynEdit class:
createSynEdit(owner,mode OPTIONAL): Creates a synedit object. mode: 0=Lua highlighting, 1=Auto Assembler highlighting 2=C code
properties
  Lines: Stringlist - Contains the text
  Gutter: Gutter - Gutter object
  ReadOnly: Boolean - Set to true for read only
  SelStart: integer
  SelEnd: integer
  SelText: string
  CanPaste: boolean
  CanRedo: boolean
  CanUndo: boolean
  CharWidth: integer READONLY
  LineHeight: integer READONLY
  CaretX, CaretY: integer

methods
  CopyToClipboard()
  CutToClipboard()
  PasteFromClipboard()
  ClearUndo()
  Redo()
  Undo()
  MarkTextAsSaved()
  ClearSelection();
  SelectAll();

ButtonControl Class: (Inheritance: WinControl->Control->Component->Object)

Button Class: (Inheritance: ButtonControl->WinControl->Control->Component->Object)
createButton(owner): Creates a Button class object which belongs to the given owner. Owner can be any object inherited from WinControl

properties
  ModalResult: ModalResult - The result this button will give the modalform when clicked

methods
  getModalResult(button)
  setModalResult(button, mr)

CheckBox Class: (Inheritance: ButtonControl->WinControl->Control->Component->Object)
createCheckBox(owner): Creates a CheckBox class object which belongs to the given owner. Owner can be any object inherited from WinControl

properties
  Checked: boolean - True if checked
  AllowGrayed: boolean - True if it can have 3 states. True/False/None
  State: checkboxstate - The state. (cbUnchecked=0, cbChecked=1, cbGrayed=2)
  OnChange: function - Function to call when the state it changed

methods
  getAllowGrayed()
  setAllowGrayed(boolean)
  getState(): Returns a state for the checkbox. (cbUnchecked, cbChecked, cbGrayed)
  setState(boolean): Sets the state of the checkbox
  onChange(function)

ToggleBox Class: (Inheritance: CheckBox->ButtonControl->WinControl->Control->Component->Object)
createToggleBox(owner): Creates a ToggleBox class object which belongs to the given owner. Owner can be any object inherited from WinControl

GroupBox Class: (Inheritance: WinControl->Control->Component->Object)
createGroupBox(owner): Creates a GroupBox class object which belongs to the given

owner. Owner can be any object inherited from WinControl

RadioGroup class: (Inheritance: GroupBox->WinControl->Control->Component->Object)
createRadioGroup(owner): Creates a RadioGroup class object which belongs to the given owner. Owner can be any object inherited from WinControl

properties
  Items: Strings - Strings derived object containings all the items in the list
  Columns: Integer - The number of columns to split the items into
  ItemIndex: Integer - The currently selected item
  OnClick: Called when the control is clicked

methods
  getRows(): Returns the number of rows
  getItems(): Returns a Strings object
  getColumns(): Returns the nuber of columns
  setColumns(integer)
  getItemIndex()
  setItemIndex(integer)
  setOnClick(function)
  getOnClick()

ListBox Class: (Inheritance: WinControl->Control->Component->Object)
createListBox(owner): Creates a ListBox class object which belongs to the given owner. Owner can be any object inherited from WinControl

properties
  MultiSelect: boolean - When set to true you can select multiple items
  Items: Strings - Strings derived object containings all the items in the list
  Selected[] - Returns true if the given line is selected. Use Items.Count-1 to find out the max index
  ItemIndex: integer - Get selected index. -1 is nothing selected
  Canvas: Canvas - The canvas object used to render on the object

methods
  clear()
  clearSelection() : Deselects all items in the list
  selectAll(): Selects all items in the list
  getItems(): Returns a strings object
  setItems(Strings): sets a strings object to the listbox
  getItemIndex()
  setItemIndex(integer)
  getCanvas()

Calendar Class: (Inheritance: WinControl->Control->Component->Object)
createCalendar(owner): Creates a Calendar class object which belongs to the given owner. Owner can be any object inherited from WinControl. Valid date is between

"September 14, 1752" and "December 31, 9999"

properties
  Date: string - current date of the Calendar, format: yyyy-mm-dd
  DateTime: number - days since December 30, 1899

methods
  getDateLocalFormat - returns current date of the Calendar, format: ShortDateFormat from OS local settings

ComboBox Class: (Inheritance: WinControl->Control->Component->Object)
createComboBox(owner): Creates a ComboBox class object which belongs to the given owner. Owner can be any object inherited from WinControl

properties
  Items: Strings - Strings derived object containings all the items in the list
  ItemIndex: integer - Get selected index. -1 is nothing selected
  Canvas: Canvas - The canvas object used to render on the object
  DroppedDown: boolean - True if currently dropped down (can be set as well)

methods
  clear()
  getItems()
  setItems()
  getItemIndex()
  setItemIndex(integer)
  getCanvas()
  getExtraWidth() : Returns the number of pixels not part of the text of the combobox (think about borders, thumb button, etc...)

ProgressBar Class: (Inheritance: WinControl->Control->Component->Object)
createProgressBar(owner): Creates a ProgressBar class object which belongs to the given owner. Owner can be any object inherited from WinControl

properties
  Min: integer - The minimum positionvalue the progressbar can have (default 0)
  Max: integer - The maximum positionvalue the progressbar can have (default 100
  Position: integer - The position of the progressbar
  Step: integer- The stepsize to step by when stepIt() is called

methods
  stepIt() - Increase position with "Step" size
  stepBy(integer) - increase the position by the given integer value
  getMax() - returns the Max property
  setMax(integer) - sets the max property
  getMin() - returns the min property

  setMin(integer)- sets the min property
  getPosition() - returns the current position
  setPosition(integer) - sets the current position
  setPosition2(integer) - sets the current position; without slow progress animation on Win7 and later

TrackBar Class : (Inheritance: WinControl->Control->Component->Object)
createTrackBar(owner): Creates a TrackBar class object which belongs to the given owner. Owner can be any object inherited from WinControl

properties
  Min: integer - Minimal value for the trackbar
  Max: integer - Maximum value for the trackbar
  Position: integer - The current position
  OnChange: function - Function to call when

methods
  getMax()
  setMax(integer)
  getMin(trackbar)
  setMin(trackbar, integer)
  getPosition(progressbar)
  setPosition(progressbar, integer)
  getOnChange()
  setOnChange(function)

CollectionItem Class: (Inheritance: Object)
Base class for some higher level classes. Often used for columns

properties
  ID: integer
  Index: integer - The index in the array this item belong to
  DisplayName: string

methods
  getID()
  getIndex()
  setIndex()
  getDisplayName()
  setDisplayName()

ListColumn class: (Inheritance: CollectionItem->Object)

```
properties
  AutoSize: boolean
  Caption: string
  MaxWidth: integer
  MinWidth: integer
  Width: integer
  Visible: boolean
methods
  getAutosize()
  setAutosize(boolean)
  getCaption()
  setCaption(caption)
  getMaxWidth()
  setMaxWidth(width)
  getMinWidth()
  setMinWidth(width)
  getWidth()
  setWidth(width)


Collection Class: (Inheritance: Object)

properties
  Count: integer
  Items[index]: CollectionItem
  []=Items[index]

methods
  clear(collection)
  getCount(collection)
  delete(collection, index)


ListColumns class : (Inheritance: Collection->Object)
properties
  Columns[]: Array to access a column
  [] = Columns[]

methods
  add(): Returns a new ListColumn object
  getColumn(index): Returns a ListColum object;
  setColumn(index, listcolumns): Sets a ListColum object (not recommended, use add
instead)

ListItem Class : (Inheritance: TObject)
properties
  Caption: boolean - The text of this listitem
  Checked: boolean - Determines if the checkbox is checked (if it has a checkbox)
  SubItems: Strings - The Strings object that hold the subitems
  Selected: boolean - Returns true if selected
```

  Index: integer - The index in the Items object of the owner of this listitem (readonly)
  ImageIndex: integer - The index in the attached imagelist (LargeImages/SmallImages)
  StateIndex: integer - The index in the attached imagelist (StateImages)
  Owner: ListItems - The ListItems object that owns this ListItem (readonly)
  Data: integer - Read/Write value up to the user to implement

methods
  delete()
  getCaption() : Returns the first columns string of the listitem
  setCaption(string) : Sets the first column string of the listitem
  getChecked() : Returns true if the listitem is checked
  setChecked(boolean): Sets the checkbox of the listbox to the given state
  getSubItems(): Returns a Strings object
  makeVisible(partial): Scrolls the listview so this item becomes visible (Cheat Engine 6.4 and later)
  displayRect(code): returns the displayed rectangle of the listitem. code can be: drBounds(0), drIcon(1), drLabel(2), drSelectBounds(3)
  displayRectSubItem(code): returns the displayed rectangle of the listitem. code can be: drBounds(0), drIcon(1), drLabel(2), drSelectBounds(3)

ListItems class : (Inheritance: TObject)
properties
  Count : Integer - The number of ListItems this object holds (Normally read only, but writable if OwnerData is true in the listview)
  Item[]: ListItem[] - Array to access each ListItem object
  [] = Item[]
methods
  clear()
  getCount()
  getItem(integer) : Return the listitem object at the given index
  add(): Returns a new ListItem object

Listview Class : (Inheritance: WinControl->Control->Component->Object)
createListView(owner): Creates a ListView class object which belongs to the given owner. Owner can be any object inherited from WinControl

properties
  Columns: ListColumns - The Listcolumns object of the listview (Readonly)
  Items: ListItems - The ListItems objects of the listview
  ItemIndex: integer - The currently selected index in the Items object (-1 if nothing is selected)
  Selected: ListItem - The currently selected listitem (nil if nothing is selected)
  TopItem: ListItem - The first visible item in the listview
  VisibleRowCount: integer - The number of lines currently visible
  Canvas: Canvas - The canvas object used to render the listview (Readonly)

  AutoWidthLastColumn: Boolean - When set to true the last column will resize when the control resizes
  HideSelection: Boolean - When set to true the selection will not hide when the focus leaves the control
  RowSelect: Boolean - When set to true the whole row will be selected instead of just the first column
  OwnerData: Boolean - When set to true the listview will call the onData function for every line being displayed. Use Items.Count to set the number of virtual lines
  LargeImages: ImageList
  SmallImages: ImageList
  StateImages: ImageList

  OnData: function(sender, ListItem) - Called when a listview with OwnerData true renders a line

  OnCustomDraw: function(Sender, {Top, Left, Bottom, Right}, DefaultDraw Optional): NewDefaultDraw
  OnCustomDrawItem: function(Sender, ListItem, {cdsSelected=true/false(nil), cdsGrayed=true/false(nil), cdsDisabled, cdsChecked, cdsFocused, cdsDefault, cdsHot, cdsMarked, cdsIndeterminate}, DefaultDraw Optional): NewDefaultDraw
  OnCustomDrawSubItem: function(Sender, ListItem, SubItemIndex, {cdsSelected=true/false(nil), cdsGrayed=true/false(nil), cdsDisabled, cdsChecked, cdsFocused, cdsDefault, cdsHot, cdsMarked, cdsIndeterminate}, DefaultDraw Optional): NewDefaultDraw

methods
  clear()
  getColumns() : ListColumns - Returns a ListColumns object
  getItemAt(x,y):ListItem - Returns the ListItem at the given index. nil if nothing
  getItems(): ListItems - Returns a ListItems object
  getItemIndex(): integer - Returns the currently selected index in the Items object
  setItemIndex(index: integer)- Sets the current itemindex
  getCanvas() : Canvas - Returns the canvas object used to render the listview
  beginUpdate() - Tells the listview to stop updating while you're busy
  endUpdate() - Applies all updates between beginUpdate and endUpdate

HeaderSection class : (Inheritance: CollectionItem)
properties
  Alignment: string - 'taLeftJustify', 'taRightJustify', 'taCenter'
  ImageIndex: imageindex
  MaxWidth: integer
  MinWidth: integer
  Text: String - The text of the headersection
  Width: integer - The width of the headersection
  Visible: boolean - Determines if the headersection is visible
  OriginalIndex: integer - The original index ignoring user reorganization (READONLY)

methods

HeaderSections class : (Inheritance: Collection->TObject)
properties
methods
  add(): THeaderSection - Adds a new header and returns it
  insert(index): THeaderSection - inserts a new header at the index and returns it
  delete(index) - deletes(and frees) the headersection at the given index

TreeNode class : (Inheritance: TObject)
properties
  Text: string - The text of the treenode
  Parent: Treenode - The treenode this object is a child of. (can be nil) (ReadOnly)
  Level: Integer - The level this node is at
  HasChildren: boolean - Set to true if it has children, or you wish it to have an expand sign
  Expanded: boolean - Set to true if it has been expanded
  Count : Integer - The number of children this node has
  Items[]: Treenode - Array to access the child nodes of this node
  [] = Items[]
  Index: Integer - The index based on the parent
  AbsoluteIndex: Integer - The index based on the TreeView's Treenodes object (Items)
  ImageIndex: integer - The image to show from the attached ImageList
  Selected: Boolean - Set to true if currently selected
  MultiSelected: Boolean - Set to true if selected as well, but not the main selected object
  Data: Pointer - Space to store 4 or 8 bytes depending on which version of CE is used
methods
  delete()
  deleteChildren()
  makeVisible()
  expand(recursive:boolean=TRUE OPTIONAL) : Expands the given node
  collapse(recursive:boolean=TRUE OPTIONAL) : collapses the given node
  getNextSibling(): Returns the treenode object that's behind this treenode on the same level
  getDisplayRect(TextOnly: Boolean=FALSE OPTIONAL): Returns a rect {Left,Top,Right,Bottom} describing the node
  add(text:string): Returns a Treenode object that is a child of the treenode used to create it

TreeNodes class : (Inheritance: TObject)
properties
  Count : Integer - The total number of Treenodes this object has

```
  Item[]: TreeNode - Array to access each node
  [] = Item[]
methods
  clear()
  getCount()
  getItem(integer) : Return the TreeNode object at the given index (based on the
TreeView's Treenodes)
  add(text:string): Returns a new root Treenode object
  insert(treenode, string): Returns a new treenode object that has been inserted
before the given treenode
  insertBehind(treenode, string): Returns a new treenode object that has been
inserted after the given treenode



Treeview Class : (Inheritance:
CustomControl->WinControl->Control->Component->Object)
createTreeView(owner)

properties
  Items: TreeNodes - The Treenodes object of the treeview (ReadOnly)
  Selected: TreeNode - The currently selected treenode

methods
  beginUpdate()
  endUpdate()
  getItems()
  getSelected()
  setSelected()
  fullCollapse()  : Collapses all the nodes, including the children's nodes
  fullExpand() : Expands all the nodes and all their children
  saveToFile(filename): Saves the contents of the treeview to disk
  loadFromFile(filename)



Timer Class : (Inheritance: Component->object)
createTimer(delay, function(...),...):
  Creates a timer object that waits the given delay, executes the given function,
and then selfdestructs.  Tip: Don't use the timer after it has ran

createTimer(owner OPT, enabled OPT):
  Creates a timer object. If enabled is not given it will be enabled by default
(will start as soon as an onTimer event has been assigned)
  Owner may be nil, but you will be responsible for destroying it instead of being
the responsibility of the owner object)

properties
  Interval: integer - The number of milliseconds (1000=1 second) between executions
  Enabled: boolean
```

```
  OnTimer: function(timer) - The function to call when the timer triggers

methods
  getInterval()
  setInterval(interval) : Sets the speed on how often the timer should trigger. In
milliseconds (1000=1 second)
  getOnTimer()
  setOnTimer(function(timer))
  getEnabled()
  setEnabled(boolean)

CustomControl class (CustomControl->WinControl->Control->Component->Object)
properties
  Canvas : The canvas object for drawing on the control/. Readonly
  OnPaint: an OnPaint event you can assign to do some extra painting
methods
  getCanvas() : Returns the Canvas object for the given object that has inherited
from customControl


Canvas Class : (Inheritance: CustomCanvas->Object)
properties
  Brush: Brush - The brush object
  Pen: Pen - The pen object
  Font: Font - The font object
  Width: integer - Width of the canvas
  Height: integer - Height of the canvas
  Handle: integer - DC handle of the canvas



methods
  getBrush(): Returns the brush object of this canvas
  getPen(): Returns the pen object of this canvas
  getFont(): Returns the font object of this canvas
  getWidth()
  getHeight()
  getPenPosition()
  setPenPosition(x,y)
  clear() - Clears the canvas

  line(sourcex, sourcey, destinationx, destinationy)
  lineTo(destinationx, destinationy)
  moveTo(destinationx, destinationy)
  rect(x1,y1,x2,y2) - Draws a rectangle
  fillRect(x1,y1,x2,y2) - Draws a filled rectangle
  roundRect(x1,y1,x2,y2,rx,ry) - Draws a rectangle with rounded corners
  drawFocusRect(x1,y1,x2,y2) - Draws the focus rectangle shape
  textOut(x,y, text)
  textRect(rect,x,y,text): write the text within the given rectangle. The text
```

supports some ansi escape characters
  getTextWidth(text)
  getTextHeight(text)
  getPixel(x,y)
  setPixel(x,y,color)
  floodFill(x,y, color OPTIONAL default=brush.Color, filltype OPTIONAL default=fsSurface): Fills the picture till/with a color.
    filltype can be
      fsSurface: fill till the color (it fills all except this color)
      fsBorder: fill this color (it fills only connected pixels of this color)

  ellipse(x1,y1,x2,y2)
  gradientFill(x1,y1,x2,y2, startcolor, stopcolor, direction) : Gradient fills a rectangle. Direction can be 0 or 1. 0=Vertical 1=Horizontal
  copyRect(dest_x1,dest_y1,dest_x2,dest_y2, sourceCanvas, source_x1,source_y1,source_x2,source_y2) : Draws an image from one source to another. Useful in cases of doublebuffering
  draw(x,y, graphic) : Draw the image of a specific Graphic class
  stretchDraw(rect, graphic): Draw the image of a specific Graphic class and stretch it so it fits in the given rectangle
  getClipRect() : Returns a table containing the fields Left, Top, Right and Bottom, which define the invalidated region of the graphical object. Use this to only render what needs to be rendered in the onPaint event of objects

