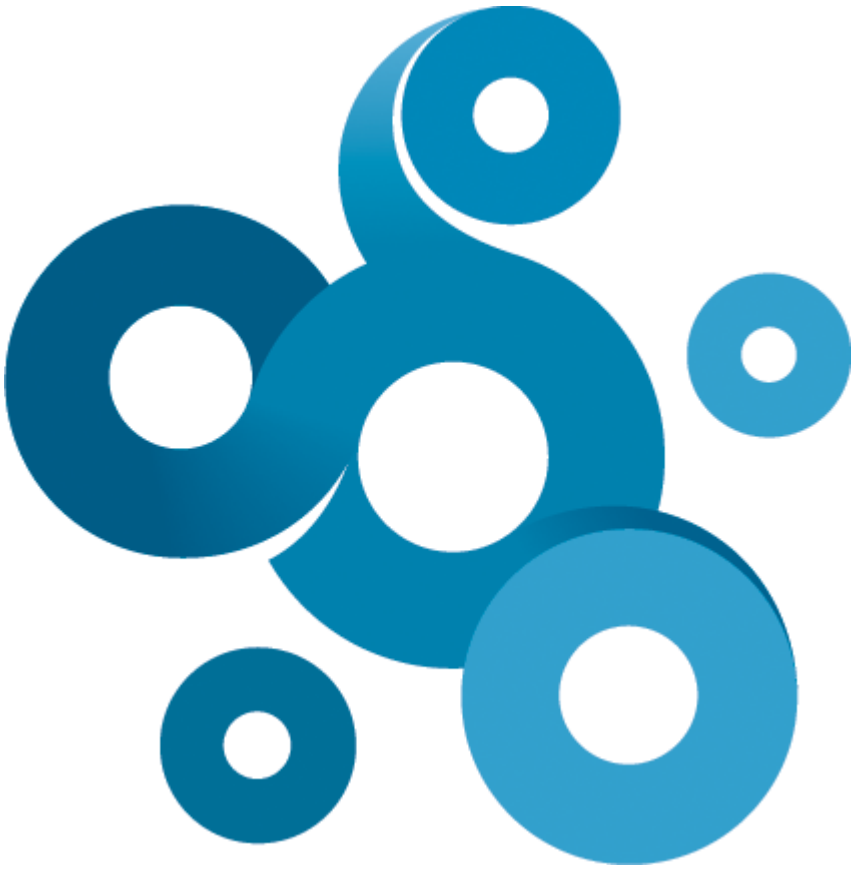
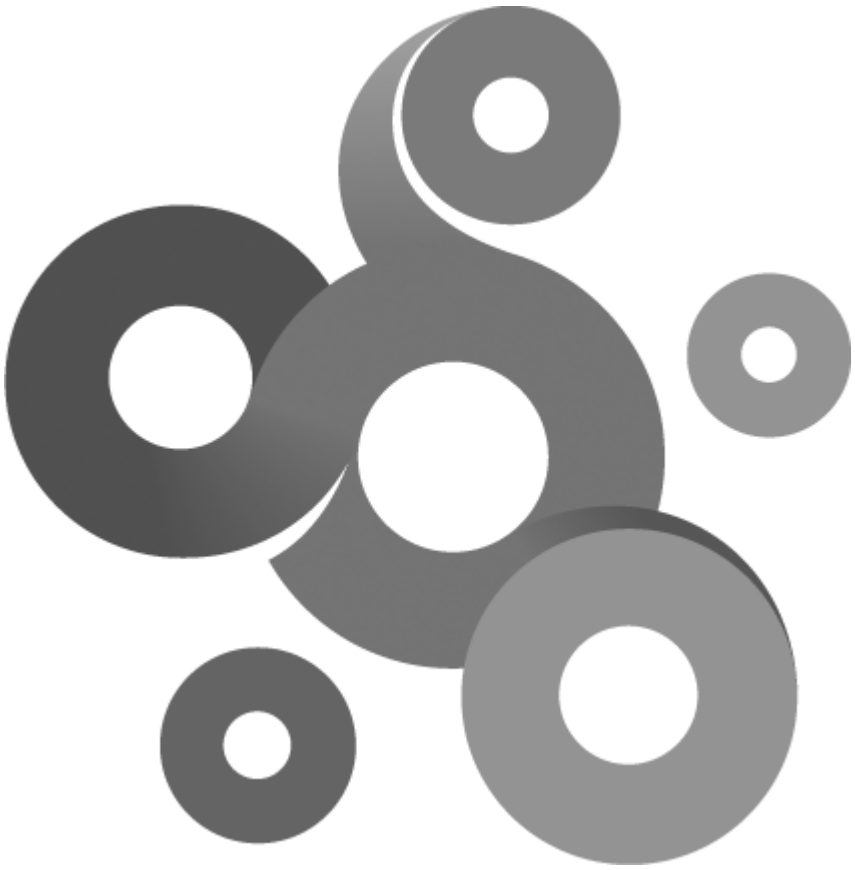


Lua API 4.0

Lua API

Printed: maj 2011 in Härnösand, Sweden

Table of Contents



Section I

1 About Lua

About Lua

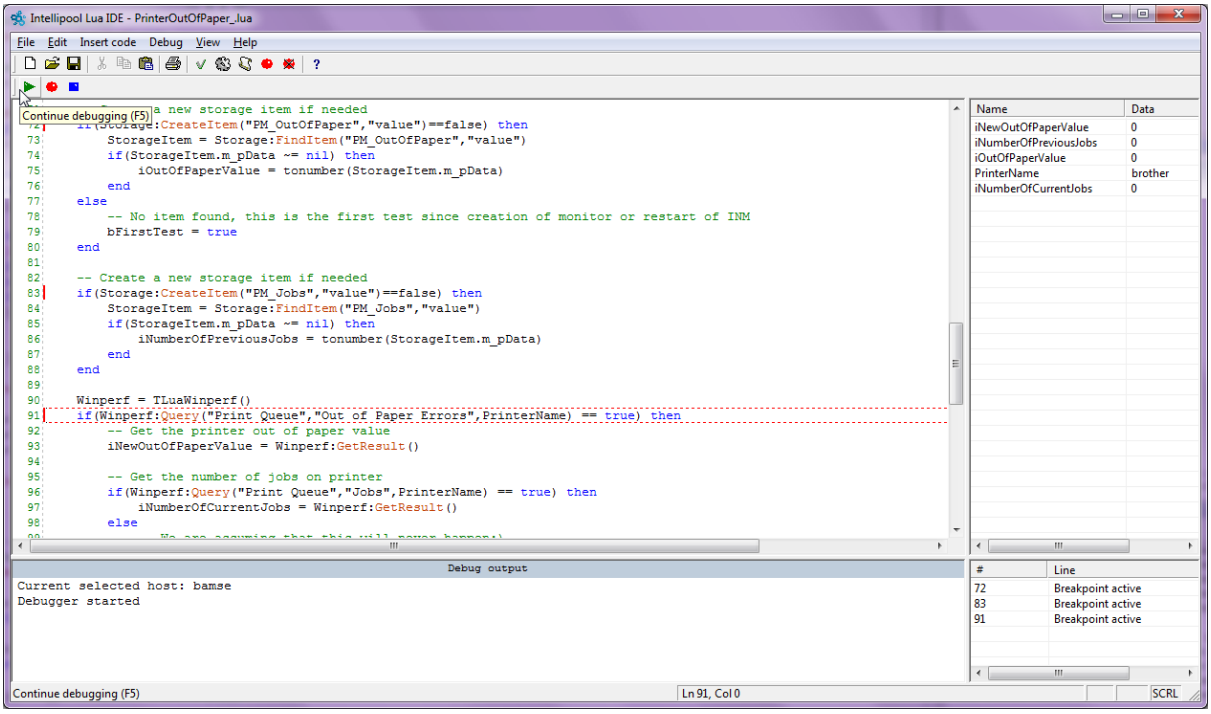
Lua is a powerful light-weight programming language designed for extending applications. Lua is also frequently used as a general-purpose, stand-alone language. Lua is free software. Lua combines simple procedural syntax with powerful data description constructs based on associative arrays and extensible semantics. Lua is dynamically typed, interpreted from byte codes, and has automatic memory management with garbage collection, making it ideal for configuration, scripting, and rapid prototyping.