Pen Class : (Inheritance: CustomPen->CanvasHelper->Object)
properties
  Color: Integer - The color of the pen
  Width: integer - Thickness of the pen
methods
  getColor()
  setColor(color)
  getWidth()
  setWidth(width)

Brush Class : (Inheritance: CustomBrush->CanvasHelper->Object)
properties
  Color : Integer
methods
  getColor()
  setColor()

Font Class : (Inheritance: CustomFont->CanvasHelper->Object)
createFont(): Returns a font object (default initialized based on the main ce window)

properties
  Name: string

```
  Size: integer
  Height: integer
  Orientation: integer
  Pitch: string - 'fpDefault', 'fpVariable', 'fpFixed'
  Color: integer
  CharSet: integer
  Quality: string - 'fqDefault', 'fqDraft', 'fqProof', 'fqNonAntialiased',
'fqAntialiased', 'fqCleartype', 'fqCleartypeNatural'
  Style: string set - ['fsBold', 'fsItalic', 'fsStrikeOut', 'fsUnderline']

methods
  getName(): Gets the fontname of the font
  setName(string): Sets the fontname of the font
  getSize(): Gets the size of the font
  setSize(integer): Sets the size of the font
  getColor(): Gets the color of the font
  setColor(integer): Sets the color of the font
  assign(font): Copies the contents of the font given as parameter to this font


Graphic Class : (Inheritance: Object) : Abstract class
properties
  Width: integer
  Height: integer
  Transparent: boolean

methods
  getWidth(graphic): Gets the current width in pixels of this graphics object
  setWidth(graphic, width): Sets thw width in pixels
  getHeight(graphic)
  setHeight(graphic, height)
  loadFromFile(filename)
  saveToFile(filename)

RasterImage class: (Inheritance: Graphic->Object) : Base class for some graphical
controls
properties
  Canvas: Canvas
  PixelFormat: PixelFormat - the pixelformat for this image. Will clear the current
image if it had one. Supported pixelformats: pf1bit, pf4bit, pf8bit, pf15bit,
pf16bit, pf24bit, pf32bit (recommended)
  TransparentColor: integer

methods
  getCanvas(): Returns the Canvas object for this image
  getPixelFormat():  Returns the current pixelformat
  getPixelFormat(pixelformat):  Sets the pixelformat for this image. Will clear the
current image if it had one. Supported pixelformats: pf1bit, pf4bit, pf8bit,
pf15bit, pf16bit, pf24bit, pf32bit (recommended)
  setTransparentColor(integer): Sets the color that will be rendered as transparent
```

when drawn
  getTransparentColor(): Returns the color set to be transparent
  saveToStream(stream) : Saves the image to a stream object
  loadFromStream(stream): Loads the image from a stream object

Bitmap class: (Inheritance: CustomBitmap->RasterImage->Graphic->Object) : Bitmap based Graphic object
createBitmap(width, height) - Returns a Bitmap object

PortableNetworkGraphic Class: (Inheritence: CustomBitmap->RasterImage->Graphic->Object)
createPNG(width, height) - Returns a PortableNetworkGraphic object

JpegImage Class: (Inheritence: CustomBitmap->RasterImage->Graphic->Object)
createJpeg(width, height) - Returns a Jpeg object

Icon Class: (Inheritence: CustomBitmap->RasterImage->Graphic->Object)
createIcon(width, height) - Returns an Icon object

Picture Class : (Inheritance: Object) : Container for the Graphic class
createPicture() : Returns a empty picture object

properties
  Graphic
  PNG
  Bitmap
  Jpeg
  Icon

methods
  loadFromFile(filename)
  saveToFile(filename)
  loadFromStream(stream, originalextension OPTIONAL) : Loads a picture from a stream. Note that the stream position must be set to the start of the picture
  assign(sourcepicture)

GenericHotkey Class : (Inheritance: Object)
createHotkey(function, keys, ...) : returns an initialized GenericHotkey class object. Maximum of 5 keys
createHotkey(function, {keys, ...}) : ^

properties
  DelayBetweenActivate: integer - Interval in milliseconds that determines the minimum time between hotkey activations. If 0, the global delay is used
  onHotkey: The function to call when the hotkey is pressed

```
methods
  getKeys()
  setKeys(key, ....)
  setOnHotkey(table)
  getOnHotkey


CommonDialog class: (Inheritance: CommonDialog->Component->Object)
  properties
    OnShow: function(sender)
    OnClose: function(sender)
    Title: string - The caption at top of the dialog
  methods
    Execute() : Shows the dialog and return true/false depending on the dialog

ColorDialog class:
createColorDialog(owner OPTIONAL) - Creates a new colordialog
properties
  property Color: integer - The currently selected color
  property CustomColors: TStrings - List of custom colors ( entry looks like ColorA
= FFFF00 ... ColorX = C0C0C0 )

methods

ColorBox class:
  Combobox like component where you can pick a color

createColorBox(owner) - Creates a new colorbox



FindDialog Class: (Inheritance: CommonDialog->Component->Object)
createFindDialog(owner)

properties
  Top
  Left
  Width
  Height
  FindText: String - The text the user wishes to find
  Options: Enum - Find Options
                   { frDown, frFindNext, frHideMatchCase, frHideWholeWord,
                     frHideUpDown, frMatchCase, frDisableMatchCase, frDisableUpDown,
                     frDisableWholeWord, frReplace, frReplaceAll, frWholeWord,
frShowHelp,
                     frEntireScope, frHideEntireScope, frPromptOnReplace,
frHidePromptOnReplace }
  OnFind: function (sender) - Called when the find button has been clicked
  OnHelp: function (sender) - Called when the help button is visible (see Options)
and clicked
```

methods

FileDialog Class: (Inheritance: CommonDialog->Component->Object)

properties

  DefaultExt: string - When not using filters this will be the default extention used if no extension is given
  Files: Strings - Stringlist containing all selected files if multiple files are selected
  FileName: string - The filename that was selected
  Filter: string - A filter formatted string
  FilterIndex: integer - The index of which filter to use

  InitialDir: string - Sets the folder the filedialog will show first
methods

OpenDialog Class: (Inheritance: FileDialog->CommonDialog->Component->Object)
createOpenDialog(owner) : Creates an opendialog object

properties
  Options: String
    A string formatted as "[param1, param2, param3]" to set OpenDialogs options
    Valid parameters are:
      ofReadOnly,
      ofOverwritePrompt : if selected file exists shows a message, that file will be overwritten
      ofHideReadOnly   : hide read only file
      ofNoChangeDir   : do not change current directory
      ofShowHelp   : show a help button
      ofNoValidate
      ofAllowMultiSelect : allow multiselection
      ofExtensionDifferent
      ofPathMustExist   : shows an error message if selected path does not exist
      ofFileMustExist   : shows an error message if selected file does not exist
      ofCreatePrompt
      ofShareAware
      ofNoReadOnlyReturn : do not return filenames that are readonly
      ofNoTestFileCreate
      ofNoNetworkButton
      ofNoLongNames
      ofOldStyleDialog
      ofNoDereferenceLinks : do not expand filenames
      ofEnableIncludeNotify
      ofEnableSizing   : dialog can be resized, e.g. via the mouse
      ofDontAddToRecent : do not add the path to the history list
      ofForceShowHidden : show hidden files

```
      ofViewDetail       : details are OS and interface dependent
      ofAutoPreview      : details are OS and interface dependent


methods
-


SaveDialog Class: (Inheritance:
OpenDialog->FileDialog->CommonDialog->Component->Object)
createSaveDialog(owner)

SelectDirectoryDialog Class: (Inheritance:
OpenDialog->FileDialog->CommonDialog->Component->Object)
createSelectDirectoryDialog(owner)


Stream Class

properties
  Size: integer
  Position: integer

methods
  copyFrom(stream, count) - Copies count bytes from the given stream to this stream
  read(count): bytetable - Returns a bytetable containing the bytes of the stream.
This increases the position
  write(bytetable, count OPTIONAL)- Writes the given bytetable to the stream
  readByte(): integer
  writeByte(integer)
  readWord(): integer
  writeWord(integer)
  readDword(): integer
  writeDword(integer)
  readQword(): integer
  writeQword(integer)
  readString(stringlengthinbytes) : Reads a given stringcount
  writeString(string, include0terminator: boolean OPTIONAL=false)
  readAnsiString(): string - Reads a string that has been written with
writeAnsiString (the length of the string is part of the data)
  writeAnsiString(string)


MemoryStream Class (Inheritance: Stream->Object)
createMemoryStream()

properties
  Memory: Integer - The address in Cheat Engine's memory this stream is loaded
(READONLY, tends to change on size change)
```

methods
  loadFromFile(filename) : Replaces the contents in the memory stream with the contents of a file on disk
  saveToFile(filename) : Writes the contents of the memory stream to the specified file
  loadFromFileNoError(filename):boolean,string - Replaces the contents in the memory stream with the contents of a file on disk. On success returns true, else false with a secondary return the error message
  saveToFileNoError(filename):boolean,string - Writes the contents of the memory stream to the specified file. On success returns true, else false with a secondary return the error message
  clear(): sets the size to 0

FileStream Class (Inheritance: HandleStream->Stream->Object)
createFileStream(filename, mode) -
  Creates a filestream object. mode can be fmCreate(0xff00), fmOpenRead, fmOpenWrite or fmOpenReadWrite and can be or-ed with
  fmShareCompat(0x0000), fmShareExclusive(0x0010), fmShareDenyWrite(0x0020), fmShareDenyRead(0x0030) or fmShareDenyNone(0x0040)

StringStream Class (Inheritance: Stream->Object)
createStringStream(string)

properties
DataString: The internal string

TableFile class (Inheritance: Object)
findTableFile(filename): Returns the TableFile class object for the saved file
createTableFile(filename, filepath OPTIONAL): TableFile - Add a new file to your table. If no filepath is specified, it will create a blank file. Otherwise, it will read the contents from disk.

properties
  Name: string
  Stream: MemoryStream
  DoNotSave: boolean

methods
  delete() : Deletes this file from your table.
  saveToFile(filename)
  getData() : Gets a MemoryStream object

xmplayer class
The xmplayer class has already been defined as xmplayer, no need to create it manually

properties
  IsPlaying : boolean - Indicator that the xmplayer is currently playing a xm file
  Initialized: boolean - Indicator that the xmplayer is actually actively loaded in memory

methods
  setVolume(int)
  playXM(filename, OPTIONAL noloop)
  playXM(tablefile, OPTIONAL noloop)
  playXM(Stream, OPTIONAL noloop)
  pause()
  resume()
  stop()

CheatComponent Class: (Inheritance: WinControl->Control->Component->Object)
The cheatcomponent class is the component used in Cheat Engine 5.x trainers
Most people will probably want to design their own components but for those that don't know much coding and use the autogenerated trainer this will be used

properties
  Color: Integer - background color
  Textcolor: integer - text color
  Activationcolor: integer - The textcolor to show when activated is true
  Activated: boolean - Toggles between the ActivationColor and the TextColor
  Editleft:integer - The x position of the optional edit field
  Editwidth: integer - the width of the optional edit field
  Editvalue:string - The string of the optional edit field
  Hotkey:string read - The hotkeypart of the cheat line
  Description:string - Description part of the cheat line
  Hotkeyleft: integer - The x position of the hotkey line
  Descriptionleft:integer - The x position of the Description line

  ShowHotkey: boolean - Decides if the hotkey label should be shown
  HasEditBox: boolean - Decides if the editbox should be shown
  HasCheckbox: boolean - Decides if the checkbox should be shown
  Font: Font - The font to use to render the text

methods
 -

MemoryRecordHotkey Class: (Inheritance: object)
The memoryrecord hotkey class is mainly readonly with the exception of the event properties to be used to automatically create trainers
Use the generic hotkey class if you wish to create your own hotkeys

properties

```
  Owner: MemoryRecord - The memoryrecord this hotkey belongs to (ReadOnly)
  Keys: Table - Table containing the keys(combination) for this hotkey
  action: integer - The action that should happen when this hotkey triggers
      mrhToggleActivation(0): Toggles between active/deactive
      mrhToggleActivationAllowIncrease(1): Toggles between active/deactive. Allows
increase when active
      mrhToggleActivationAllowDecrease(2): Toggles between active/deactive. Allows
decrease when active
      mrhActivate(3): Sets the state to active
      mrhDeactivate(4):  Sets the state to deactive
      mrhSetValue(5):  Sets a specific value to the value properyy (see value)
      mrhIncreaseValue(6):  Increases the current value with the value property (see
value)
      mrhDecreaseValue(7):  Decreases the current value with the value property (see
value)
  value: string - Value used depending on what kind of hotkey is used
  ID: integer - Unique id of this hotkey (ReadOnly)
  Active: boolean - True if it's hotkey will be handled, false if this hotkey is
ignored
  Description: string - The description of this hotkey
  HotkeyString: string - The hotkey formatted as a string (ReadOnly)
  ActivateSound: string - Tablefile name of a WAV file inside the table which will
get played on activate events
  DeactivateSound: string - Tablefile name of a .WAV file inside the table which
will get played on deactivate events
  OnHotkey: function(sender) - Function to be called when a hotkey has just been
pressed
  OnPostHotkey: function(sender) - Function to be called when a hotkey has been
pressed and the action has been performed


methods
  doHotkey: Executes the hotkey as if it got triggered by the keyboard


MemoryRecord Class:
The memoryrecord objects are the entries you see in the addresslist

properties
  ID: Integer - Unique ID
  Index: Integer - The index ID for this record. 0 is top. (ReadOnly)
  Description: string- The description of the memory record
  Address: string - Get/set the interpretable address string. Useful for simple
address settings.
  AddressString: string - Get the address string shown in CE (ReadOnly)
  OffsetCount: integer - The number of offsets. Set to 0 for a normal address
  Offset[] : integer - Array to access each offset
  OffsetText[] : string - Array to access each offset using the interpretable text
style
```

CurrentAddress: integer - The address the memoryrecord points to
VarType: ValueType (string) - The variable type of this record. See vtByte to vtCustom
Type: ValueType (number) - The variable type of this record. See vtByte to vtCustom
If the type is vtString then the following properties are available:
String.Size: Number of characters in the string
String.Unicode: boolean
String.Codepage: boolean

If the type is vtBinary then the following properties are available
Binary.Startbit: First bit to start reading from
Binary.Size : Number of bits

If the type is vtByteArray then the following properties are available
Aob.Size : Number of bytes

CustomTypeName: String - If the type is vtCustom this will contain the name of the CustomType
Script: String - If the type is vtAutoAssembler this will contain the auto assembler script
Value: string - The value in stringform.
NumericalValue: number - The value in numerical form. nil if it can not be parsed to a number
Selected: boolean - Set to true if selected (ReadOnly)
Active: boolean - Set to true to activate/freeze, false to deactivate/unfreeze
Color: integer
ShowAsHex: boolean - Self explanatory
ShowAsSigned: boolean - Self explanatory
AllowIncrease: boolean - Allow value increasing, unfreeze will reset it to false
AllowDecrease: boolean - Allow value decreasing, unfreeze will reset it to false
Collapsed: boolean - Set to true to collapse this record or false to expand it. Use expand/collapse methods for recursive operations.
IsGroupHeader: boolean - Set to true if the record was created as a Group Header with no address or value info.
IsAddressGroupHeader: boolean - Set to true if the record was created as a Group Header with address.
IsReadable: boolean - Set to false if record contains an unreadable address. NOTE: This property will not be set until the value property is accessed at least once. (ReadOnly)
Selected: boolean
Options: String set - a string enclosed by square brackets filled with the options seperated by a comma. Valid options are: moHideChildren, moActivateChildrenAsWell, moDeactivateChildrenAsWell, moRecursiveSetValue, moAllowManualCollapseAndExpand, moManualExpandCollapse, moAlwaysHideChildren

DropDownLinked: boolean - if dropdown list refers to list of another memory record eg. (memrec name)
DropDownLinkedMemrec: string - Description of linked memrec or emptystring if not linked

DropDownList : StringList - list of "value:description" lines, lists are still separate objects when linked, read-write
DropDownReadOnly: boolean - true if 'Disallow manual user input' is set
DropDownDescriptionOnly: boolean - self explanatory
DisplayAsDropDownListItem: boolean - self explanatory
DropDownCount: integer - equivalent to .DropDownList.Count
DropDownValue[index] : Array to access values in DropDownList (ReadOnly)
DropDownDescription[index] : Array to access Descriptions in DropDownList (ReadOnly)

Count: Number of children
Child[index] : Array to access the child records
[index] = Child[index]
Parent: MemoryRecord - The parent of the memory record

HotkeyCount: integer - Number of hotkeys attached to this memory record
Hotkey[] : Array to index the hotkeys

Async: Boolean - Set to true if activating this entry will be asynchronious. (only for AA/Lua scripts)
AsyncProcessing: Boolean - True when async is true and it's being processed
AsyncProcessingTime: qword - The time that it has been processing in milliseconds

HasMouseOver: boolean - True if the mouse is currently over it

OnActivate: function(memoryrecord,before,currentstate):boolean - The function to call when the memoryrecord will change (or changed) Active to true. If before is true, not returning true will cause the activation to stop.
OnDeactivate: function(memoryrecord,before,currentstate):boolean - The function to call when the memoryrecord will change (or changed) Active to false. If before is true, not returning true will cause the deactivation to stop.
OnActivationFailure: function(memoryrecord,reason,reasonText) - Called when activating a record fails. You can use this to inform the user of an issue, or adjust the table/script and try again. Return true when you wish to try again. Warning: Watch out for infinite loops...
reason can be: afInaccessible, afGenericAutoAssembler, afAllocateFailure, afSyntaxError, afSyntaxErrorInLua, afStructureDefinitionError, afAssertFailure, afAOBModuleNotFound, afAOBNotFound, afIncludeNotFound, afDLLInjectionFailure
reasonText is a description provided by cheat engine

OnDestroy: function() - Called when the memoryrecord is destroyed.
OnGetDisplayValue: function(memoryrecord,valuestring):boolean,string - This function gets called when rendering the value of a memory record. Return true and a new string to override the value shown
OnValueChanged: function(memoryrecord, oldvalue, newvalue): This function gets called whenever the value of a memory record has changed

  OnValueChangedByUser: function(memoryrecord, oldvalue, newvalue):This function gets called whenever the value of a memory record has changed by the user
  DontSave: boolean - Don't save this memoryrecord and it's children

methods
  getDescription()
  setDescription()
  getAddress() : Returns the interpretable addressstring of this record. If it is a pointer, it returns a second result as a table filled with the offsets
  setAddress(string) : Sets the interpretable address string, and if offsets are provided make it a pointer

  getOffsetCount(): Returns the number of offsets for this memoryrecord
  setOffsetCount(integer): Lets you set the number of offsets

  getOffset(index) : Gets the offset at the given index
  setOffset(index, value) : Sets the offset at the given index

  getCurrentAddress(): Returns the current address as an integer (the final result of the interpretable address and pointer offsets)

  appendToEntry(memrec): Appends the current memory record to the given memory record

  getHotkey(index): Returns the hotkey from the hotkey array
  getHotkeyByID(integer): Returns the hotkey with the given id

  reinterpret()
  createHotkey({keys}, action, value OPTIONAL, description OPTIONAL): Returns a hotkey object

  disableWithoutExecute(): Sets the entry to disabled without executing the disable section
  beginEdit() : Call when you wish to take a long time to edit a record. (e.g external editor) It prevents the record from getting deleted
  endEdit() : to mark the end of your long edit sequence

global events
  function onMemRecPreExecute(memoryrecord, newstate BOOLEAN):
    If above function is defined it will be called before action* has been performed.
    Active property is about to change to newState.

  function onMemRecPostExecute(memoryrecord, newState BOOLEAN, succeeded BOOLEAN):
    If above function is defined it will be called after action*.
    Active property was supposed to change to newState.
    If 'succeeded' is true it means that Active state has changed and is newState.

    newState and succeeded are read only.

*action can be: running auto assembler script (ENABLE or DISABLE section), freezing and unfreezing.

Addresslist Class: (Inheritance: Panel->WinControl->Control->Component->Object)
properties
  LoadedTableVersion: integer - Returns the tableVersion of the last loaded table
  Count: Integer - The number of records in the table
  SelCount: integer- The number of records that are selected
  SelectedRecord: MemoryRecord - The main selected record
  MemoryRecord[]: MemoryRecord - Array to access the individial memory records
  List: The internal Treeview control of the addresslist
  [] = MemoryRecord - Default accessor

  CheckboxActiveSelectedColor: color
  CheckboxActiveColor: color
  CheckboxSelectedColor: color
  CheckboxColor: color
  SelectedBackgroundColor: color
  SelectedSecondaryBackgroundColor: color
  ExpandSignColor: color
  IncreaseArrowColor: color
  DecreaseArrowColor: color

  MouseHighlightedRecord() : Returns the memoryrecord that the mouse points at. nil if nothing

  OnDescriptionChange: function(addresslist,memrec):boolean - called when the user initiates a description column change on a record. Return true if you handle it, false for normal behaviour
  OnAddressChange: function(addresslist,memrec):boolean - called when the user initiates an address column change on a record. Return true if you handle it, false for normal behaviour
  OnTypeChange: function(addresslist,memrec):boolean - called when the user initiates a type column change on a record. Return true if you handle it, false for normal behaviour
  OnValueChange: function(addresslist,memrec):boolean - called when the user initiates a value column change on a record. Return true if you handle it, false for normal behaviour
  OnAutoAssemblerEdit: function(addresslist,memrec) - Called when the user initiates a memoryrecord AA script edit. This function will be responsible for changing the memory record

methods
  getCount()
  getMemoryRecord(index)
  getMemoryRecordByDescription(description): returns a Memory Record object
  getMemoryRecordsWithDescription(description): returns a table with MemoryRecords

```
that have the same description (slower than when using unique names)
  getMemoryRecordByID(ID)
  createMemoryRecord() : creates an generic cheat table entry and add it to the list

  getSelectedRecords():  Returns a table containing all the selected records

  doDescriptionChange() : Will show the GUI window to change the description of the
selected entry
  doAddressChange() : Will show the GUI window to change the address of the selected
entry
  doTypeChange() : Will show the GUI window to change the type of the selected
entries
  doValueChange() : Will show the GUI window to change the value of the selected
entries

  getSelectedRecord() : Gets the main selected memoryrecord
  setSelectedRecord(memrec) : Sets the currently selected memoryrecord. This will
unselect all other entries

  disableAllWithoutExecute(): Disables all memory records without executing their
[Disable] section
  rebuildDescriptionCache(): Rebuilds the description to record lookup table


MemScan Class (Inheritance: Object)
getCurrentMemscan() : Returns the current memory scan object. If tabs are used the
current tab's memscan object
createMemScan(progressbar OPTIONAL) : Returns a new MemScan class object

setSpecialScanOptionsOverride({}): Sets certain scan options that are usually only
set in settings
  options are:
    MEM_PRIVATE: boolean - Scan memory owned by just the target process
    MEM_IMAGE: boolean - scan memory belonging to modules
    MEM_MAPPED: boolean - scan memory of files mapped in memory and accessed on
demand

  Not setting an entry will make them revert back to the original state if they
where previously set

properties
  LastScanWasRegionScan: boolean - returns true is the previous scan was an unknown
initial value
  LastScanValue: string
  LastScanType: ScanType/string - 'stNewScan', 'st v   ', 'stNextScan'
  ScanresultFolder: string - Path where the results are stored(READONLY)
  OnScanDone: function(memscan) - Set a function to be called when the scan has
finished
  OnGuiUpdate: function(memscan, TotalAddressesToScan, CurrentlyScanned,
ResultsFound) - Called during the scan so you can update the interface if needed
```

```
  FoundList: FoundList - The foundlist currently attached to this memscan object
  OnlyOneResult: boolean - If this is set to true memscan will stop scanning after
having found the first result, and written the address to "Result"
  IsUnique: boolean - Same as OnlyOneResult but will use multiple threads, so if the
value is not unique you will be given a random address
  Result: Integer - If OnlyOneResult is used this will contain the address after a
scan has finished

  CodePage: boolean;
  ScanOption: TScanoption
  VariableType: TVariableType
  VarType: TVariableType : ^
  Roundingtype: TRoundingType
  Scanvalue: string : Value to scan
  Scanvalue1: string : ^
  Scanvalue2: string : Secondary value to scan (e.g value between scan)
  Startaddress: integer
  Stopaddress: integer
  Hexadecimal: boolean
  BinaryStringAsDecimal: boolean
  UTF16: boolean
  Casesensitive: boolean
  Fastscanmethod: TFastScanMethod
  Fastscanparameter: string
  Customtype: TCustomType

  ScanWritable:    TScanregionpreference ('scanDontCare', 'scanExclude',
'scanInclude')
  ScanExecutable:  TScanregionpreference ('scanDontCare', 'scanExclude',
'scanInclude')
  ScanCopyOnWrite: TScanregionpreference ('scanDontCare', 'scanExclude',
'scanInclude')

  Percentage: boolean
  CompareToSavedScan: boolean
  SavedScanName: string

methods
  scan(): Does either a first scan or next scan based on the given property values
  firstScan() : Does a first scan based on the given property values
  nextScan() : Does a next scan based on the given property values
  newScan() : Clears the current results

  firstScan(scanoption, vartype, roundingtype, input1, input2 ,startAddress
,stopAddress ,protectionflags ,alignmenttype ,"alignmentparam" ,isHexadecimalInput
,isNotABinaryString, isunicodescan, iscasesensitive);
    Does an initial scan.
    memscan: The MemScan object created with createMemScan
    scanOption: Defines what type of scan is done. Valid values for firstscan are:
      soUnknownValue: Unknown initial value scan
```

```
      soExactValue: Exact Value scan
      soValueBetween: Value between scan
      soBiggerThan: Bigger than ... scan
      soSmallerThan: smaller than ... scan

    vartype: Defines the variable type. Valid variable types are:
      vtByte
      vtWord  2 bytes
      vtDword 4 bytes
      vtQword 8 bytes
      vtSingle float
      vtDouble
      vtString
      vtByteArray
      vtGrouped
      vtBinary
      vtAll

    roundingtype: Defined the way scans for exact value floating points are handled
      rtRounded : Normal rounded scans. If exact value = "3" then it includes 3.0 to
3.49999999. If exact value is "3.0" it includes 3.00 to 3.0499999999
      rtTruncated: Truncated algorithm. If exact value = "3" then it includes 3.0 to
3.99999999. If exact value is "3.0" it includes 3.00 to 3.099999999
      rtExtremerounded: Rounded Extreme. If exact value = "3" then it includes
2.0000001 to 3.99999999. If exact value is "3.0" it includes 2.900000001 to
3.099999999

    input1: If required by the scanoption this is a string of the given variable
type
    input2: If requires by the scanoption this is the secondary input

    startAddress : The start address to scan from. You want to set this to 0
    stopAddress  : The address the scan should stop at. (You want to set this to
0xffffffffffffffff)

    protectionflags : See aobscan about protectionflags
    alignmenttype : Scan alignment type. Valid options are:
      fsmNotAligned : No alignment check
      fsmAligned    : The address must be dividable by the value in alignmentparam
      fsmLastDigits : The last digits of the address must end with the digits
provided by alignmentparam

    alignmentparam : String that holds the alignment parameter.

    isHexadecimalInput: When true this will handle the input field as a hexadecimal
string else decimal
    isNotABinaryString: When true and the varType is vtBinary this will handle the
input field as a decimal instead of a binary string
    isunicodescan: When true and the vartype is vtString this will do a unicode
(utf16) string scan else normal utf8 string
```

    iscasesensitive : When true and the vartype is vtString this check if the case matches

  nextScan(scanoption, roundingtype, input1,input2, isHexadecimalInput, isNotABinaryString, isunicodescan, iscasesensitive, ispercentagescan, savedresultname OPTIONAL);
    Does a next scan based on the current addresslist and values of the previous scan or values of a saved scan
    memscan: The MemScan object that has previously done a first scan
    scanoption:
      soExactValue: Exact Value scan
      soValueBetween: Value between scan
      soBiggerThan: Bigger than ... scan
      soSmallerThan: smaller than ... scan
      soIncreasedValue: Increased value scan
      soIncreasedValueBy: Increased value by scan
      soDecreasedValue: Decreased value scan
      soDecreasedValueBy: Decreased value by scan
      soChanged: Changed value scan
      soUnchanged: Unchanged value scan

    roundingtype: Defined the way scans for exact value floating points are handled
      rtRounded : Normal rounded scans. If exact value = "3" then it includes 3.0 to 3.49999999. If exact value is "3.0" it includes 3.00 to 3.0499999999
      rtTruncated: Truncated algoritm. If exact value = "3" then it includes 3.0 to 3.99999999. If exact value is "3.0" it includes 3.00 to 3.099999999
      rtExtremerounded: Rounded Extreme. If exact value = "3" then it includes 2.0000001 to 3.99999999. If exact value is "3.0" it includes 2.900000001 to 3.099999999

    input1: If required by the scanoption this is a string of the given variable type
    input2: If requires by the scanoption this is the secondary input

    isHexadecimalInput: When true this will handle the input field as a hexadecimal string else decimal
    isNotABinaryString: When true and the varType is vtBinary this will handle the input field as a decimal instead of a binary string
    isunicodescan: When true and the vartype is vtString this will do a unicode (utf16) string scan else normal utf8 string
    iscasesensitive : When true and the vartype is vtString this check if the case matches
    ispercentage: When true and the scanoption is of type soValueBetween, soIncreasedValueBy or soDecreasedValueBy will cause CE to do a precentage scan instead of a normal value scan
    savedResultName: String that holds the name of a saved result list that should be compared against. First scan is called "FIRST"

waitTillDone() : Waits for the memscan thread(s) to finish scanning. Always use this
saveCurrentResults(name) : Save the current scanresults to a unique name for this memscan. This save can be used to compare against in a subsequent next scan
getSavedResultList(): Returns an indexed table of all the saved results (strings)
getSavedResultHandler(name): Gets the 'SavedResultHandler' object with this name (nil on failure)
getAttachedFoundlist() : Returns a FoundList object if one is attached to this scanresults. Returns nil otherwise

setOnlyOneResult(state): If set to true before you start a scan, this will cause the scanner to only return one result. Note that it does not work with a foundlist
getOnlyResult(): Only works if returnOnlyOneResult is true. Returns nil if not found, else returns the address that was found (integer)
getProgress(): returns a table with fields: TotalAddressesToScan, CurrentlyScanned, ResultsFound