KNM and Lua

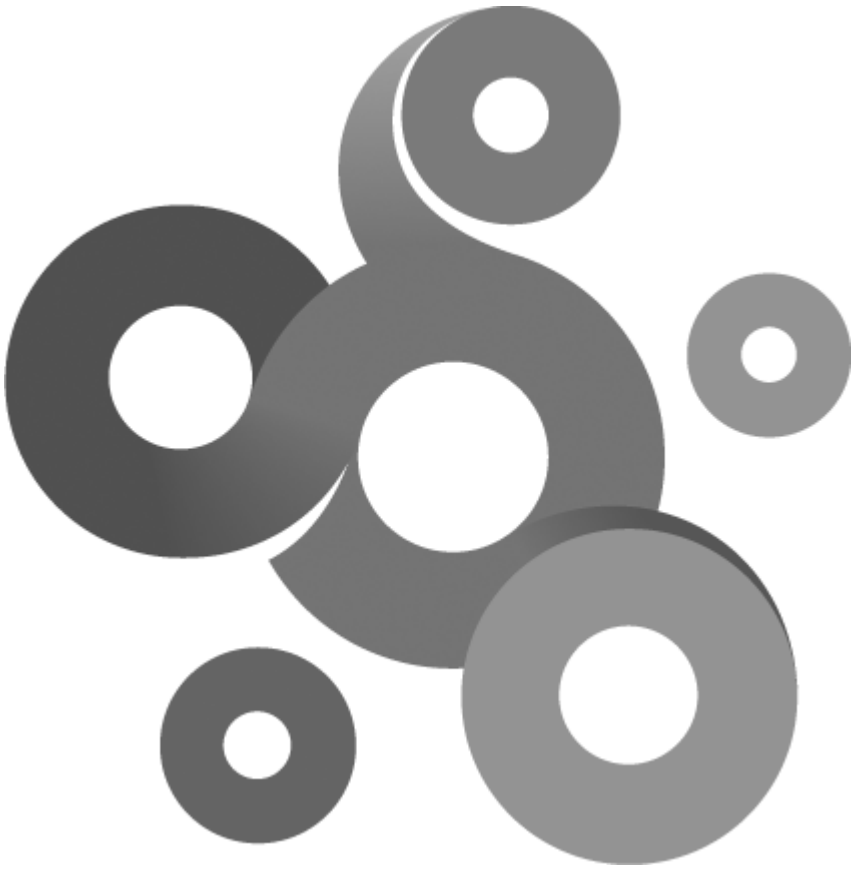
Kaseya Network Monitor includes support for the Lua scripting language (www.lua.org).

- Customers can create custom made monitors to test systems and equipment not supported by any current monitoring solution.
- New monitors, actions and events can be created and tested in the development environment provided by Kaseya, before they are exported and used in Kaseya Network Monitor.
- A comprehensive library of pre-made classes, such as SFTP client, HTTP client and file management, are available to developers.
- The develop environment includes debugger, keyword highlighting, integrated help and other features available in state-of-the-art development tools.

The development environment can be downloaded from our homepage at http://www.kaseya.com/



Lua IDE v3

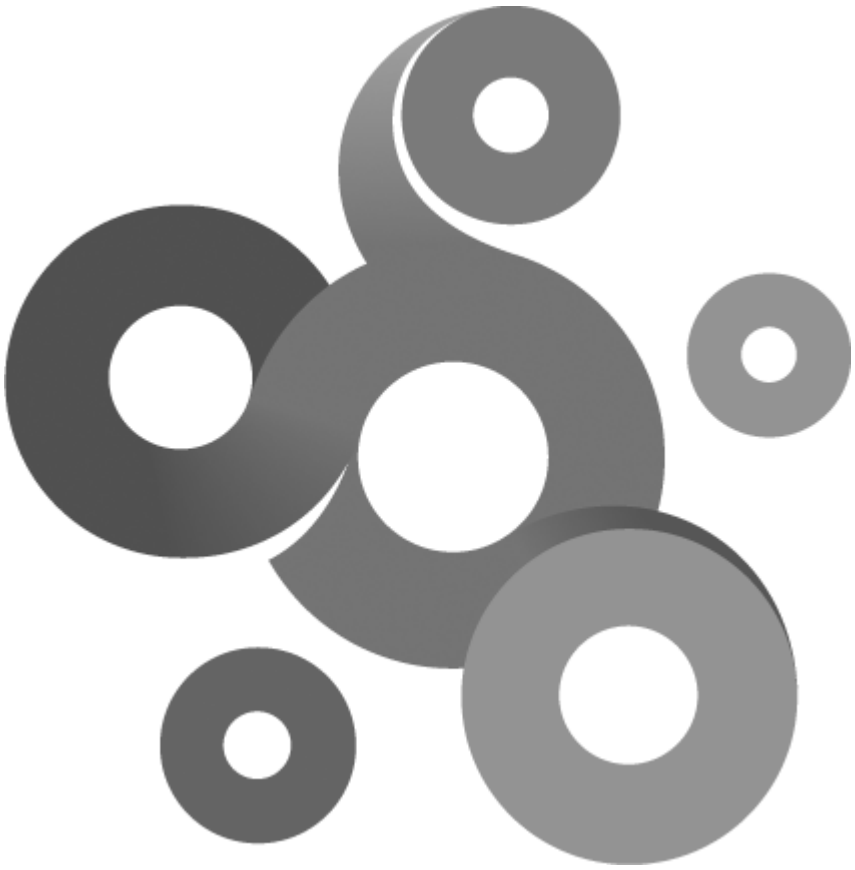


Section II

2 Introduction

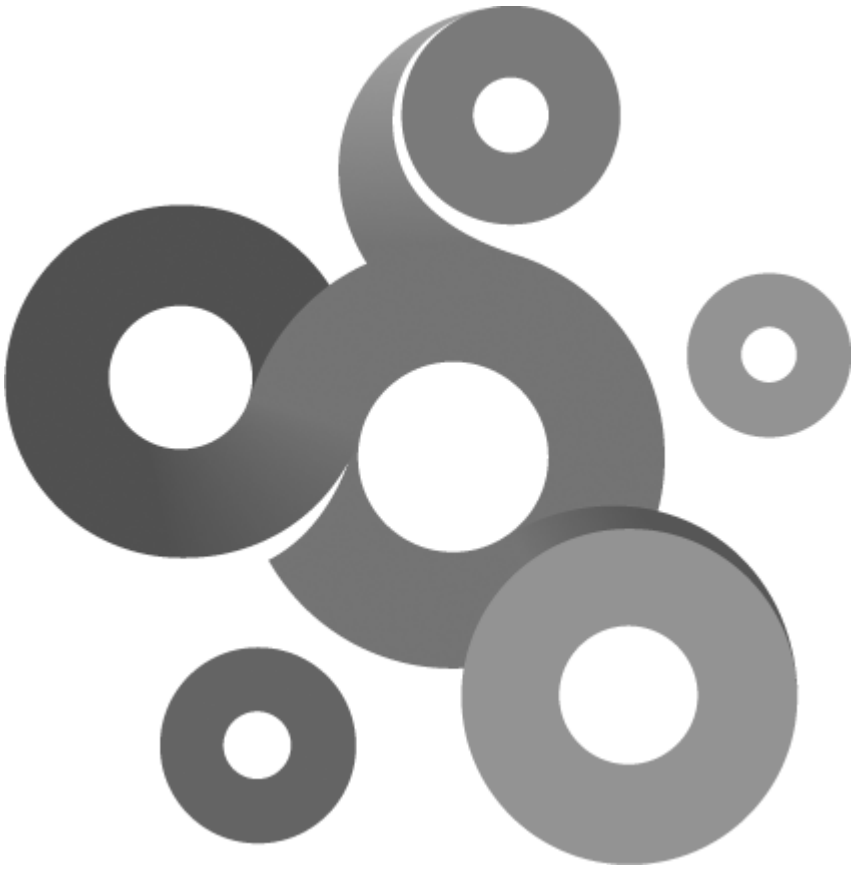
This documentation covers the Kaseya Network Monitor Lua API. This documentation do not cover the Lua language, for more information about the Lua language visit http://www.lua.org.

KNM uses Lua 5.0.



Section III

3 Copyright information



Section IV

4 Programing model

When creating a custom Lua script for use in Kaseya Network Monitor there is a number of requirements that the script must fulfill in order to be successfully executed by Kaseya Network Monitor.

4.1 Advanced script

The advanced script model gives the script author new powerful tools to control parameters giving as arguments to the script. This makes it possible to make Lua scripts that have the same look and feel like native monitor types.

Reserved function names

There are two reserved function names, used by KNM to query information. These function names can not be used for any other purpose.

OnConfigure

This function is called by KNM to have the script populate a LuaScriptConfigurator class instance. The information is then used to create a user interface for the script. The name of the instance must be "Config" (note that casing) so KNM may find it in the Lua stack when the function returns.

OnEnumerate

Each field in the user interface can be enumerated, KNM calls the OnEnumerate function to have the script populate a data strcture, LuaScriptEnumResult, with values the user can select. The OnEnumerate have one parameter, sFieldToEnum, that is used by the script to determin which field/argument to provide enumeration results for. The returned instance must be named "Enum" (note the casing).

The entry point

The advanced script model requires the OnConfigure function to set the name of the entry point function. This function is called by KNM to start the execution of the script. The name of the entry point is by default "main" but can be set by the programmer to any name except the reserved function names.

Example

```
-- This function is called by KNM when enumerating a field

function OnEnumerate(sFieldToEnum)

      -- The variable returned must be called "Config" so KNM can find it.
      Enum = LuaScriptEnumResult()

      -- Second argument
      if sFieldToEnum == "Argument 2" then
            Enum:Add("First value")
            Enum:Add("Second value")
            Enum:Add("Third value")
      end

      return Enum
end

-- This function is called by KNM to retrieve a script configuration

function OnConfigure()

      -- The variable returned must be called "Config" so KNM can find it.
      Config = LuaScriptConfigurator()
```

```
      -- Author.
      Config:SetAuthor("My name")

      -- Description.
      Config:SetDescription("Description of the script, including usage,
parameters etc")

      -- Minimum build version of KNM, set to zero for if no specific
build version is required.
      Config:SetMinBuildVersion(0)

      -- Script version (major/minor)
      Config:SetScriptVersion(1,0)

      -- A parameter configuration, add them in the order the script is
extracting them.
      Config:AddArgument("Argument 1","This is the description of the
first argument",LuaScriptConfigurator.CHECK_NOT_EMPTY)

      -- Add another parameter, a select box with 3 values.
      Config:AddArgument("Argument 2","This is the description of the
second argument",LuaScriptConfigurator.
CHECK_NOT_EMPTY+LuaScriptConfigurator.ENUM_AVAIL)


      -- Set the entry point, this is the function called by KNM
      Config:SetEntryPoint("main")

      -- Done with configuration, return the object
      return Config
end

-- This is the entry point

function main()

      sFirstArgument = GetArgument(0)
      sSecondArgument = GetArgument(1)

      SetExitStatus("OK",true)

end
```

4.2 Simple script

The simple script model has been used in KNM since the first release and should now be seen as deprecated. Its maintained for compatibility with older scripts.

4.3 Object context

Functions are relative to the object context.

All calls that are accessing resources is relative to the parent object. For example, if the scripts opens a file the path supplied to the open function must be relative to the object.

Example

Set the host to be the address of the windows computer "domainserver".

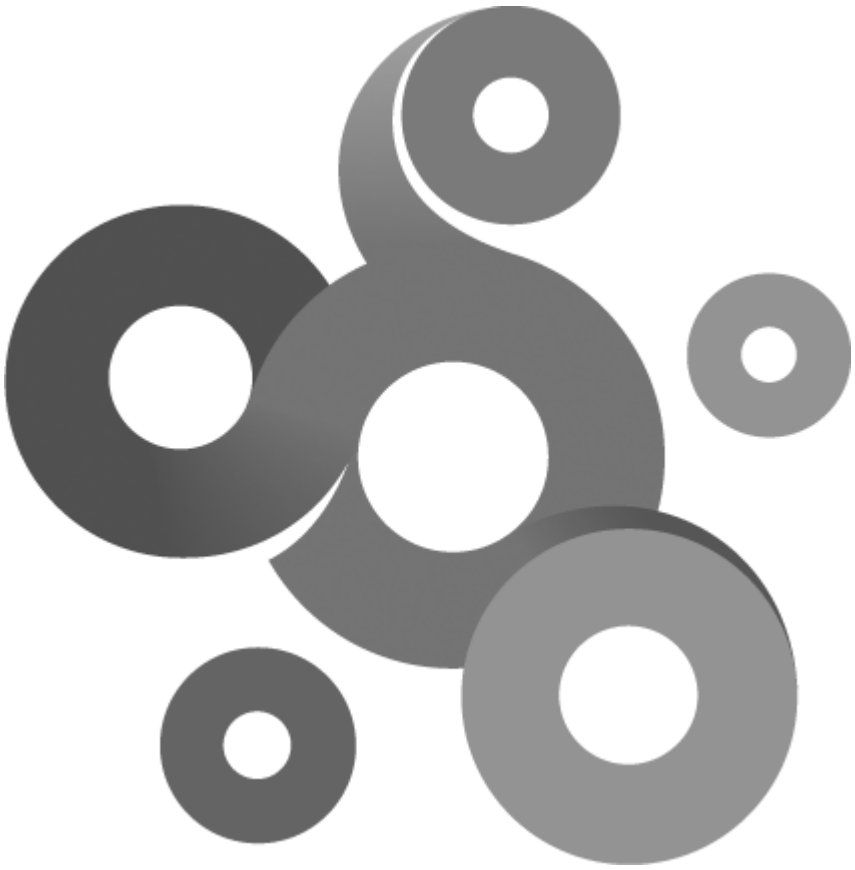
```
TLuaFile:Open("C:\\test.txt");
```

Calling the function would make the script open the file test.txt located on the C: harddisk of the computer "domainserver".