FoundList Class
The foundlist is an object that opens the current memscan's result file and provides an interface for reading out the addresses

createFoundList(memscan)

properties
Count: integer - Number of results found
Address[index] : string - Returns the address as a string at the given index (index starts at 0)
Value[index]: string - Returns the value as a string at the given index (index starts at 0)
[index]: string - Same as Address (index starts at 0)

methods
initialize() : Call this when a memscan has finished scanning. This will open the results for reading
deinitialize() : Release the results
getCount()
getAddress(index) : Returns the address as a string
getValue(index) : Returns the value as a string

SavedResultHandler:
properties
methods
getStringFromAddress(address, hexadecimal:boolean OPTIONAL, signed: boolean OPTIONAL) : Returns the value in string format at the given address

```
Memoryview class: (Inheritance:
Form->ScrollingWinControl->CustomControl->WinControl->Control->Component->Object)
createMemoryView() - Creates a new memoryview window. This window will not receive
debug events. Use getMemoryViewForm() function to get the main memoryview window
properties
  DisassemblerView: The disassemblerview class of this memoryview object
  HexadecimalView: The hexadecimalview class of this memoryview object
methods
  -

DisassemblerviewLine class: (Inheritance: Object)
properties
  Address: The current address of this line
  Owner: The Disassemblerview that owns this line
  OnDisassemblerViewOverride: Called when a line is about to be rendered
    funtion(address, addressstring, bytestring, opcodestring, parameterstring,
specialstring)
      return addressstring, bytestring, opcodestring, parameterstring, specialstring
    end
  History: OrderedList - Holds the history navigation

methods
  -

Disassemblerview class: (Inheritance:
Panel->CustomControl->WinControl->Control->Component->Object)
  The visual disassembler used on the memory view window
properties
  SelectedAddress: integer - The currently selected address in the disassemblerview
  SelectedAddress2: integer - The secondary selected address in the disassemblerview
  SelectionSize: integer - The size of the selected area
  TopAddress: Integer - The first address to show
  ShowJumplines: boolean - Determines if the jumplines should be shown
  HideFocusRect: boolean - If set to true the focus rectangle won't be shown
  SpaceAboveLines: integer
  SpaceBelowLines: integer
  OnSelectionChange: function(sender, address, address2) - Function to call when the
selection has changed
  OnExtraLineRender: function(sender, Address, AboveInstruction, Selected):
RasterImage OPTIONAL, x OPTIONAL, y OPTIONAL
    Function to call when you wish to provide the disassembler view with an extra
image containing data you wish to show.
    This function is called once to get an image to show above the instruction, and
once to get an image to show under the instruction and optional comments.
    The image for both calls must be different objects as rendering will only be
done when both calls have been completed

    Sender is a DisassemblerviewLine object
    If no coordinates are given the image will be centered above/below the
```

instruction

  Osb: Bitmap : Background picture of the disasemblerview

methods
  -

Hexadecimal class: (Inheritance:
Panel->CustomControl->WinControl->Control->Component->Object)
  The visual hexadecimal object used on the memory view window
properties
  Address: integer - The top address
  HasSelection: boolean - True if something is selected
  SelectionStart: integer
  SelectionStop: integer
  History: OrderedList
  DisplayType: displaytype - The type being shown: dtByte, dtByteDec, dtWord,
dtWordDec, dtDword, dtDwordDec, dtQword, dtQwordDec, dtSingle, dtDouble
  OnAddressChange(hexadecimalview, function): function(hexadecimalview, address)
  OnByteSelect(hexadecimalview, function): function(hexadecimalview, address,
address2)
  OnCharacterRender: function(sender, address, text) return text end - Called when a
character is being rendered. Use this to change it or change the canvas fonts
(Warning: slow)
  OnValueRender: function(sender, address, text) return text end - Called when a
value (depending on the displaytype) is being rendered. Use this to change it or
change the canvas fonts (Warning: slow)

methods
  -

Thread Class: (Inheritance: Object)
createThread(function(Thread,...), ...) :
  Executes the given function in another thread using the systems thread mechanism
  The function returns the Thread class object
  function declaration: function (Thread, ...)

createThreadSuspended(function(Thread,...), ...) :
  Same as createNativeThread but it won't run until resume is called on it

createThreadNewState(scripttext): Creates a new thread in a new lua state. This is
more efficient as no locking inside lua takes place, but has no access to
userdefined lua functions and only limited base CE functions. called inside a
function(t) where t is the thread. Watch for t.Terminated to quit. Notice: Unlike
createThread, the created thread does not freeOnTerminate so you can read out the
"Result" property which will be set on thread finish

getCurrentThreadObject() : Returns a thread object for the current thread (7.6+)

properties
  OnDestroy: function(thread) - Called when a thread is about to be freed (Can be by itself, so can run in the same context as the thread) 7.6+
  Name: string - This name will be shown when the thread terminated abnormally
  Finished: boolean - Returns true if the thread has reached the end. Do not rely on this if the thread is freeOnTerminate(true) (which is the default)
  Terminated: boolean - Returns true if the Terminate method has been called
  Result: string - The result of the thread function as a string

methods
  freeOnTerminate(state) :
    When set to true the thread object will free itself when the function ends (default=true)
    Note: Use this only from inside the thread function as the thread might have already terminated and freed itself when called

  synchronize(function(thread, ...), ...) :
    Called from inside the thread. This wil cause the tread to get the main thread to execute the given function and wait for it to finish.
    Usually for GUI access
    Returns the return value of the given function

  waitfor() :
    Waits for the given thread to finish (Not recommended to call this from inside the thread itself)

  suspend() :
    Suspend the thread's execution

  resume() :
    Resume the thread's execution

  terminate() :
    Tells the thread it should terminate. The Terminated property will become true

CriticalSection class: (Inheritance: Object)
createCriticalSection(): Returns a critical section object

properties
-

methods
  enter()
  leave()
  tryEnter(): Returns true if entered, false if not

Event class: (Inheritance: Object)
createEvent(ManualReset, InitialState): Returns an event object

properties
-

methods
  resetEvent()
  setEvent()
  waitFor(timeout): Waits for the event to be set. Returns wrSignaled(0), wrTimeout(1), wrAbandoned(2) or wrError(3);

Semaphore class: (Inheritance: Object)
createSemaphore(count): Returns an semaphore object
properties
-

methods
  acquire()
  release()

MultiReadExclusiveWriteSynchronizer class: (Inheritance: Object)
createMultiReadExclusiveWriteSynchronizer(): Returns a createMultiReadExclusiveWriteSynchronizer

properties
-

methods
  beginWrite()
  endWrite()
  beginRead()
  endRead()

StructureFrm class:
createStructureForm(address, groupname OPTIONAL, structurename OPTIONAL)
enumStructureForms() : returns a table of StructureFrm objects (can be useful for finding a structure window with the wanted structure)

properties:
MainStruct: structure - The currently selected structure

ColumnCount: integer - the number of columns (columns=address)

Column[index]: structColumn - Fetches a structColumn object from the structure form

GroupCount: integer - The number of groups
Group[index]: structGroup - Fetches a structGroup object from the structure form
OnStatusbarUpdate(function(notificationbar)) - Called when the statusbar gets updated

methods:
structChange() : Forces a refresh
addColumn(): Adds a new column in the currently focuses group and returns it's structColumn object
addGroup(): Adds a new group and returns the structGroup object
getSelectedStructElement(): Returns the currently selected StructureElement , and as second result a table containing an indexed list continaing 'struct', and 'element' describing the path to the base

structColumn class:
properties:
Address: integer - The current address
AddressText: string - Gets/sets the visual address
Focused: boolean - Gets/sets the focused state

methods:
focus(): focuses the current column

structGroup class:
properties:
name: string - gets the current name
box: Groupbox - Gets the groupbox object
columnCount: integer- Gets the number of columns in the group
columns[index]: structColumn - Returns the specific structColumn object

methods:
addColumns(): Adds a new columns to the specific group and returns it's structColumn objecy

Structure class related functions:
getStructureCount(): Returns the number of Global structures. (Global structures are the visible structures)
getStructure(index): Returns the Structure object at the given index
createStructure(name): Returns an empty structure object (Not yet added to the Global list. Call structure.addToGlobalStructureList manually)

createStructureFromName(string): If PDB files are loaded this will create a structure with that name if it can be found

structure class: (Inheritance: Object)
Properties:
  Name: String - The name of the structure
  Size: Integer - The number of bytes between the last element and the start. ReadOnly
  Count: Integer - Number of elements in the structure. ReadOnly
  Element[]: structureElement - Returns the structure element at the given index. Readonly
  [] = Element[]
Methods:
  getName(): Returns the name
  setName(name): Sets the name
  getElement(index): Returns a structureElement object (Changing offsets can change the index)
  getElementByOffset(offset): Returns a structureElement object where the specified offset is at least the requested offset
  addElement(): Adds a new blank structureElement and returns it
  autoGuess(baseaddresstoguessfrom, offset, size)
  fillFromDotNetAddress(address, changeName): Fills the structure with the layout gathered from querying .NET. If changeName is true, the structure will take the name of the .NET class. (6.4+)

  beginUpdate(): Call this when you want to make multiple updates to a structure. It will speed up the update process
  endUpdate(): Call this when done
  addToGlobalStructureList(): Add this to the list of structures for the user to select from. (Global structures will get saved to the table)
  removeFromGlobalStructureList(): Remove from the list of structures.

StructureElement class: (Inheritance: Object)
Properties:
  Owner: structure - The structure this element belongs to. Readonly
  Offset: integer - The offset of this element
  Name: string - The name of this element
  Vartype: integer - The variable type of this element
  CustomType: CustomType - if Vartype is vtCustom this is the associated custom type
  CustomTypeName: string - The name of the custom type. Empty string if no custom type
  DisplayMethod: string - The displaymethod of the entry : 'dtUnsignedInteger', 'dtSignedInteger' or 'dtHexadecimal'
  ChildStruct: structure - If not nil this element is a pointer to the structure defined here
  ChildStructStart: integer - The number of bytes inside the provided childstruct. (E.g: It might point to offset 10 of a certain structure)

  Bytesize: integer - The number of bytes of this element. Readonly for basic types, writable for types that require a defined length like strings and array of bytes
  BackgroundColor: integer - The background color of the element

Methods:
  getOwnerStructure(): Returns the structure this element belongs to
  getOffset(): Returns the offset of this element
  setOffset(offset): Sets the offset of this element
  getName(): Returns the name of this element
  setName(name): Sets the name of this element (tip: Leave blank if you only want to set the name of the variable)
  getVartype(): Returns the variable type of this element (check Variable types in defines.lua)
  setVartype(vartype)
  getValue(address) : Gets the memory from the specified address and interprets it according to the element type
  setValue(address,value): Sets the memory at the specified address to the interpreted value according to the element type
  getValueFromBase(baseaddress): same as getValue but uses the offset to calculate the final address
  setValueFromBase(baseaddress,value): same as setValue but uses the offset to calculate the final address
  getChildStruct()
  setChildStruct(structure)
  getChildStructStart()
  setChildStructStart(offset)
  getBytesize(): Gets the bytesize of the element. Usually returns the size of the type, except for string and aob
  setBytesize(size): sets the bytesize for types that are affected (string, aob)

supportCheatEngine(attachwindow, hasclosebutton, width, height, position ,yoururl OPTIONAL, extraparameters OPTIONAL, percentageshown OPTIONAL):
  Will show an advertising window which will help keep the development of Cheat Engine going.
  If you provide your own url it will be shown Up to 75% of the time.

  attachwindow: Type=Form : The form that the ad is attached to
  hasclosebutton: Type=boolean : If true the window will have a border an a close button at top
  width, height: Type=integer :
    The client width and height of the window.
    Prefered formats are : 120x600 , 160x600, 300x250, 468x60, 728x90 ,But you are free to use different formats

  Position: Type=integer/enum: The place of the window
    0=Top, 1=Right, 2=Bottom, 3=left

Yoururl: Type=string: The url you want to show. When given instead of showing CE's ads 100% it will show your url up to 75%.
You can use it for your own income, or for updating users about new versions of your trainer or whatever you feel like

Extraparameters: Type=String : are url request parameters you can add to the default parameters (e.g trainername=mytrainer for tracking purposes)

PercentageShown: You can change the default of 75% to a smaller value like 50%

fuckCheatEngine() : Removes the ad window if it was showing

Following are some more internal functions for Cheat Engine

dbk_initialize() : Returns true if the dbk driver is loaded in memory. False if it failed for whatever reason (e.g 64-bit and not booted with unsigned driver support)
dbk_initialized() : Returns true if the dbk driver is loaded in memory and available to CE. Does not try to load the driver
dbk_useKernelmodeOpenProcess() : Switches the internal pointer of the OpenProcess api to dbk_OpenProcess
dbk_useKernelmodeProcessMemoryAccess() : Switches the internal pointer to the ReadProcessMemory and WriteProcessMemory apis to dbk_ReadProcessMemory and dbk_WriteProcessMemory
dbk_useKernelmodeQueryMemoryRegions() : Switches the internal pointer to the QueryVirtualMemory api to dbk_QueryVirtualMemory
dbk_usePhysicalMemoryAccess() : Changes all memory access to physical memory (will use dbvm if available. Won't load dbk if not)
dbk_setSaferPhysicalMemoryScanning(state: BOOLEAN): When set to true CE's memory scanner will skip hardware device owned memory. Default state is true
dbk_readPhysicalMemory(address, size): bytetable
dbk_writePhysicalMemory(address, size): boolean
dbk_getPEProcess(processid) : Returns the pointer of the EProcess structure of the selected processid
dbk_getPEThread(threadid) : Gets the pointer to the EThread structure

dbk_readMSR(msr): Reads the msr
dbk_writeMSR(msr, msrvalue): Writes the msr
dbk_executeKernelMemory(address, parameter) :
Executes a routine from kernelmode (e.g a routine written there with auto assembler)
parameter can be a value or an address. It's up to your code how it's handled

dbvm_initialize(offloados:Boolean OPTIONAL, reason:String OPTIONAL) : Initializes the dbvm functions (dbk_initialize also calls this) offloados is a boolean that when set will offload the system onto dbvm if it's not yet running (and only IF the dbk driver is loaded)

dbvm_initialized(): Returns true if dbvm is loaded and working. Does not load it
dbvm_setKeys(key1,key2,key3) - Sets the keys to operate DBVM. Key1 and Key3 are pointersize, key2 is 32-bit. Note that if key1 or key3 are 64-bit wide, 32-bit CE can not use DBVM. Returns true if DBVM is working, and automatically updates the current DBVM keys in CE and the driver if DBVM was already connected (e.g default keys)
dbvm_getMemory(): Returns the total memory free for DBVM, and the total number of full pages as secondary result
dbvm_addMemory(pagecount): Adds memory to DBVM (one page is 4096 bytes)
dbvm_readMSR(msr): See dbk_readMSR but then using dbvm
dbvm_writeMSR(msr, value): See dbk_writeMSR
dbvm_getCR4(): Returns the real Control Register 4 state

dbvm_readPhysicalMemory(address, size): bytetable
dbvm_writePhysicalMemory(address, bytetable)

dbvm_watch_writes(PhysicalAddress, bytesize OPTIONAL, OPTIONS OPTIONAL, internalentrycount OPTIONAL, Optional1, Optional2) :
  Starts watching writes to the given address range
  OPTIONS is a binary field.
    (1 << 0): Log the same RIP multiple times (if different registers)
    (1 << 1): Ignore the size field and log everything in the specified page
    (1 << 2): Logs record the floating point state
    (1 << 3): Logs contain a 4KB stack snapshot
    (1 << 4): does nothing
    (1 << 5): If the number of recorded entries gets bigger than internalentrycount, grow the list instead of discarding the entries
    (1 << 6): <reserved>
    (1 << 7): DBVMBP. Not a watch! When triggered changes RIP to Optional1 if in UserMode and Optional2 if in Kernelmode. These addresses need to contain an 0xcc (int3) . If 0 RIP will not be changed, and also not if the current state is not interuptable. Look at dbvm_bp_* functions for more information
  On success returns an ID to use with dbvm_watch_retrievelog and dbvm_watch_disable

dbvm_watch_reads(PhysicalAddress, bytesize OPTIONAL, OPTIONS OPTIONAL, internalentrycount OPTIONAL, Optional1, Optional2) : see dbvm_watch_writes but then for reads and writes
dbvm_watch_executes(PhysicalAddress, bytesize OPTIONAL, OPTIONS OPTIONAL, internalentrycount OPTIONAL, Optional1, Optional2) : see dbvm_watch_writes but then for executes
dbvm_watch_retrievelog(ID) : Returns an array of watch event data. (Context of the system at the time of the event, like registers)
dbvm_watch_disable(ID) : Disables the watch operation

dbvm_cloak_activate(physicalbase, virtualbase OPTIONAL):
  Hides an executable memory range (4096 bytes) from snooping eyes
  Note: It is recommended to cause a copy-on-write on the target first, else this will affect all processes that have this memory block loaded

dbvm_cloak_deactivate(physicalbase): Disables the cloak and restores the executable memory to what the system thinks it is
dbvm_cloak_readOriginal(physicalbase): Reads the memory that will get executed. On success returns a 4096 byte long bytetable starting from the base of the page (remember, lua indexes start at 1, so offset 0 is index 1)
dbvm_cloak_writeOriginal(physicalbase, bytetable[4096]): Writes the memory that will get executed.

dbvm_changeregonbp(physicaladdress, changereginfo, virtualaddress OPTIONAL): boolean
  sets a breakpoint at the given position. When a breakpoint hits the registers will be changed according to the changereginfo table
    changereginfo table: (set the field to nil, or don't define it, if you don't want to change it)
      newCF: integer/boolean (false=0, true=1)
      newPF: integer/boolean (false=0, true=1)
      newAF: integer/boolean (false=0, true=1)
      newZF: integer/boolean (false=0, true=1)
      newSF: integer/boolean (false=0, true=1)
      newOF: integer/boolean (false=0, true=1)
      newRAX: integer
      newRBX: integer
      newRCX: integer
      newRDX: integer
      newRSI: integer
      newRDI: integer
      newRBP: integer
      newRSP: integer
      newRIP: integer
      newR8: integer
      newR9: integer
      newR10: integer
      newR11: integer
      newR12: integer
      newR13: integer
      newR14: integer
      newR15: integer

dbvm_removechangeregonbp(physicaladdress) : Disables the changeregonbp breakpoint

dbvm_traceonbp(PhysicalAddress, stepcount, VirtualAddress, {secondaryoptions}) : Sets a int3 breakpoint at the given address after cloaking that page and when hit does a trace.
secondaryoptions is a table:
  logFPU: boolean
  logStack: boolean

dbvm_traceonbp_getstatus() : status, count, maxcount - Returns the status (0=no trace configured. 1=trace configured but not started yet, 2=trace configured and started, 3=trace done) and the number of steps the trace currently holds
dbvm_traceonbp_stoptrace() : requests the trace to stop

dbvm_traceonbp_remove(pa,force: boolean) - Disables the current trace and removes all results
dbvm_traceonbp_retrievelog() : Returns an array of traceentries. (Context of the system at the time of the event, like registers)

dbvm_bp_getBrokenThreadListSize() : Returns the number of breakpoint slots currently available
dbvm_bp_getBrokenThreadEventShort(id) : Returns a table with information about the specific breakpoint slow
dbvm_bp_getBrokenThreadEventFull(id) : Returns a bigger table (fpu and stack)
dbvm_bp_setBrokenThreadEventFull(id,state) : Sets the state of the frozen thread
dbvm_bp_resumeBrokenThread(id, continueMethod) : Resumes the specific thread. continueMethod can be 0=run, 1=step into
dbvm_bp_getProcessAndThreadIDFromEvent(ThreadEvent): processid,threadid - Returns the processid and threadid of the provided threadEvent. On failure processid will be nil, and threadid will contain text

dbvm_log_cr3_start() : Tells DBVM to record (up to 512) unique CR3 values it encounters
dbvm_log_cr3_stop() : Stops the logging and returns the results as a table

dbvm_speedhack_setSpeed(speed): Changes the speed the timestamp counter goes up (similar to speedhack in a process but affects the whole system including the clock)
dbvm_setTSCAdjust(enabled, timeout): If enabled (default true with timeout 2000) will return a small(20-30) timestamp between multiple rdtsc/rdtscp instructions. The timeout is the number of actual TSC to watch else the actual time is given. A high timeout can make your system unstable

dbvm_startcpuidlog() : starts logging of cpuid occasions (Watch dbvm's memory usage, it allocated and reallocates it's log while running making it possible to run out of memory)
dbvm_stopcpuidlog() : stops the logging
dbvm_getcpuidlog() returns an indexed table in the order of appearance. Each entry contains 2 fields: cr3 and rip

dbk_getCR0(): Returns Control Register 0
dbk_getCR3(): Returns Control Register 3 of the currently opened process. (Note: This will also work without dbk when only dbvm is loaded)
dbk_getCR4(): Returns Control Register 4
dbk_getPhysicalAddress(address): Returns the physical address of the given address
dbk_writesIgnoreWriteProtection(state): Set to true if you do not wish to initiate copy-on-write behaviour

getPhysicalAddressCR3(CR3, address): Looks up the physical address for the given virtual address in the given pagetable base. Returns nil if not paged
readProcessMemoryCR3(CR3, address, size): Reads the virtual memory of the given process's CR3 value. Returns a bytetable on success, nil if fail to read (paged out)
writeProcessMemoryCR3(CR3, address, bytetable): Reads the virtual memory of the

given process's CR3 value

allocateKernelMemory(size) : Allocates a block of nonpaged memory and returns the address
freeKernelMemory(address) : Frees the given memory region

mapMemory(address, size, frompid OPTIONAL, topid OPTIONAL): maps a specific address to the usermode context from the given PID to the given PID. If the PID is 0 or not specified, the cheat engine process is selected. This functions returns 2 results. Address and MDL. The MDL you will need for unmapMemory()
unmapMemory(address, mdl)

onAPIPointerChange(function): Registers a callback when an api pointer is changed (can happen when the user clicks ok in settings, or when dbk_use*** is used. Does NOT happen when setAPIPointer is called)

setAPIPointer(functionid, address): Sets the pointer of the given api to the given address. The address can be a predefined address set at initialization by Cheat Engine, or an address you got from an autoassembler script or injected dll (When Cheat Engine itself was targeted)

functionid:
  0: OpenProcess
    Known compatible address defines:
      windows_OpenProcess
      dbk_OpenProcess

  1: ReadProcessMemory
    Known compatible address defines:
      windows_ReadProcessMemory
      dbk_ReadProcessMemory
      dbk_ReadPhysicalMemory
      dbvm_ReadPhysicalMemory

  2: WriteProcessMemory
    Known compatible address defines:
      windows_WriteProcessMemory
      dbk_WriteProcessMemory
      dbk_WritePhysicalMemory
      dbvm_WritePhysicalMemory

  3: VirtualQueryEx

```
    Known compatible address defines:
      windows_VirtualQueryEx
      dbk_VirtualQueryEx
      VirtualQueryExPhysical

  4: CloseProcess
  5: IsWow64Process
  6: CreateToolhelp32Snapshot
  7: GetPhysicalAddress - stdcall BOOL (HANDLE hProcess; void* virtualaddress;
uint64_t *physicaladdress)

Extra variables defined:
dbk_NtOpenProcess : Address of the NtOpenProcess implementation in DBK32


The dbvm_ addresses should only be used with auto assembler scripts injected into
Cheat Engine
dbvm_block_interrupts  : Address of function dbvm_block_interrupts : DWORD; stdcall;
dbvm_raise_privilege   : Address of function dbvm_raise_privilege : DWORD; stdcall;
dbvm_restore_interrupts: Address of function dbvm_restore_interrupts : DWORD;
stdcall;
dbvm_changeselectors   : Address of function dbvm_changeselectors(cs,ss,ds,es,fs,gs:
dword): DWORD; stdcall;


D3DHOOK class:
The d3dhook functions provide a method to render graphics and text inside the game,
as long as it is running in directx9, 10 or 11

createD3DHook(textureandcommandlistsize OPTIONAL, hookmessages OPTIONAL)
  Hooks direct3d and allocates a buffer with given size for storage of for the
rendercommand list

  hookmessages defines if you want to hook the windows message handler for the
direct3d window. The d3dhook_onClick function makes use of that


  If no size is provided 16MB is used and hookmessages is true

  Note: You can call this only once for a process

  It returns a d3dhook object

properties
  Width: Integer : The width of the screen (readonly)
  Height: integer: The height of the screen (readonly)
  DisabledZBuffer: boolean : Set this to true if you don't want previously rendered
walls to overlap a newly rendered object (e.g map is rendered first, then the
players are rendered)
  WireframeMode: boolean : Set this to true if you don't want the faces of 3d
```

objects to be filled
  MouseClip: boolean : Set this if to true if you have one of those games where your mouse can go outside of the gamewindow and you don't want that.
  OnClick: function(d3dhook_sprite, x, y)
    A function to be called when clicked on an sprite (excluding the mouse)
    x and y are coordinates in the sprite object. If sprites overlap the highest zorder sprite will be given. It does NOT care if a transparent part is clicked or not

    Note: If you set this it can cause a slowdown in the game if there are a lot of sprites and you press the left button a lot

  OnKeyDown: function(virtualkey, char)
    function(vkey, char) : boolean
      A function to be called when a key is pressed in the game window (Not compatible with DirectInput8)
      Return false if you do not wish this key event to pass down to the game

methods
  beginUpdate() : Use this function when you intent to update multiple sprites,textcontainers or textures. Otherwise artifacts may occur (sprite 1 might be drawn at the new location while sprite 2 might still be at the old location when a frame is rendered)
  endUpdate() : When done updating, call this function to apply the changes
  enableConsole(virtualkey): Adds a (lua)console to the specific game. The given key will bring it up (0xc0=tilde(`~))
  createTexture(filename) : Returns a d3dhook_texture object
  createTexture(picture, transparentColor OPTIONAL): Returns a d3dhook_texture object
    if the picture is not a transparent image the transparentcolor parameter can be used to make one of it's colors transparent

  createFontmap(font) : Returns a d3dhook_fontmap object created from the given font
  createSprite(d3dhook_texture): returns a d3dhook_sprite object that uses the given texture for rendering
  createTextContainer(d3dhook_fontmap, x, y, text): Returns a d3dhook_textContainer object

D3DHook_Texture Class (Inheritance: Object)
This class controls the texture in memory. Without a sprite to use it, it won't show

properties
  Height: integer (ReadOnly)
  Width: integer (ReadOnly)
methods
  loadTextureByPicture(picture)

D3DHook_FontMap Class (Inheritance: D3DHook_Texture->Object)
A fontmap is a texture that contains extra data regarding the characters. This class is used by the textcontainer
Current implementation only supports 96 characters (character 32 to 127)

properties
  -
methods
  changeFont(font): Changes the fontmap to the selected font
  getTextWidth(string): Returns the width of the given string in pixels

D3DHook_RenderObject Class (Inheritance: Object)
The renderobject is the abstract class used to control in what manner objects are rendered.
The sprite and TextContainer classed inherit from this

properties
  X: Float - The x-coordinate of the object on the screen
  Y: Float - The y-coordinate of the object on the screen
  CenterX: Float - X coordinate inside the object. It defines the rotation spot and affects the X position
  CenterY: Float - Y " "
  Rotation: Float - Rotation value in degrees (0 and 360 are the same)
  Alphablend: Float - Alphablend value. 1.0 is fully visible, 0.0=invisible
  Visible: boolean - Set to false to hide the object
  ZOrder: integer - Determines if the object will be shown in front or behind another object
methods
  -

D3DHook_Sprite Class (Inheritance: D3DHook_RenderObject->Object)
A d3dhook_sprite class is a visible texture on the screen.

properties
  Width: Integer - The width of the sprite in pixels. Default is the initial texture width
  Height: Integer - The height of the sprite in pixels. Default is the initial texture height
  Texture: d3dhook_texture - The texture to show on the screen

methods
  -

D3Dhook_TextContainer Class (Inheritance: D3DHook_RenderObject->Object)

A d3dhook_sprite class draws a piece of text on the screen based on the used fontmap.
While you could use a texture with the text, updating a texture in memory is slow. So if you wish to do a lot of text updates, use a textcontainer

properties
  FontMap : The D3DHook_FontMap object to use for rendering text
  Text : The text to render
methods
  -

Disassembler Class (Inheritance: Object)

createDisassembler() - Creates a disassembler object that can be used to disassemble an instruction and at the same time get more data
getDefaultDisassembler() - Returns the default disassembler object used by a lot of ce's disassembler routines (Only use this from the main thread)
getVisibleDisassembler() - Deprectad. Returns a stub disassembler for backward compatability. It's function overrides are set the other visible disasemblers will use that if they themselves don't have an ovverride. Special codes are: {H}=Hex value {R}=Register {S}=Symbol {N}=Nothing special {C######}=RGB color , {B######}=Background RGB color were ###### is 0xBBGGRR

registerGlobalDisassembleOverride(function(sender: Disassembler, address: integer, LastDisassembleData: Table): opcode, description): Same as Disassembler.OnDisassembleOverride, but does it for all disassemblers, including newly created ones. Tip: Check the sender to see if you should use syntax highlighting codes or not
  This function returns an ID you can pass on to unregisterGlobalDisassembleOverride() 6.4+

unregisterGlobalDisassembleOverride(id)

properties
  LastDisassembleData : Table
  OnDisassembleOverride: function(sender: Disassembler, address: integer, LastDisassembleData: Table): opcode, description
  OnPostDisassemble: function(sender: Disassembler, address: integer, LastDisassembleData: Table, result: string, description: string): result, description
  syntaxhighlighting: boolean : This property is set if the syntax highlighting codes are accepted or not

Methods
  disassemble(address): Disassembles the given instruction and returns the opcode.