This is also why all communication related classes such as TLuaFTPClient, TLuaHTTPClient and TLUASocket only takes an port number argument, the IP address is hard coded to the current object by the framework.

4.4 Result

When an script exit it needs to tell KNM if the test was successful or not. A global function are provided for this purpose, SetExitStatus 14. SetExitStatus is mandatory and must be called before the script terminates.



Section V

5 Global functions

Global functions is functions not associated with an object. There is a number of global functions in the KNM Lua API, some is required to call when a scripts exit.

5.1 ConvertFromUTF16

string ConvertFromUTF16(local UTF16data,int iSize)

Return values

A 8 bit string converted from the UTF16 string.

Parameters

UTF16data

UTF16 (double byte) string read by TLuaFile::ReadData.

iSize

Size of string.

Remarks

The function only accepts data created by the TLuaFile::ReadData function.

5.2 FormatErrorString

string FormatErrorString(int iError)

Return values

A string describing the error code iError.

Parameters

iError

An Windows error code obtained previously by calling the GetLastError function.

Remarks

This function can be useful supplying meaningful text to the user instead of an error code.

5.3 GetAccountPassword

string GetAccountPassword()

Return values

The password of the logon account selected in the monitor.

Remarks

A calling application can pass an account to a Lua script for customize logon behaviour. Note that this function always returns an empty string in the IDE and is only meaningful in a script executed by KNM.

5.4 GetAccountUser

string GetAccountUser()

Return values

The username of the logon account selected in the monitor.

Remarks

A calling application can pass an account to a Lua script for customize logon behaviour. This function

provide a way to get the username part of the account, use GetAccountPassword to extract the password part. Note that this function always returns an empty string in the IDE and is only meaningful in a script executed by KNM.

5.5 GetArgument

string GetArgument(int iNumber)

Return values

A argument passed by calling application.

Parameters

iNumber

An zero based index of the argument to retrieve. The max number of arguments can be determined by calling GetArgumentCount.

Remarks

A calling application can pass a number of arguments to a Lua script to customize its behaviour. With this function and the related GetArgumentCount the programmer can extract the arguments.

5.6 GetArgumentCount

int GetArgument()

Return values

The number of arguments passed to the program by a calling application.

Remarks

A calling application can pass a number of arguments to a Lua script to customize its behaviour. With this function the programmer can determine how many arguments there is to extract.

5.7 GetLastError

int GetLastError()

Return values

The last error code generated by a call to a library function. The error code is a standard windows error code.

Remarks

The SetLastError can be used to clear the current Windows error code before calling a function.sds

5.8 GetObjectAddress

string GetObjectAddress()

Return values

The address entered into the object address field.

Remarks

The string can be used as a unique identifier when saving data to TLuaStorage.

5.9 IsIDE

bool IsIDE()

Return values

Boolean true if the script is executed by the IDE, false if the script is executed by KNM.

Remarks

This functions can be used if the script is executed by KNM or the IDE.

5.10 MessageBox

MessageBox(string sText)

Parameters

sText

Text to display in message box.

Remarks

This functions invokes a standard OS message box to display a string. This function is only available in the IDE. Note that the message box when displayed, halts the execution of the script until its closed.

5.11 print

print(string sText)

Parameters

sText

Text to be printed to the output window

Remarks

This function can be used to print text to the output window for debug purpose. When the script is executed by KNM the text printed with this function serve no purpose.

5.12 SetExitStatus

SetExitStatus(string sString,bool bSuccess)

Parameters

sString

An string describing the result for the script.

bSuccess

If non-zero (boolean true) the script is considered to been executed successfully by the framework. If this value is set to zero (boolean false) the function SetErrorString should be called as well, with a string describing the error status.

Remarks

This function must be called when a script is exiting, the function tells KNM if the script was successfully or not, the text supplied with the function will be used by KNM to set last status text in the interface if the script is executed in the context of an agent.

5.13 SetLastError

SetLastError(int iErrorCode)

Parameters

iErrorCode

An integer corespondent to a Windows specific error code.

Remarks

The function sets the last error code that later can be retrieved by GetLastError 13.

5.14 StoreStatisticalData

bool StoreStatisticalData(int iRecordSet,float fData,float fThreshold,string Unit)

Return values

True if data was successfully stored to statistical database, false if there was an parameter error.

Parameters

iRecordSetIndex

A zero based index of the statistical channel to store data into. See remarks for valid constants.

fData

Floating point data sampled by the script.

fThreshold

Optional threshold value for the sample data, this value should be constant in all calls.

Unit

Optional string describing the unit of the data, this value should be constant in all calls. The string can be max 16 chars in length or the call will fail.

Remarks

This function gives the script the ability to store statistical data. Currently there is 8 channels that can be used for the purpose.

The record set index parameter can be one of the following constants.

- LUA_RECORDSET_1
- LUA_RECORDSET_2
- LUA_RECORDSET_3
- LUA_RECORDSET_4
- LUA_RECORDSET_5
- LUA_RECORDSET_6
- LUA_RECORDSET_7
- LUA_RECORDSET_8

5.15 StoreStatisticalData

bool StoreStatisticalData(int iRecordSet,float fData,float fThreshold,int iVirtualType,int iVirtualUnit,string Unit)

Return values

True if data was successfully stored to statistical database, false if there was an parameter error.

Parameters

iRecordSetIndex

A zero based index of the statistical channel to store data into. See remarks for valid constants.

fData

Floating point data sampled by the script.

fThreshold

Optional threshold value for the sample data, this value should be constant in all calls.

iVirtualType

Type of data stored.

iVirtualUnit

Selected unit of stored type. See remarks for valid combinations of types and units.

Unit

Optional string describing the unit of the data, this value should be constant in all calls. The string can be max 16 chars in length or the call will fail.

Remarks

This function is only available for advanced scripts. The difference between this function and the old function with the same name is the ability to store type information with the data.

iVirtualType and iVirtualUnit can be used in the following combinations:

VT_SWAP_UTILIZATION

VT_MEMORY_UTILIZATION

VT_DISK_UTILIZATION

VT_CPU_UTILIZATION

UNIT_TYPE_PERCENT

VT_FREE_DISKSPACE

UNIT_TYPE_MEGABYTE

UNIT_TYPE_GIGABYTE

UNIT_TYPE_TERABYTE

VT_SQL_QUERY

UNIT_TYPE_NONE

VT_TEMPERATURE:

UNIT_TYPE_FAHRENHEIT

UNIT_TYPE_CELSIUS

UNIT_TYPE_KELVIN

VT_HUMIDITY

UNIT_TYPE_PERCENT

VT_WETNESS

UNIT_TYPE_NONE

VT_VOLTAGE

UNIT_TYPE_VOLT

VT_BANDWIDTH_UTILIZATION

UNIT_TYPE_PERCENT

VT_BANDWIDTH_USAGE

UNIT_TYPE_KBPS

UNIT_TYPE_MBPS

UNIT_TYPE_GBPS

VT_DIRECTORY_SIZE:

UNIT_TYPE_MEGABYTE

UNIT_TYPE_GIGABYTE

UNIT_TYPE_TERABYTE

VT_DIRECTORY_COUNT

UNIT_TYPE_NONE

VT_PING_ROUNDTRIP

UNIT_TYPE_MILLISECONDS

UNIT_TYPE_SECONDS

VT_PING_PACKETLOSS

UNIT_TYPE_PERCENT

VT_MAIL_ROUNDTRIP:

UNIT_TYPE_MILLISECONDS

UNIT_TYPE_SECONDS

VT_MEMORY_USAGE

UNIT_TYPE_MEGABYTE

UNIT_TYPE_GIGABYTE

VT_TRANSFER_SPEED

UNIT_TYPE_NONE

VT_HTTP_FETCHTIME

UNIT_TYPE_MILLISECONDS

UNIT_TYPE_SECONDS

VT_WMI_GENERIC_VALUE

VT_LUA_GENERIC_VALUE

VT_WINPERF_GENERIC_VALUE

VT_SSH2SCRIPT_GENERIC_VALUE

VT_SNMP_GENERIC_VALUE

UNIT_TYPE_NONE

VT_CURRENT

UNIT_TYPE_AMPERE

VT_FANSPEED

UNIT_TYPE_RPM

VT_LUMINOSITY

UNIT_TYPE_LUX

The record set index parameter can be one of the following constants.

- LUA_RECORDSET_1
- LUA_RECORDSET_2
- LUA_RECORDSET_3
- LUA_RECORDSET_4
- LUA_RECORDSET_5
- LUA_RECORDSET_6
- LUA_RECORDSET_7
- LUA_RECORDSET_8

5.16 Wait

Wait(int iMs)

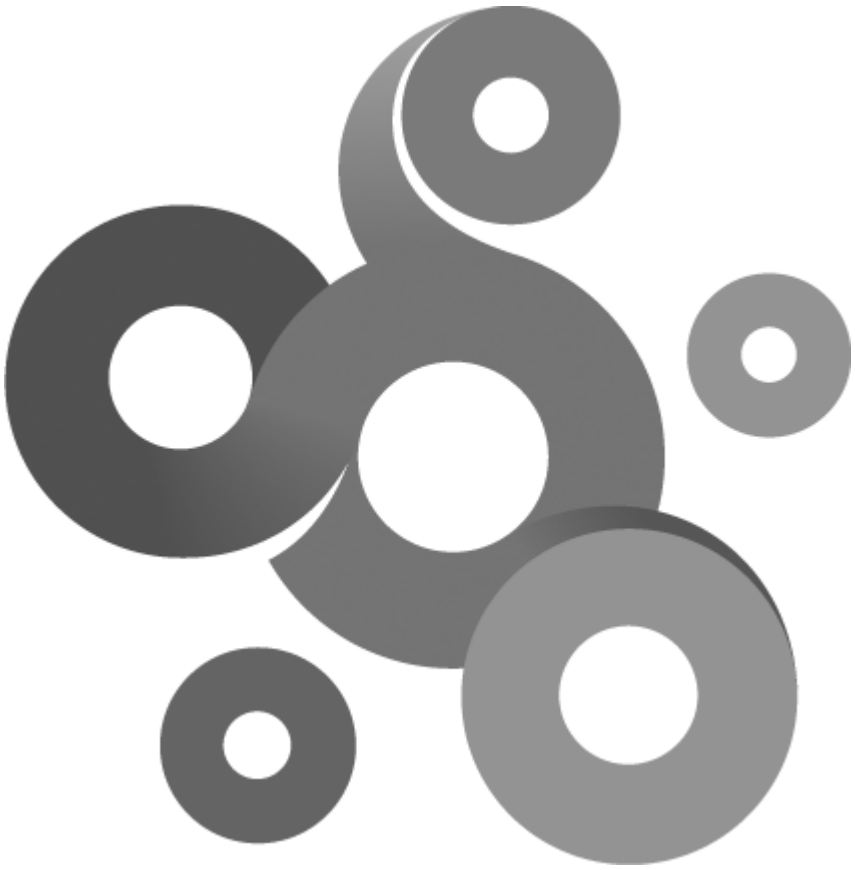
Parameters

iMs

The number of milliseconds the script execution should wait.

Remarks

This functions invokes the OS function "Sleep" to suspend execution of the thread the script is executed by.



Section VI

6 LuaScriptEnumResult

This class provides an interface to enter enumeration results in the OnEnumeration callback function.

Example

```
function OnEnumerate(sFieldToEnum)

      -- The variable returned must be called "Config" so KNM can find it.
      Enum = LuaScriptEnumResult()

      -- Second argument
      if sFieldToEnum == "Argument 2" then
            Enum:Add("First value")
            Enum:Add("Second value")
            Enum:Add("Third value")
      end

      return Enum
end
```

6.1 Add

Add(const string &sDisplayValue,const string &sUsageValue="")

Parameters

sDisplayValue

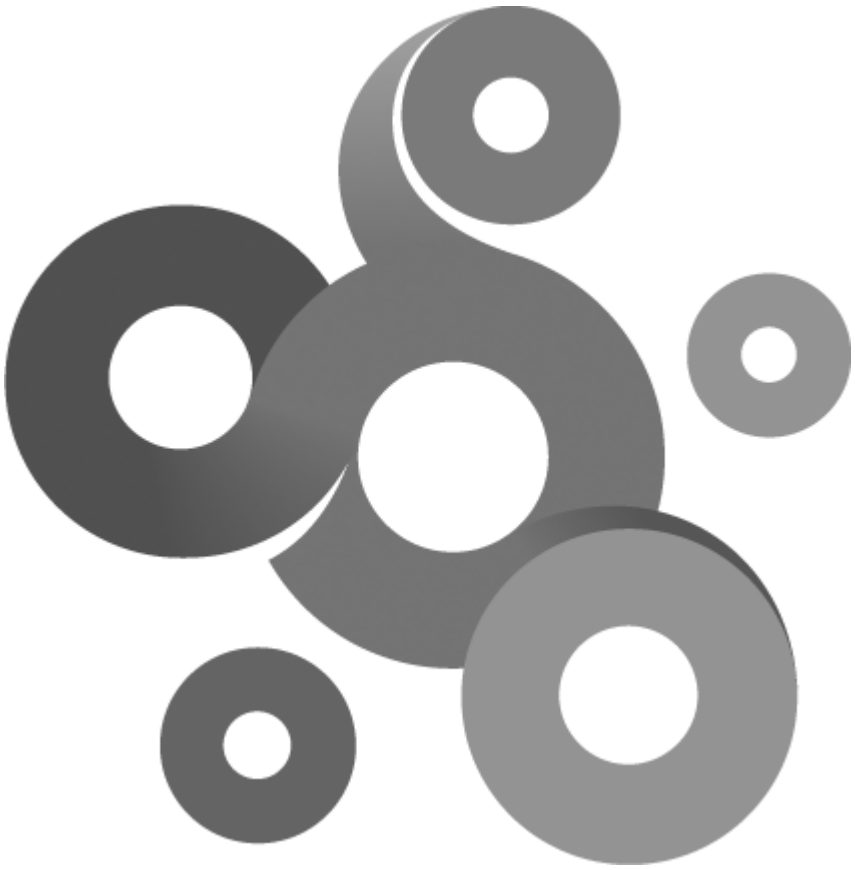
Value to display in as a option to select.

sUsageValue

(Optional) A value that will be used instead of the display value.