It also fills in a LastDisassembleData record
  decodeLastParametersToString() : Returns the unedited "Comments" information. Does not display userdefined comments
  getLastDisassembleData() : Returns the LastDisassembleData table.
    The table is build-up as follow:
      address: integer - The address that was disassembled
      opcode: string - The opcode without parameters
      parameters: string - The parameters
      description: string - The description of this opcode
      commentsoverride: string - If set, this will be the comments/LastParamatersToString result
      bytes: table - A table containing the bytes this instruction consists of (1..)

      modrmValueType: DisAssemblerValueType - Defines the type of the modrmValue field (dvtNone=0, dvtAddress=1, dvtValue=2)
      modrmValue: Integer - The value that the modrm specified. modrmValueType defines what kind of value

      parameterValueType: DisAssemblerValueType
      parameterValue: Integer - The value that the parameter part specified

      isJump: boolean - Set to true if the disassembled instruction can change the EIP/RIP (not ret)
      isCall: boolean - Set to true if it's a Call
      isRet: boolean - Set to true if it's a Ret
      isRep: boolean - Set to true if it's preceded by a Rep
      isConditionalJump: boolean - Set to true if it's a conditional jump

DissectCode class: (Inheritance: Object)
getDissectCode() : Creates or returns the current code DissectCode object

properties:
methods:
  clear() : Clears all data
  dissect(modulename) : Dissects the memory of a module
  dissect(base,size) : Dissect the specified memory region

  addReference(fromAddress, ToAddress, type, OPTIONAL isstring):
    Adds a reference. Type can be jtCall, jtUnconditional, jtConditional, jtMemory
    In case of jtMemory setting isstring to true will add it to the referenced strings list

  deleteReference(fromAddress, ToAddress)

  getReferences(address) : Returns a table containing the addresses that reference this address and the type

  getReferencedStrings(): Returns a table of addresses and their strings that have been referenced. Use getReferences to find out which addresses that are
  getReferencedFunctions(): Returns a table of functions that have been referenced. Use getReferences to find out which callers that are

  saveToFile(filename)
  loadFromFile(filename)

RIPRelativeScanner class: (Inheritance: Object)
createRipRelativeScanner(startaddress, stopaddress, includejumpsandcalls OPTIONAL):
createRipRelativeScanner(modulename, includejumpsandcalls OPTIONAL): Creates a RIP relative scanner. This will scan the provided module for RIP relative instructions which you can use for whatever you like
properties:
  Count: integer - The number of instructions found that have a RIP relative address
  Address[]: integer - An array to access the results. The address is the address of the RIP relative offset in the instruction

methods:
-

LuaPipe class: (Inheritance: Object)
  Abstract class that LuaPipeServer and LuaPipeclient inherit from. It implements the data transmission methods

properties
  Connected: boolean - True if the pipe is connected
  Timeout: integer - The number of seconds a read or write can take before the connection is closed and OnTimeout is called. Set to 0 to not time out. (Use this when you expect a function to take longer than normal)
  OnTimeout: function(sender) - Called when a read or write timeout has taken place and the connection has been terminated
  OnError: function(sender) - Called when a read or write encounters an error like disconnection
  OnAboutToTimeout: function(sender): boolean - Called before the connection is terminated and OnTimeout is called. Return false if you wish to wait longer (timer resets back to 0)

methods
  lock() : Acquire a lick on this pipe till unlock is called. If lock can not be acquired, wait. Recursive calls are allowed
  unlock()
  readIntoStream(stream,size)
  writeFromStream(stream,size OPTIONAL) : Writes the contents of the stream for size bytes into the pipe. If size is omitted, it sends all from the current pointer

  writeBytes(ByteTable, size OPTIONAL): Writes the provided byte table to the pipe.

if size is not provided, the whole table is sent. Returns the number of bytes sent, or nil on failure
  readBytes(size: integer): returns a byte table from the pipe, or nil on failure

  readDouble(): Read a double from the pipe, nil on failure
  readFloat(): Read a float from the pipe, nil on failure
  readQword(): Read an 8 byte value from the pipe, nil on failure
  readQwords(count): Reads 'count' times 8 byte values from the pipe and returns it as a table
  readDword(): Read a 4 byte value from the pipe, nil on failure
  readDwords(count): Reads 'count' times 4 byte values from the pipe and returns it as a table
  readWord(): Read a 2 byte value from the pipe, nil on failure
  readWords(count): Reads 'count' times 2 byte values from the pipe and returns it as a table
  readByte(): Read a byte from the pipe, nil on failure

  readString(size: integer): Reads a string from the pipe, nil on failure. (Can support 0-byte chars)
  readWideString(size: integer): Reads a widestring from the pipe, nil on failure

  writeDouble(v: double): Writes a double to the pipe. Returns the number of bytes sent, nil on failure
  writeFloat(v: single): writes a float to the pipe. Returns the number of bytes sent, nil on failure
  writeQword(v: qword): writes an 8 byte value to the pipe. Returns the number of bytes sent, nil on failure
  writeDword(v: dword): writes a 4 byte value to the pipe. Returns the number of bytes sent, nil on failure
  writeWord(v: word): writes a word to the pipe. Returns the number of bytes sent, nil on failure
  writeByte(v: byte): writes a byte to the pipe. Returns the number of bytes sent, nil on failure

  writeString(str: string; include0terminator: boolean OPTIONAL); Writes a string to the pipe. If include0terminator is false or not provided it will not write the 0 terminator byte. Returns the number of bytes written, or nil on failure
  writeWideString(str: widestring; include0terminator: boolean OPTIONAL); Writes a widestring to the pipe. If include0terminator is false or not provided it will not write the 0 terminator bytes. Returns the number of bytes written, or nil on failure

LuaPipeClient class: (Inheritance: LuaPipe>Object)
Class implementing a client that connects to a pipe

connectToPipe(pipename,timeout OPTIONAL): Returns a LuaPipeClient connected to the given pipename. Nil if the connection fails. Timeout is number of milliseconds before it disconnects on read/write operations. 0 or nil means never

properties:
methods:

-

LuaPipeServer Class: (Inheritance: LuaPipe>Object)
  Class launching the server side of a pipe

createPipe(pipename, inputsize OPTIONAL, outputsize OPTIONAL, maxInstanceCount OPTIONAL) : Creates a LuaPipeServer which can be connected to by a pipe client. InputSize and Outputsize define buffers how much data can be in the specific buffer before the writer halts. Default input and output size is 4096 for both. MaxInstanceCount determines the number of concurrent pipes of this names can be available. Default 1

properties
  valid: boolean - Returns true if the pipe has been created properly. False on failure (e.g wrong pipename)
  handle: integer - The handle of the pipe serverside (this can be used with duplicateHandle to get a handle into the target process)

methods
  acceptConnection() - Waits for a client to connect to this pipe (Warning: Freezes the thread this is executed in)

openLuaServer(Name):
  Opens a pipe with the given name. The LuaClient dll needs this name to connect to ce

  LuaClient.dll functions: (STDCALL calling machanic)
    BOOL CELUA_Initialize(char *name) : Initializes
    UINT_PTR CELUA_ExecuteFunction(char *luacode, UINT_PTR parameter)
      This function executes a lua function with parameters (parameter) and with the luacode as body Parameter will be treated as an integer
      In short:
        function(parameter)
          <luacode>
        end

    the return value of this function is the return value of the lua function (integer)

    UINT_PTR CELUA_ExecuteFunctionAsync(char *luacode, UINT_PTR parameter)
      See CELUA_ExecuteFunction but runs in the server thread instead of passing it to the main GUI and then wait for it's return

    integer CELUA_GetFunctionReferenceFromName(char *functionname): Returns a reference ID you can pass on to CELUA_ExecuteFunctionByReference

UINT_PTR CELUA_ExecuteFunctionByReference(int refid, int paramcount, PVOID *parameters, BOOL async):
This functions executes the function specified by reference id. If async is true, the code will run in a seperate thread instead of the main thread
paramcount is the number of parameters to pass on to the function
parameters is a pointer to a list of integers. 32-bit in 32-bit targets, 64-bit in 64-bit targets

Settings class
This class can be used to read out and set settings of cheat engine and of plugins, and store your own data

global functions
reloadSettingsFromRegistry(): This will cause cheat engine to reload the settings from the registry and apply them

getSettings(path Optional, nilResults OPTIONAL): Settings - Returns a settings object. If path is nil it will point to the Cheat Engine main settings (Registry) . If name is provided the settings currently accessed will be the one at the subkey provided. if nilResults is true values that don't exist return nil instead of an empty string
Note: Keep in mind that it returns a new object each call, even if he same name is used multiple times

properties
Path: string - Gets/Sets the current subkey (nil if main)
Value[]: A table access into the settings. e.g: Value["Count"]=12 .Setting vcalue to nil will delete it
[] : Same as Value[]

methods
getBinaryValue(name, stream) : Gets binary data from the registry value. Adds it after the current 'position'
setBinaryValue(name, stream, size OPTIONAL) : Sets binary data in the registry value. If size is given, write from the current position. If not, or size=0, writes from position 0 to the end
getBinaryValue(name): bytetable
setBinaryValue(name, bytetable)

SymbolList class
This class can be used to look up an address to a symbolname, and a symbolname to an address
It can also be registered with the internal symbol handler of cheat engine

  This class makes use of a special "Symbol" table construction that contains size and optionally other data
    Symbol Table:
      modulename: string
      searchkey: string
      address: integer
      symbolsize: integer

Global functions
  createSymbolList() : Creates an empty symbollist
  createSymbolList(initlist :{name,address}, name: string OPTIONAL) : creates a symbollist with addresses initialized based on a lua table that is build using name:address (like the result of compile) If name is provided automatically register the list under that name
  getMainSymbolList(): Returns the symhandler internal symbol list (Note: This does not contain .net, modulelist, or other info)
  enumRegisteredSymbolLists() : Returns a table containing all the registered symbollists

Properties
  PID: integer - The processid it refers to
  Name: string - A name that can be set to make it easier to identify

Methods
  clear()
  getSymbolFromAddress(address) : Searches the list for the given address. The address does not have to match the exact address. As long as it falls withing the range
  getSymbolFromString(searchkey)
  addModule(modulename, modulepath, address, size, is64bit)
  deleteModule(modulename)
  deleteModule(address)
  addSymbol(modulename, searchkey, address, symbolsize, skipAddressToSymbolLookup OPTIONAL, extradata OPTIONAL)
    Adds a symbol to the symbollist
    extradata is a table which can be used to fill in a return type and parameters for function calls. It has the following fields:
      returntype: string
      parameters: string

  deleteSymbol(searchkey)
  deleteSymbol(address)
  register() : Registers the current symbol list with the symbol handler
  unregister(): Unregisters the current symbol list from the symbol handler
  getModuleList(): Returns an indexed table with all the modules
  getSymbolList(): Returns an unindex table with each symbol being an element containing an address

```
Pagecontrol Class (WinControl->Control->Component->Object)
  This is an object that can hold multiple pages

global functions
  createPageControl(owner)
properties
  ShowTabs: boolean - Shows the tabs
  TabIndex: integer - Gets and sets the current tab
  ActivePage: TabSheet - Returns the current tabsheet.
  PageCount: integer - Gets the number of pages
  Page[]: TabSheet - Get a specific page (TabSheet)
methods
  addTab() : TabSheet - Creates a new TabSheet
  tabRect(index): Rect - returns a rect table describing the position of the
specific tab

TabSheet class (WinControl->Control->Component->Object)
  Part of a page control. This object can contain other objects
properties
  TabIndex: integer - the current index in the pagelist of the owning pagecontrol
methods

Internet class (Object)
global functions
  getInternet(string) - Returns an internet class object.  The string provided will
be the name of the client provided

properties
  Header : string - the additional header to be sent with the next getURL request
methods
  getURL(path) - returns a string containing the contents of the url. nil on failure
  postURL(path, urlencodeddata) - posts the given data to the path and returns the
results


CustomType class (Object)
The custom type is an convertor of raw data, to a human readable interpretation.

global functions
  registerCustomTypeLua(typename, bytecount, bytestovaluefunction,
valuetobytesfunction, isFloat, isString)
    Registers a Custom type based on lua functions
    The bytes to value function should be defined as "function bytestovalue
(b1,b2,b3,b4)" and return an integer as result
    The value to bytes function should be defined as "function valuetobytes
(integer)" and return the bytes it should write

    returns the Custom Type object
```

```
  registerCustomTypeAutoAssembler(script)
    Registers a custom type based on an auto assembler script. The script must
allocate an "ConvertRoutine" and "ConvertBackRoutine"

    returns the Custom Type object

  getCustomType(typename) : Returns the custom type object, or nil if not found

properties
  name: string
  functiontypename: string
  CustomTypeType: TCustomTypeType - The type of the script
  script: string - The custom type script
  scriptUsesFloat: boolean - True if this script interprets it's user side values as
float

methods
  byteTableToValue({bytetable},Address Optional)
  valueToByteTable(value, Address Optional)

TFrmTracer class (Inheritance:
Form->ScrollingWinControl->CustomControl->WinControl->Control->Component->Object)

properties
  Count: integer - number of entries in the list
  SelectionCount: integer - The number of selected entries

  Entry[index]: table - Information about each entry. Read only. (Index starts at 0)
    table is formatted as:
    {
      address: integer - address of the instruction
      instruction: string - disassembled instruction
      instructionSize: integer - bytesize of the instruction
      referencedAddress: integer - address the code references
      referencedData: bytearray - The bytes of the referenced data at the time of
tracing
      context: contexttable - the state of the cpu when this instruction got
executed (contains registers(EAX/RAX, ...), floating points(FP) and XMM values
      hasStackSnapshot: boolean - set to true if there is a stack entry
      selected: boolean - Set to true if the entry is selected

    }


  StackEntry[index]: bytearray - The stacksnapshot of that entry. Nil if not
available

methods
```

TfrmUltimap2 class(Inheritance:
Form->ScrollingWinControl->CustomControl->WinControl->Control->Component->Object)
getUltimap2(): TfrmUltimap2 - Returns the ultimap2 form, nil if not open

properties
  Count: integer - The number of entries in the list (READONLY)
methods
  isInList(address): boolean,count - returns true if the current address is in the list of addresses. In case of true, it also returns the count value (up to 255)

TfrmCodeFilter class(Inheritance:
Form->ScrollingWinControl->CustomControl->WinControl->Control->Component->Object)
getCodeFilter(): TfrmCodeFilter - Returns the codefilter form

properties
methods
  isInList(address): boolean - Returns true if this address is in the list

----SQL Classes----
CustomConnection class (Inheritance: Component->Object)
properties
  Connected: Boolean - Gets the current connection state, and lets you connect as well
  LoginPrompt: Boolean
  AfterConnect: function(sender)
  AfterDisconnect: function(sender)
  BeforeConnect: function(sender)
  BeforeDisconnect: function(sender)

methods
  close(forceClose:Boolean Optional)
  open()

Database Class (Inheritance: CustomConnection->Component->Object)
properties
  Connected: Boolean - Sewt this to true to activate the connection. (turns back to false on failure)
  DatabaseName: string
  KeepConnection: Boolean
  Params: Strings
  TransactionCount: integer readonly