Remarks

The optional sUsageValue can be used when you have a very complex and long values and need a simpler way to display the options. When used the sDisplayValue will be the value presented to the user, but the sUsageValue will be the value used by KNM.



Section VII

7 LuaScriptConfigurator

This class provides an interface to create configuration information that KNM uses to present an user interface for the script.

Example

```
function OnConfigure()

      -- The variable returned must be called "Config" so KNM can find it.
      Config = LuaScriptConfigurator()

      -- Author.
      Config:SetAuthor("My name")

      -- Description.
      Config:SetDescription("Description of the script, including usage,
parameters etc")

      -- Minimum build version of KNM, set to zero for if no specific
build version is required.
      Config:SetMinBuildVersion(0)

      -- Script version (major/minor)
      Config:SetScriptVersion(1,0)

      -- A parameter configuration, add them in the order the script is
extracting them.
      Config:AddArgument("Argument 1","This is the description of the
first argument",LuaScriptConfigurator.CHECK_NOT_EMPTY)

      -- Add another parameter, a select box with 3 values.
      Config:AddArgument("Argument 2","This is the description of the
second argument",LuaScriptConfigurator.
CHECK_NOT_EMPTY+LuaScriptConfigurator.ENUM_AVIL)


      -- Set the entry point, this is the function called by KNM
      Config:SetEntryPoint("main")

      -- Done with configuration, return the object
      return Config
end
```

7.1 AddArgument

int AddArgument(string sName,string sDescription,int iFlags);

Return values

A handle that can be used referring to this argument in subsequent calls.

Parameters

sName

Name of the argument field

sDesc

Description of the field

iFlags

Flags controlling validation. See remarks for flags.

Remarks

These are the valid flags. Some of them can be combined.

CHECK_NOTHING	Default value, any type, including no text, is accepted.
CHECK_NOT_EMPTY	Check if argument is empty. Can not be combined with CHECK_NOTHING.
CHECK_RANGE_LOW	Must be used with CHECK_NUMERIC. Validates numeric value is within range (low range).
CHECK_RANGE_HIGH	Must be used with CHECK_NUMERIC. Validates numeric value is within range (high range).
CHECK_NUMERIC	Validates that value is numeric (real or integer)
ENUM_AVAIL	Indicates that there is a enumeration callback with pre-defined values available for this field.

7.2 SetCharacterLimits

SetCharacterLimits(int iArgIndex,int iMaxCharacters,int iMaxVisibleCharacters)

Parameters

iArgIndex

Handle returned by AddArgument 22

iMaxCharacters

Max input characters for argument.

iMaxVisibleCharacters

Max visible characters, must be equall to or less than iMaxCharacters.

Remarks

The function sets the maximum length of an argument and how many of those characters that are visible in the interface (length of input field).

7.3 SetNumericLimits

SetNumericLimits(int iArgIndex,float fLow,float fHigh)

Parameters

iArgIndex

Handle returned by AddArgument 22

Low

Low range

High

High range

Remarks

This function sets the acceptable range of real and integer values entered into the field. The argument must have CHECK_RANGE_LOW and CHECK_RANGE_HIGH flags set.

7.4 SetEntryPoint

SetEntryPoint(string sName)

Parameters

sName

Name of the entry point function.

Remarks

The function register the name of the entry point function. This is the function that KNM will call as the starting point of execution. The default value is "main".

7.5 SetAuthor

SetAuthor(string sName)

Parameters

sName

Name of the author of the script.

Remarks

This function sets the author of the script. Its used for descriptive purpose when a user loads a third party script, to inform him/her who has written the script.

7.6 SetDescription

SetDescription(string sDescription)

Parameters

sDescription

A description of the function of the script.

Remarks

The description of a script should in a few lines tell the user what the script do and if there is any known limitations to the script. There is no upper limit of the text, but it should be kept brief.

7.7 SetMinBuildVersion

SetMinBuildVersion(int iMinBuildNumber)

Parameters

iMinBuildNumber

The minimum build number of KNM that the script requires.

Remarks

The minimum build number is a very important field to set. It tells KNM if the script can be used with the current version of KNM. By default, this number should be set to the build number the author have used to test the script with.

7.8 SetScriptVersion

SetScriptVersion(int iMajor,int iMinor)

Parameters

iMajor

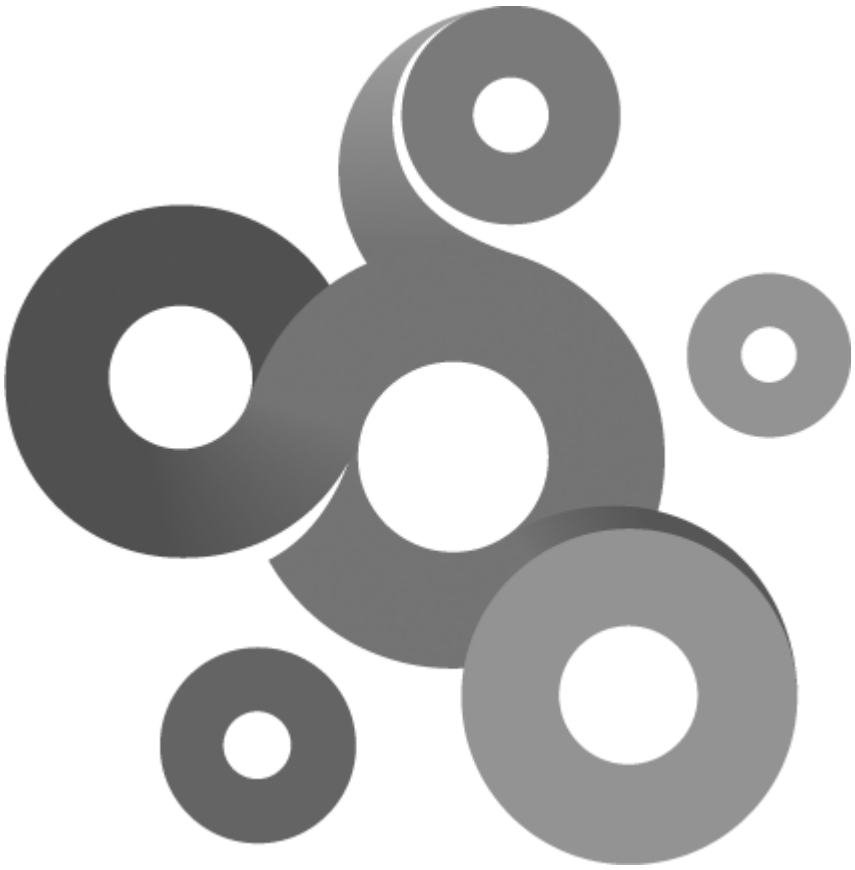
The major version number of the script.

iMinor

The minor version number of the script.

Remarks

The author of the script should set a version number of the script. A major version of 0 indicates that the script is in a "beta" stage and should only be used for further development by other users. Each time the script is modified, version number should be increased. A change in the major version number should reflect a large re-write or improvement, the minor version number indicates a smaller improvement.



Section VIII

8 TLuaDateTime

The TLuaDateTime provides with date and time functions. Time is local time represented as seconds from 1st of January 1970.

8.1 Add

Add(TLuaDateTime DateTime)

Parameters

DateTime

TLuaDateTime instance obtained from other class function or constructed.

Remarks

The function will add the time contained in the DateTime parameter to the object.

8.2 Create

Create(int iYear,int iMonth,int iDay,int iHour,int iMinute,int iSecond)

Parameters

iYear

Year, eg. 1972

iMonth

Number of month, eg. 10

iDay

Number of day in month, eg. 2

iHour

Hour to use, can be zero

iMinute

Minute to use, can be zero

iSecond

Second to use, can be zero

Remarks

The function creates a TLuaDateTime containing an absolute time.

8.3 CreateSpan

CreateSpan(int iHour,int iMinute,int iSecond)

Parameters

iHour

Hours to use, can be zero

iMinute

Minutes to use, can be zero

iSecond

Seconds to use, can be zero

Remarks

The function creates a TLuaDateTime containing not an absolute time but a time span that can be used to add or subtract from another TLuaDateTime object.

8.4 Equal

bool Equal(TLuaDateTime DateTime)

Return values

True if DateTime is equal; otherwise false.

Parameters

DateTime

TLuaDateTime instance obtained from other class function or constructed.

8.5 Get

int Get()

Return values

Number of seconds contained in this instance.

Remarks

Function can be used to retrieve the number of seconds the instance contains, in absolute time.

8.6 GetDate

string GetDate(string sFormat=NULL)

Return values

Returns a string with the current time, formatted as specified by the parameter sFormat, or in the default format.

Parameters

sFormat

Optional string containing an alternate format of the returned time. The default format is YY-MM-DD. See remarks section for flags that can be used.

Remarks

Returns a string with the time contained in the instance, the default format is YY-MM-DD. By suppling your own format code you can alter the way the time is returned.

Format flags

%a - Abbreviated weekday name

%A - Full weekday name

%b - Abbreviated month name

%B - Full month name

%c - Date and time representation appropriate for locale

%d - Day of month as decimal number (01 – 31)

%H - Hour in 24-hour format (00 – 23)

%I - Hour in 12-hour format (01 – 12)

%j - Day of year as decimal number (001 – 366)

%m - Month as decimal number (01 – 12)

%M - Minute as decimal number (00 – 59)

%p - Current locale's A.M./P.M. indicator for 12-hour clock

%S - Second as decimal number (00 – 59)

%U - Week of year as decimal number, with Sunday as first day of week (00 – 53)

%w - Weekday as decimal number (0 – 6; Sunday is 0)

%W - Week of year as decimal number, with Monday as first day of week (00 – 53)

%x - Date representation for current locale

%X - Time representation for current locale

%y - Year without century, as decimal number (00 – 99)

%Y - Year with century, as decimal number

%z, %Z - Time-zone name or abbreviation; no characters if time zone is unknown

8.7 GetTime

string GetTime(string sFormat=NULL)

Return values

Returns a string with the current time, formated as specified by the parameter sFormat, or in the default format.

Parameters

sFormat

Optional string containing an alternate format of the returned time. The default format is HH:MM:SS. See remarks section for flags that can be used.

Remarks

Returns a string with the time contained in the instance, the default format is HH:MM:SS. By suppling your own format code you can alter the way the time is returned.

Format flags

%a - Abbreviated weekday name

%A - Full weekday name

%b - Abbreviated month name

%B - Full month name

%c - Date and time representation appropriate for locale

%d - Day of month as decimal number (01 – 31)

%H - Hour in 24-hour format (00 – 23)

%I - Hour in 12-hour format (01 – 12)

%j - Day of year as decimal number (001 – 366)

%m - Month as decimal number (01 – 12)

%M - Minute as decimal number (00 – 59)

%p - Current locale's A.M./P.M. indicator for 12-hour clock

%S - Second as decimal number (00 – 59)

%U - Week of year as decimal number, with Sunday as first day of week (00 – 53)

%w - Weekday as decimal number (0 – 6; Sunday is 0)

%W - Week of year as decimal number, with Monday as first day of week (00 – 53)

%x - Date representation for current locale

%X - Time representation for current locale

%y - Year without century, as decimal number (00 – 99)

%Y - Year with century, as decimal number

%z, %Z - Time-zone name or abbreviation; no characters if time zone is unknown

8.8 Greater

bool Greater(TLuaDateTime DateTime)

Return values

True if DateTime is less; otherwise false.

Parameters

DateTime

TLuaDateTime instance obtained from other class function or constructed.

8.9 GreaterOrEqual

bool GreaterOrEqual(TLuaDateTime DateTime)

Return values

True if DateTime is less or equal; otherwise false.

Parameters

DateTime

TLuaDateTime instance obtained from other class function or constructed.

8.10 Less

bool Less(TLuaDateTime DateTime)

Return values

True if DateTime is greater; otherwise false.

Parameters

DateTime

TLuaDateTime instance obtained from other class function or constructed.

8.11 LessOrEqual

bool LessOrEqual(TLuaDateTime DateTime)

Return values

True if DateTime is greater or equal; otherwise false.

Parameters

DateTime

TLuaDateTime instance obtained from other class function or constructed.

8.12 NotEqual

bool NotEqual(TLuaDateTime DateTime)

Return values

True if DateTime is not equal; otherwise false.

Parameters

DateTime

TLuaDateTime instance obtained from other class function or constructed.

8.13 Set

Set(int iNewTime)

Parameters

iNewTime

Offset in seconds or absolute time in seconds from 1st of January 1970

Remarks

Function can be used to create an TLuaDateTime instance that will later be added, subtracted or compared with another TLuaDateTime instance.