methods

SQLConnection Class (Inheritance: Database->CustomConnection->Component->Object)
properties
  Password: String

```
  UserName: string
  Transaction: SQLTransaction - SQLTransaction object. Needs to be set
  CharSet: string
  HostName: string
  Options: string set - [scoExplicitConnect, scoApplyUpdatesChecksRowsAffected]

methods
  startTransaction()
  endTransaction()
  executeDirect(sql)
  getTableNames() : Returns a counted table with all tablenames


SQLite3Connection class(Inheritance:
SQLConnection->Database->CustomConnection->Component->Object)
createSQLite3Connection(owner) - creates an SQLite3Connection object
setSQLiteLibraryName(pathwithdllname)- Lets you set the path to the sqlite3.dll in
case it's not .\win*\sqlite3.dll

properties
methods
  createDB()
  dropDB()
  getInsertID(): integer

ODBCConnection class(Inheritance:
SQLConnection->Database->CustomConnection->Component->Object)
createODBCConnection(owner) - creates an ODBCConnection object
properties
  DatabaseName: string - Name of the odbc connection
  Driver: string
  FileDSN: string

methods



DBTransaction class (Inheritance: Component->Object)
properties
  Active: boolean
  DataBase: Database
methods
  closeDataSets()

SQLTransaction class (Inheritence: DBTransaction->Component->Object)
createSQLTransaction(owner): Creates an SQLTransaction object
properties
  SQLConnection: SQLConnection
  Params: StringList
  Options: string - set of [stoUseImplicit, stoExplicitStart]
```

```
  Action: string - options between caNone, caCommit, caCommitRetaining, caRollback,
    caRollbackRetaining

methods
  commit()
  commitRetaining()
  rollback()
  rollbackRetaining()
  startTransaction()
  endTransaction()


Param class (Inheritence: CollectionItem->Object))
properties
  Name: string
  Value: something
  DataType: string
  AsBoolean
  AsByteTable
  AsInteger
  AsNumber
  AsString
  Text
  Size
  Precision
  IsNull: boolean


methods



Params class (Inheritence: Collection->Object)
properties
  Items[index]: Param

methods
  AddParam(Param)


Field class (Inheritance: Component->Object)
properties
  FieldName: string
  Index: integer
  Value: something
  DataType: string
  AsBoolean
  AsByteTable
  AsInteger
  AsNumber
```

```
  AsString
  Text
  Size
  IsNull: boolean

methods


Fields class (Inheritence: Object)
properties
  Count: integer
  Fields[index]: Field
methods
  add(Field)
  clear()
  fieldByName(name): Field
  fieldByNumber(integer): Field
  indexOf(field): integer



Dataset class (Inheritance: Component->Object)
properties
  BlockReadSize: integer
  BOF: boolean; READONLY
  CanModify: boolean READONLY
  DefaultFields: boolean READONLY
  EOF: boolean; READONLY
  FieldCount: integer; READONLY
  Fields: Fields READONLY
  Found: boolean READONLY
  Modified: boolean READONLY
  IsUniDirectional: boolean READONLY
  RecordCount: integer READONLY
  RecNo: integer

  FieldValues[fieldname]: something
  Filter: string
  Filtered: boolean
  FilterOptions: set of [foCaseInsensitive, foNoPartialCompare]
  Active: boolean
  AutoCalcFields: boolean


methods
  append()
  appendRecord({values})
  cancel()
  checkBrowseMode()
  clearFields()
```

```
  close();
  controlsDisabled(): boolean
  cursorPosChanged;
  delete;
  disableControls;
  edit;
  enableControls;
  fieldByName(fieldname): Field
  findField(fieldname): Field
  findFirst() boolean
  findLast()
  findNext()
  findPrior()
  first()
  insert()
  isEmpty()
  last()
  locate(KeyFields, KeyValues, options:"[loCaseInsensitive, loPartialKey]"): boolean
  lookup(keyfields, KeyValues, ResultFields): something
  moveBy(distance)
  next()
  open()
  post()
  prior()
  refresh()
  updateCursorPos()
  updateRecord()



DBDataset class (Inheritance: Dataset->Component->Object)
properties
  DataBase: Database
  Transaction: DBTransaction

methods
-


CustomBufDataset class (Inheritance: DBDataset->Dataset->Component->Object)
properties
  FileName: string
  PacketRecords: integer
  UniDirectional: boolean
  IndexName: string
  MaxIndexesCount: integer
  ChangeCount: integer
  ReadOnly: boolean

methods
```

```
  applyUpdates(MaxErrors Optional)
  cancelUpdates()
  loadFromStream(stream)
  saveToStream(stream)
  loadFromFile(filename)
  saveToFile(filename)
  createDataset()


CustomSQLQuery class (Inheritance:
CustomBufDataset->DBDataset->Dataset->Component->Object)
properties
  prepared: boolean READONLY
  SQLConnection: SQLConnection
  SQLTransaction: SQLTransaction



methods
  prepare()
  unprepare()
  execSQL()
  rowsAffected()
  paramByName(paramname): Param




SQLQuery class (Inheritance:
CustomSQLQuery->CustomBufDataset->DBDataset->Dataset->Component->Object)
createSQLQuery(owner)
properties
  Database: Database

  SchemaType: string READFONLY - can be: stNoSchema, stTables, stSysTables,
stProcedures, stColumns, stProcedureParams, stIndexes, stPackages, stSchemata,
stSequences
  StatementType: string READONLY - can be :stUnknown, stSelect, stInsert, stUpdate,
stDelete,
      stDDL, stGetSegment, stPutSegment, stExecProcedure,
      stStartTrans, stCommit, stRollback, stSelectForUpd

  Params: Params object
  ParamCheck: Boolean
  ParseSQL: Boolean
  UpdateMode: string - can be :upWhereAll, upWhereChanged, upWhereKeyOnly
  UsePrimaryKeyAsKey: boolean
  ReadOnly: boolean
```

```
  SQL: string
  InsertSQL: stringlist
  UpdateSQL: stringlist
  DeleteSQL: stringlist
  RefreshSQL: stringlist
  Options: string - set of [sqoKeepOpenOnCommit, sqoAutoApplyUpdates, sqoAutoCommit,

      sqoCancelUpdatesOnRefresh, sqoRefreshUsingSelect]


methods
  -

------------------------------------------------------------

HotkeyHandlerThread(Inheritence: Thread)
getHotkeyHandlerThread() : Returns the hotkey handler thread used internally by CE

properties
  state: 0 ('htsActive')=Active , 1('htsMemrecOnly')=Memory records only,
2('htsNoMemrec')=Everything except memoryrecords, 3('htsDisabled')=disabled

methods
  -

RemoteThread class(Inheritance: -)
createRemoteThread(address, parameter)

properties
  Result : The 32-bit value returned by the thread

methods
  waitForThread(timeout OPTIONAL) : Waits for the thread to finish. Timeout is time
in milliseconds. If nil, the timeout it infinite. If 0, it returns without wait

ModuleLoader(Inheritance: -)
loadModule(pathtodll, executeEntryPoint OPTIONAL default=true, timeout OPTIONAL
default=nil=infinite)
loadModule(memorystream or tablefile, internalfilename, executeEntryPoint OPTIONAL
default=true, timeout OPTIONAL default=nil=infinite)
loadModuleLocal(^^^) : Same as loadModule but loads the module into CE itself

properties:
  loaded: boolean - true if successfuly mapped
  exports: Table containing all exports
  entryPoint: integer - address of the entrypoint


WriteLog class(Inheritence: -)
The writelog is the log that keeps all writes (when enabled)
```

getWriteLog() : Gets the current log

properties
  status: boolean
  logsize: integer

methods
  getLog(): table - Returns an indexed table with the write logs. each entry has a table with the fields: address, original and new

DotNetDataCollector class
getDotNetDataCollector() - Returns the current dotnetdatacollector object

properties
Attached: boolean - Returns true if attached to a process

methods
enumDomains(): table - Returns an index table containing all domains. Each entry is build up as {DomainHandle, Name}
enumModuleList(DomainHandle) : table - Returns an indexed table containing information about all modules. Each module entry is build up as {ModuleHandle, BaseAddress, Name}
enumTypeDefs(ModuleHandle) : table - Returns an indexed table containing information about all TypeDefs (classes) . Each entry is build up as {TypeDefToken, Name, Flags, Extends}
getTypeDefMethods(ModuleHandle, TypedefToken) : table - Returns a table containing all the methods in the specified typedef. Each entry is build up as {MethodToken, Name, Attributes, ImplementationFlags, ILCode, NativeCode, SecondaryNativeCode[]: Integer)
getTypeDefParent(ModuleHandle, TypedefToken): {ModuleHandle, TypedefToken}
getTypeDefData(ModuleHandle, TypedefToken) : table - Returns a table containing all the fields in the specified typedef. {ObjectType, ElementType, CountOffset, ElementSize, FirstElementOffset, ClassName, Fields[]{Offset, FieldType, Name} }
getMethodParameters(ModuleHandle, MethodDefToken): table - Returns a table containing all the parameters of the for method. {Name, CType}
getAddressData(address): table - Queries a specific address and returns information about it, assuming it is a valid object. It contains the following data: {StartAddress, ObjectType, ElementType, CountOffset, ElementSize, FirstElementOffset, ClassName, Fields[]{Offset, FieldType, Name} }

enumAllObjects(): table - Queries ALL defined objects. {StartAddress, Size, TypeID{token1,token2}, ClassName}   WARNING: This will take a LOOOOOONG time and if done from the main thread will make it look like CE is frozen
enumAllObjectsOfType(ModuleHandle, TypedefToken): {} - Returns a list of addresses that have this type

Diagram class(Inheritance:
CustomControl->WinControl->Control->LclComponent->Component->Object):
The diagram class lets you represent data using blocks and lines.

createDiagram(owner)

properties
  LineThickness: integer - Default thickness of lines in pixels
  LineColor: integer - Sets the default color of lines
  PlotPointColor: integer - Sets the default color for points
  BlockBackground: integer - Color of the diagramBlocks
  BackgroundColor: integer - Color of the diagram background
  ArrowStyles: string - can be one or more of the following between [and] :
                          asOrigin : There will be an arrow at the point of origin
                          asDestination : There will be an arrow at the destination
                          asPoints: There will be an arrow at plot point locations
                          asCenterBetweenPoints: There will be an arrow between two
points
                          asCenter: There will be an arrow in the center of the line

  BlockCount: integer - Returns the number of blocks in the diagram (readonly)
  Block[index]: DiagramBlock - Returns the block at the specific index
  LinkCount: integer - Returns the number of linkes in the diagram (readonly)
  Link[index]: DiagramLink - Returns the link at the specific index
  DrawPlotPoints: boolean - If set to true linkpoints will be shown
  AllowUserToCreatePlotPoints: boolean - If true(default) will allow the user to
create plotpoints by clicking on lines
  AllowUserToMovePlotPoints: boolean - If true(default) will allow the user to move
plotpoints by dragging them
  AllowUserToResizeBlocks: boolean - If true(default) will allow the user to resize
blocks
  AllowUserToMoveBlocks: boolean - If true(default) will allow the user to move
blocks around by draging the caption
  AllowUserToChangeAttachPoints: boolean - If true(default) will allow the user to
move move the point where a line connects to the block by dragging it
  ScrollX: integer - The horizontal scroll
  ScrollY: integer - The vertical scroll
  Zoom: float - The zoom level to use

methods
  createBlock(): DiagramBlock - Creates a new uninitialized block
  addConnection(Source: DiagramBlock, Destination: DiagramBlock): DiagramLink -
Creates a link between the two blocks
  addConnection(Source: BlockSideDescriptorTable, Destination:
BlockSideDescriptorTable) - Creates a link using the BlockSideDescritorTable table
definition

```
      BlockSideDescriptorTable=
      {
         Block: DiagramBlock - The block to attach to
         Side: integer - One of the blockside typedefs: dbsTop=0, dbsLeft=1,
dbsRight=2, dbsBottom=3, dbsTopLeft=4, dbsTopRight=5, dbsBottomLeft=6,
dbsBottomRight=7
         Position: integer - Position on the provided side based on the center. Only
for Sides 0 to 3
      }

  saveAsImage(filenamepng): saves the current display as an PNG image
  saveToFile(filename): Saves the state of the diagram to a file
  loadFromFile(filename): loads the state of the diagram from a file
  saveToStream(stream): Saves the state of the diagram to a stream
  loadFromStream(stream): loads the state of the diagram from a stream
  getObjectAt({x,y}): Returns the object at this position. nil if nothing



DiagramBlock class:
The diagramBlock is a block with a header and body which can contain text (ansi
escape codes supported)

properties
  Owner: Diagram - The diagram that this block is in
  X: integer - x-position of the block
  Y: integer - y-position of the block
  Width: integer - Width of the block
  Height: integer - Height of the block
  Caption: string - Header of the block (ansi escape codes supported)
  Strings: Strings - Strings object containing the lines of the block (ansi escape
codes supported)
  BackgroundColor: integer - When set overrides the default color for the block to
the given one
  TextColor: Integer - The starting/default textcolor
  Name: string - The name of the block
  AutoSize: boolean - If true the width and height of the box will adjust to the
given Caption and Strings
  AutoSide: boolean - If true the connection side of a link will be picked for you
instead of providing the side yourself
  AutoSideDistance: integer - If autoside is true and there are multiple lines on
the same side, this determines the distance between
  ShowHeader: boolean - If true show ther header. (default true)
  DragBody: boolean - If true allows dragging of the body. Useful when there is no
header (default false)
  Tag: integer - Use for whatever you like
  OnDoubleClickHeader: function(DiagramBlock) - Function to call when the block's
header is doubleclicked
  OnDoubleClickBody: function(DiagramBlock) - Function to call when the block's body
is doubleclick
```

  OnRenderHeader: function(sender,rect,beforeownerdraw): boolean - Function to call when the header is being rendered. This is called twice, before and after the normal painting code. In case of before and the function returns nil or false, the original text will not be drawn
  OnRenderBody: function(sender,rect,beforeownerdraw): boolean - ^ but then for body
  OnDragStart: funtion(DiagramBlock) - Called when a block starts getting dragged
  OnDrag: function(DiagramBlock) - Called when a block is dragged around
  OnDragEnd: function(DiagramBlock) - Called when a dragged block is released

methods
  getLinks(): table - Returns a table two elements: 'asSource' and 'asDestination'. each of those will have a table with DiagramLinks linking with the box
  overlapsWith(block): boolean - Returns true if the two blocks overlap
  intersectsWithLine({x,y},{x,y}): boolean,intersectpoint - Returns true and the point of intersection or false if no itersection

DiagramLink class:
Links between blocks

properties
  OriginBlock: DialogBlock - The source of the link
  DestinationBlock: DialogBlock - The destination of the link
  OriginDescriptor: BlockSideDescriptorTable
  DestinationDescriptor: BlockSideDescriptorTable
  PointCount: integer - The number of points in the table
  Points[index]: table{x,y} - Return a table with x,y coordinates for the point at the given index (index starts at 0)
  LineColor: integer - Color of the line , when set overrides the default
  Linethickness: integer - thickness of the line, when set overrides the default
  ArrowStyles: string - See diagram ArrowStyles
  Name: string - Name of the link
  Tag: integer - Use for whatever youy like
  OnDblClick: function(sender) - Function to call when the link is doubleclicked

methods
  hasLinkToBlock(DiagramBlock): boolean - Returns true/false depending on if this link has a connection to the provided block
  reset() - Sets all custom colors sizes back to default
  updateSide(BlockSideDescriptorTable): Updates the side of the block described by the BlockSideDescriptorTable
  getPointIndexAt(x,y): integer - Returns the pointindex at the given x, y coordinate
  addPoint(x,y,index OPTIONAL): creates a point. If no index is given if inside the line at that line index, else at the end of the list. If a index is given at that specific index.
  removeAllPoints(): Removes all points

RemoteExecutor class

The remoteExecute class creates an executor thread inside the target process waiting for commands to execute specific functions
An executor can only run one piece of code at a time, but you can have more than one executor

createExecuteMethodStub(callmethod, address, classregisternr, {type=typenr},typenr,{type=5, size=y, isOutputOnly=true/false, isInputOnly=true/false }, ...) - Creates an ExecuteCodeExStub object which the executor can execute
createExecuteCodeExStub(callmethod, address, {type=typenr},typenr,{type=5, size=y, isOutputOnly=true/false, isInputOnly=true/false }, ...) - Creates an ExecuteCodeExStub object which the executor can execute
createRemoteExecutor() - creates a new remote executor

properties

methods
  executeStub(ExecuteCodeExStub,{parameters},timeout,waittilldone)

CECustomButton class(Inheritance: CustomControl->WinControl->Control->Component->Object)
A more customizable button instead of the windows theme'd button, and lets you repaint it from scratch as well

createCECustomButton(owner)

properties
  ShowPrefix: boolean - Process first single '&' per line as an underscore and draw '&&' as '&'
  BorderColor: Color - The color of the button border
  BorderSize: integer - The thickness of the border
  ButtonColor: Color - The color of the button face
  ButtonHighlightedColor: Color - The color of the buttonface when highlighted(hovered over with the mouse)
  ButtonDownColor: Color - The color of the buttonface when the mouse is pressed down on it
  DrawFocusRect: boolean   - If true will draw a focus roundrect showing it has focus
  DrawBorder: boolean - default=true. Will draw a border around the button
  FocusedSize: integer - The with of the focus roundrect
  FocusElipseColor: Color - The color of the focus roundrect
  GrowFont: boolean read - When true the font will get resized till the caption fits the button
  RoundingX: integer
  RoundingY: integer
  CustomDrawn: boolean - Do your own drawing in the OnPaint property of the button
  FramesPerSecond: integer - If animation is used this will determine how often per second the OnPaint gets called
  ButtonAnimationSpeed: integer - If not customdrawn, this determnines how long the animations for enter/leave take

```
methods
  startAnimatorTimer() - Starts the animator timer which will trigger an OnPaint
with the speed of the current framesPerSecond property
  stopAnimatorTimer() - Stops the animator timer


------- XML -------
DOMNode class(Inheritance:)
properties
methods
  writeToFile(filename)
  writeToStream(stream)


DOMDocument class(Inheritance: DOMNode_TopLevel->DOMNode_WithChildren->DOMNode)
properties
methods


XMLDocument class(Inheritance:
DOMDocument->DOMNode_TopLevel->DOMNode_WithChildren->DOMNode)
createXMLDocument() - Creates an empty XML document
createXMLDocumentFromFile(filename) - reads the given filename and return an
XMLDocument with the parsed contents of the file
createXMLDocumentFromStream(stream) - reads the given stream and returns an
XMLDocument with the parsed contents of the stream

properties
methods

VirtualTreeColumn class(Inheritance: CollectionItem->Persistent->Object):
properties
  Index: integer - The position of this column in the header
  Text: string - the text the column shows
  Visible: shortcut for coVisible in Options
  Options: set(string):  a comma seperated list of the folowing options:
             coAllowClick           - Column can be clicked (must be enabled too).
             coDraggable            - Column can be dragged.
             coEnabled              - Column is enabled.
             coParentBidiMode       - Column uses the parent's bidi mode.
             coParentColor          - Column uses the parent's background color.
             coResizable            - Column can be resized.
             coShowDropMark         - Column shows the drop mark if it is currently
the drop target.
             coVisible              - Column is shown.
             coAutoSpring           - Column takes part in the auto spring feature
of the header (must be resizable too).
```

```
            coFixed               - Column is fixed and can not be selected or
scrolled etc.
            coSmartResize         - Column is resized to its largest entry which
is in view (instead of its largest visible entry).
            coAllowFocus          - Column can be focused.
            coDisableAnimatedResize- Column resizing is not animated.
            coWrapCaption         - Caption could be wrapped across several header
lines to fit columns width.
            coUseCaptionAlignment  - Column's caption has its own aligment.
            coEditable            - Column can be edited

methods

VirtualTreeColunmns class:
properties
  []: VirtualTreeColunmn

methods
  add(text OPTIONAL) : Created ands returns a new VirtualTreeColumn object

Header class:
properties
  AutoSizeIndex: integer - When Options contains hoAutoResize this determines which
column will be resized on resize of the control
  AutoResize: boolean - shortcut to access the hoAutoResize flag in Options
  MainColumn: integer - Column index to draw the treepart in.  (e.g: column0 is not
visible)
  Columns: VirtualTreeColunmns
  Options: set(string) - Options is a comma seperated string which can be one of the
following:
              hoAutoResize            - Adjust a column so that the header never
exceeds the client width of the owner control.
              hoColumnResize          - Resizing columns with the mouse is allowed.
              hoDblClickResize        - Allows a column to resize itself to its
largest entry.
              hoDrag                  - Dragging columns is allowed.
              hoHotTrack              - Header captions are highlighted when mouse
is over a particular column.
              hoOwnerDraw             - Header items with the owner draw style can
be drawn by the application via event.
              hoRestrictDrag          - Header can only be dragged horizontally.
              hoShowHint              - Show application defined header hint.
              hoShowImages            - Show header images.
              hoShowSortGlyphs        - Allow visible sort glyphs.
              hoVisible               - Header is visible.
              hoAutoSpring            - Distribute size changes of the header to all
columns, which are sizable and have the// coAutoSpring option enabled.
              hoFullRepaintOnResize   - Fully invalidate the header (instead of
subsequent columns only) when a column is resized.
              hoDisableAnimatedResize - Disable animated resize for all columns.
```

```
            hoHeightResize          - Allow resizing header height via mouse.
            hoHeightDblClickResize  - Allow the header to resize itself to its
default height.
            hoHeaderClickAutoSort   - Clicks on the header will make the clicked
column the SortColumn or toggle sort direction if it already was the sort column



methods


StringTreeOptions class:
properties
    AnimationOptions: comma seperated string containing one or more of:
          toAnimatedToggle          - Expanding and collapsing a node is animated
(quick window scroll).
          toAdvancedAnimatedToggle  - Do some advanced animation effects when
toggling a node.

    AutoOptions: comma seperated string containing one or more of:
          toAutoDropExpand          - Expand node if it is the drop target for
more than a certain time.
          toAutoExpand              - Nodes are expanded (collapsed) when getting
(losing) the focus.
          toAutoScroll              - Scroll if mouse is near the border while
dragging or selecting.
          toAutoScrollOnExpand      - Scroll as many child nodes in view as
possible after expanding a node.
          toAutoSort                - Sort tree when Header.SortColumn or
Header.SortDirection change or sort node if child nodes are added.
          toAutoSpanColumns         - Large entries continue into next column(s)
if there's no text in them (no clipping).
          toAutoTristateTracking    - Checkstates are automatically propagated for
tri state check boxes.
          toAutoHideButtons         - Node buttons are hidden when there are child
nodes, but all are invisible.
          toAutoDeleteMovedNodes    - Delete nodes which where moved in a drag
operation (if not directed otherwise).
          toDisableAutoscrollOnFocus - Disable scrolling a node or column into view
if it gets focused.
          toAutoChangeScale         - Change default node height automatically if
the system's font scale is set to big fonts.
          toAutoFreeOnCollapse      - Frees any child node after a node has been
collapsed (HasChildren flag stays there).
          toDisableAutoscrollOnEdit  - Do not center a node horizontally when it is
edited.
          toAutoBidiColumnOrdering  - When set then columns (if any exist) will be
reordered from lowest index to highest index and vice versa when the tree's bidi
mode is changed.
```

```
    MiscOptions: comma seperated string containing one or more of:
           toAcceptOLEDrop            - Register tree as OLE accepting drop target
           toCheckSupport             - Show checkboxes/radio buttons.
           toEditable                 - Node captions can be edited.
           toFullRepaintOnResize      - Fully invalidate the tree when its window is
resized (CS_HREDRAW/CS_VREDRAW).
           toGridExtensions           - Use some special enhancements to simulate
and support grid behavior.
           toInitOnSave               - Initialize nodes when saving a tree to a
stream.
           toReportMode               - Tree behaves like TListView in report mode.
           toToggleOnDblClick         - Toggle node expansion state when it is
double clicked.
           toWheelPanning             - Support for mouse panning (wheel mice only).
This option and toMiddleClickSelect are mutal exclusive, where panning has
precedence.
           toReadOnly                 - The tree does not allow to be modified in
any way. No action is executed and node editing is not possible.
           toVariableNodeHeight       - When set then GetNodeHeight will trigger
OnMeasureItem to allow variable node heights.
           toFullRowDrag              - Start node dragging by clicking anywhere in
it instead only on the caption or image. Must be used together with
toDisableDrawSelection.
           toNodeHeightResize         - Allows changing a node's height via mouse.
           toNodeHeightDblClickResize - Allows to reset a node's height to
FDefaultNodeHeight via a double click.
           toEditOnClick              - Editing mode can be entered with a single
click
           toEditOnDblClick           - Editing mode can be entered with a double
click
           toReverseFullExpandHotKey  - Used to define Ctrl+'+' instead of
Ctrl+Shift+'+' for full expand (and similar for collapsing)

    PaintOptions: comma seperated string containing one or more of:
           toHideFocusRect            - Avoid drawing the dotted rectangle around
the currently focused node.
           toHideSelection            - Selected nodes are drawn as unselected nodes
if the tree is unfocused.
           toHotTrack                 - Track which node is under the mouse cursor.
           toPopupMode                - Paint tree as would it always have the focus
(useful for tree combo boxes etc.)
           toShowBackground           - Use the background image if there's one.
           toShowButtons              - Display collapse/expand buttons left to a
node.
           toShowDropmark             - Show the dropmark during drag'n drop
operations.
           toShowHorzGridLines        - Display horizontal lines to simulate a grid.
           toShowRoot                 - Show lines also at top level (does not show
the hidden/internal root node).
           toShowTreeLines            - Display tree lines to show hierarchy of
```

nodes.
            toShowVertGridLines   - Display vertical lines (depending on columns) to simulate a grid.
            toThemeAware   - Draw UI elements (header, tree buttons etc.) according to the current theme if enabled (Windows XP+ only, application must be themed).
            toUseBlendedImages   - Enable alpha blending for ghosted nodes or those which are being cut/copied.
            toGhostedIfUnfocused   - Ghosted images are still shown as ghosted if unfocused (otherwise the become non-ghosted images).
            toFullVertGridLines   - Display vertical lines over the full client area, not only the space occupied by nodes. This option only has an effect if toShowVertGridLines is enabled too.
            toAlwaysHideSelection   - Do not draw node selection, regardless of focused state.
            toUseBlendedSelection   - Enable alpha blending for node selections.
            toStaticBackground   - Show simple static background instead of a tiled one.
            toChildrenAbove   - Display child nodes above their parent.
            toFixedIndent   - Draw the tree with a fixed indent.
            toUseExplorerTheme   - Use the explorer theme if run under Windows Vista (or above).
            toHideTreeLinesIfThemed   - Do not show tree lines if theming is used.
            toShowFilteredNodes   - Draw nodes even if they are filtered out.

    SelectionOptions: comma seperated string containing one or more of:
            toDisableDrawSelection   - Prevent user from selecting with the selection rectangle in multiselect mode.
            toExtendedFocus   - Entries other than in the main column can be selected, edited etc.
            toFullRowSelect   - Hit test as well as selection highlight are not constrained to the text of a node.
            toLevelSelectConstraint   - Constrain selection to the same level as the selection anchor.
            toMiddleClickSelect   - Allow selection, dragging etc. with the middle mouse button. This and toWheelPanning are mutual exclusive.
            toMultiSelect   - Allow more than one node to be selected.
            toRightClickSelect   - Allow selection, dragging etc. with the right mouse button.
            toSiblingSelectConstraint - Constrain selection to nodes with same parent.
            toCenterScrollIntoView   - Center nodes vertically in the client area when scrolling into view.
            toSimpleDrawSelection   - Simplifies draw selection, so a node's caption does not need to intersect with the selection rectangle.
            toAlwaysSelectNode   - If this flag is set to true, the tree view tries to always have a node selected. This behavior is closer to the Windows TreeView and useful in Windows Explorer style applications.
            toRestoreSelection   - Set to true if upon refill the previously

selected nodes should be selected again. The nodes will be identified by its caption only.

    StringOptions: comma seperated string containing one or more of:
           toSaveCaptions   - If set then the caption is automatically saved with the tree node, regardless of what is saved in the user data.
           toShowStaticText   - Show static text in a caption which can be differently formatted than the caption but cannot be edited.
           toAutoAcceptEditChange   - Automatically accept changes during edit if the user finishes editing other then VK_RETURN or ESC. If not set then changes are cancelled.

methods

VirtualStringTree class:
createVirtualStringTree(owner)
properties
  NodeDataSize: integer - The number of bytes to assign for data storage in a node (default is set to hold enough for a pointer)
  OnExpanding: function(sender, node): boolean - called when a node gets expended. Return true to allow
  OnGetText: function(sender, nodeindex, columnindex, node, texttype) : string - called when the text to draw is requested. return the string you wish the field to have
  OnPaintText: function(sender, canvas, node, column, texttype) - called when the text is about to be painted. Use this to change the canvas font colors or do some background painting
  OnDrawText: function(sender, canvas, node, column, celltext, cellrect): defaultdraw - called when text is being painted. return true if you wish the normal painting to happen besides your own, false if you wish to do it all yourself
  OnFreeNode: function(sender, node) - Called when a node gets deleted
  OnInitNode: function(sender, parentnode, node, initialStates) : initialStates - Called when a node gets created. Return the initialStates set (string) to set it's state. initialStates can be a comma seperated string containing one or more of: ivsDisabled, ivsExpanded, ivsHasChildren, ivsMultiline, ivsSelected,ivsFiltered, ivsReInit
  TreeOptions: StringTreeOptions
  FullRowSelect: boolean - Shortcut to TreeOptions->SelectionOptions->toFullRowSelect
  FocusedNode: node - gets/sets the focused node
  FocusedColumn: integer - gets/sets the focused column index

  NodeParent[node]: node - gets/sets the node parent
  NodeHeight[node]: height of the node - gets/sets the height of the node
  HasChildren[node]: boolean - gets/sets the state that the node has children
  Selected[node]: boolean - gets/sets the selected state of the node
  Expanded[node]: boolean - gets/sets if the node is expanded (calls onExpanding when set to true)

```
  NodeChildCount[node]: integer - Gets the number of childnodes
  NodeIndex[node]: integer - Gets the index of the node in relation to it's parent

methods
  saveToFile(filepath)
  loadFromFile(filepath)


  clear() - deletes all nodes
  beginUpdate()
  endUpdate()
  addChild(parent): node - Adds a child to the tree. Node is an ambiguous object
that can only be accessed by the VirtualTreeString object
  addToSelection(node)
  removeFromSelection(node)

  getRootNode(): node - returns the rootnode

  nodeSelected(node): boolean - returns true if the node is selected
  nodeChecked(node): boolean - returns true if the node is checked
  enumSelectedNodes(): table - returns an indexed table of selected nodes
  enumCheckedNodes() : table - returns an indexed table of checked nodes
  enumChildren(node) : table - returns an indexed table of childnodes

  deleteNode(node)  - Deletes the given node
  deleteSelectedNodes() - Deletes all selected nodes

  getNodeData(node) : bytetable- returns the data of the node as a bytetable
  setNodeData(node, bytetable) - sets the data of the node using a bytetable
  getNodeDataAsInteger(node):integer - returns the node data interpreted as a single
integer
  setNodeDataAsInteger(node, integer) - sets the node data as an integer
  getNodeDataPointer(node): integer - Returns the pointer to the node data. You can
use the Read/Write*Local functions to access it instead of a bytetable. (Handy when
nodedatasize is very big)

  getNodeParent(node): node - returns a parent node, or nil
  getFirstChild(node): node - returns the first child of a node
  getNextSibling(node): node - gets the next sibling
  getNodeLevel(node): integer - gets the level at which this node resided (it does
this by calling getNodeParent until it hits nil)

GDBServerDebuggerIntaace functions: (only functional when using the gdb server
debugger interface)
  gdb_connected(): returns true when connected
  gdb_stopped(): returns true if the target is suspended by gdb
  gdb_break(): sends a ctrl+c to the gdb server which tells it to stop the target
  gdb_command(string, timeout OPTIONAL): Sends a command to the gdb server and wait
for timeout in milliseconds for a reply (default is 2000) if the string is empty
only receive data
```

  gdb_getcurrentstopreason(): returns the current stopreason. Nil if not stopped
  gdb_setCurrentInstructionpointer(address)
  gdb_getCurrentInstructionpointer(address)

----CEServer----
connectToCEServer(hostname,port) - Connects to the given host and port. On success, most commands subsequent will be handled by the server. Like processlist, memory reading, etc...
isConnectedToCEServer(): boolean - returns true if currently connected to a ceserver
getCEServerPath(): string - returns the path where ceserver is located on the target
getCEServerInterface(): returns a ceserverInterface object

CEServerInterface class:
properties
  connected: boolean
  path: string
  Option[optionname]: string
methods
  getOptionList(): table - returns an indexed table with all available options. Each table entry contains: optname, parentoptname, optdescription, acceptablevalues, currentvalue, optiontype(0=not an option, just a label, 1=bool, 2=int, 3=float, 4=double, 5=text)
  getCurrentPath(): Returns the current working path
  setCurrentPath(): Changes the current working path
  enumFiles(path OPTIONAL): Returns an indexed list describing the contents of the given path. It constists of tables containing 'name' and 'type'. Type can be : DT_UNKNOWN, DT_FIFO, DT_CHR, DT_DIR, DT_BLK, DT_REG, DT_LNK, DT_SOCK, DT_WHT

  createDir(path): Create the provided path
  getFilePermission(path) : returns the filemode bitmask of the given file (parse the bitmask yourself)
  setFilePermission(path, permissions) : sets the filemode bitmask of the given file
  getFile(path, stream) : reads a file from the given path and stores it in the provided stream object
  getFilePart(path, stream, startoffset, length) : reads a file from the given path and stores it in the provided stream object. lets you specify the start offset and number of bytes to read
  putFile(path, stream) : creates a file at the given path with the contents of the stream object
  getLoadModuleError(): Returns a string that explains why the last module injection failed

ce lua extensions:
string table:
split(character) -> table of strings separated by character
endsWith('string') : boolean - Returns true if the string ends with the given string
startsWith('string') : boolean - Returns true if the string starts with the given string