8.14 Sub

Sub(TLuaDateTime DateTime)

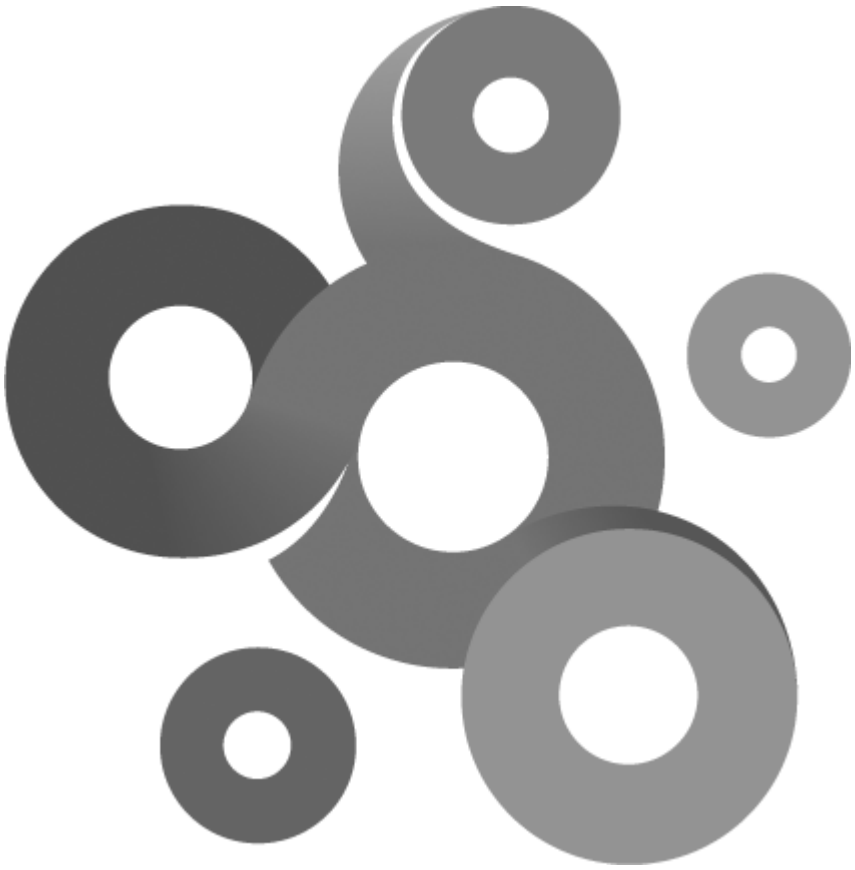
Parameters

DateTime

TLuaDateTime instance obtained from other class function or constructed.

Remarks

The function will subtract the time contained in the DateTime parameter with the time stored in the object.



Section IX

9 TLuaDB

This class provides functions to query SQL databases.

Example

```
-- Create new DB object
DB = TLuaDB();

-- Connect to a DSN
if (DB:Connect("DSN=testdsn;",TLuaDB.CLIENT_ODBC) == true) then

      -- Insert a few rows
      bok = DB:Execute("insert into test (iID,sTest) values(10,'test');");

      -- Select all rows in table
      bok = DB:Execute("select * from test;");
      if ( bok == true) then
            -- Check if we got rows back
            if(DB:ResultAvilable() == true) then
                  -- Print how many columns this table contains
                  iColCount = DB:ColCount();
                  print("Columns in this table: "..iColCount);
                  -- Get first row
                  while (DB:NextRow() == true) do
                        for iCurrentCol = 1, iColCount do
                              -- GetColType and GetCol take a 1 based
index
                              iColType = DB:GetColType(iCurrentCol);
                              sData = DB:GetCol(iCurrentCol);
                              -- Print column #, column type and data
                              print("Col #"..iCurrentCol.." Type: "..
iColType.." Data: "..sData);
                        end
                  end
            end
      else
            -- Print error and exit
            SetExitStatus("Failed" .. DB:GetErrorDescription(),false);
      end

else
      -- Print error and exit
      SetExitStatus("Failed to connect"..DB:GetErrorDescription(),false);
end
```

9.1 ColCount

int ColCount()

Return values

Returns the number of columns in the result from an successful query.

9.2 Connect

bool Connect(string sConnectString,int iClientType=TLuaDB.CLIENT_ODBC)

Return values

True if the connection was successfully executed, false if an error occurred.

Parameters

sConnectString

A client specific connect string. See remarks section for more information

iClientType

Type of client to communicate with, see options below:

Remarks

The connect string is client specific , below are the currently supported clients.

CLIENT_ODBC

Connect string example:

sConnectString = "DSN=test;UID=test;PWD=test";

This connect string uses a datasource named test and supplies the username (UID) and password (PWD) to the DSN.

If no username and password is needed the connect can be formated like this.

sConnectString = "DSN=test;";

KNM is executing as a service, the datasource needs to be a system datasource. This is different from the IDE that can utilize user DSN as well.

CLIENT_SQLSERVER

Connect string example:

sConnectString = "myserver@mydatabase";

To connect to a named instance of SQL Server 2000:

sConnectString = "myserver\\instance_name@mydatabase";

CLIENT_MYSQL

Connect string example:

sConnectString = "myserver";

Connecting to a server listening to a custom port

sConnectString = "myserver:portnumber";

Note that the MySQL client library needs to be installed and included in the default search path so it may be found by KNM and the IDE.

9.3 Connect(2)

bool Connect(string sConnectString,string sUser,string sPassword,int iClientType=TLuaDB.CLIENT_ODBC)

Return values

True if the connection was successfully executed, false if an error occurred.

Parameters

sConnectString

A client specific connect string. See remarks section for more information

sUsername

Credentials to use with the connection.

sPassword

Credentials to use with the connection, cant be empty if username is specified.

iClientType

Type of client to communicate with, currently the only option is CLIENT_ODBC.

Remarks

The connect string is client specific , see Connect (?) 33 for more information.

9.4 Execute

bool Execute(string sSQL)

Return values

True if the query was successfully executed, false if an error occurred.

Parameters

sSQL

A SQL statement.

Remarks

If an error occurs when executing the SQL statement the GetErrorDescription() function will return a string with a description of the error.

9.5 GetCol

string GetCol(int iIndex)

Return values

Returns an string with the retrieved data from the column.

Parameters

iIndex

A 1 based column index.

Remarks

Note that the column index is 1 based. If ColCount() return 10 the valid index range are 1-10. To retrieve the type of the data, call GetColType()

9.6 GetCol_AsDateTime

TLuaDateTime GetCol_AsDateTime(int iIndex)

Return values

Returns an TLuaDateTIme structure with the retrieved date and time from the column.

Parameters

iIndex

A 1 based column index.

Remarks

Note that the column index is 1 based. If ColCount() return 10 the valid index range are 1-10. This function must not be called if the column is not a date time type.

9.7 GetColType

int GetColType(int iIndex)

Return values

Returns an integer representing the type of data the column contains.

Parameters

iIndex

A 1 based column index.

Remarks

The GetCol() function always return the data as an string, the GetColType function can be used to determine the type of the column that conversion of the string can be done after extraction.

The following types exists:

TYPE_BOOL - Boolean value

TYPE_NUMERIC - Numeric

TYPE_SHORT - Short

TYPE_LONG - Long

TYPE_DOUBLE - Double (real)

TYPE_DATETIME - Data/time

TYPE_STRING - String

TYPE_UNKNOWN - Unknown field type / not supported

TYPE_BYTES - Bytes

TYPE_LONG_BINARY - Long binary

TYPE_LONG_CHAR - Long char

TYPE_BLOB - Binary object

TYPE_DBMS_SPECIFIC - Client specific data (no conversion)

9.8 GetErrorDescription

string GetErrorDescription(void)

Return values

Returns a the latest error description generated by the TLuaDB API.

9.9 NextRow

bool NextRow()

Return values

True if a new result was fetched, false if no more results of the query exists.

Remarks

The function retrieves a new result generated by a previous call to the Execute function. This function must be called before the first call to ColCount, GetCol or GetColType functions.

9.10 ResultAvilable

bool ResultAvilable()

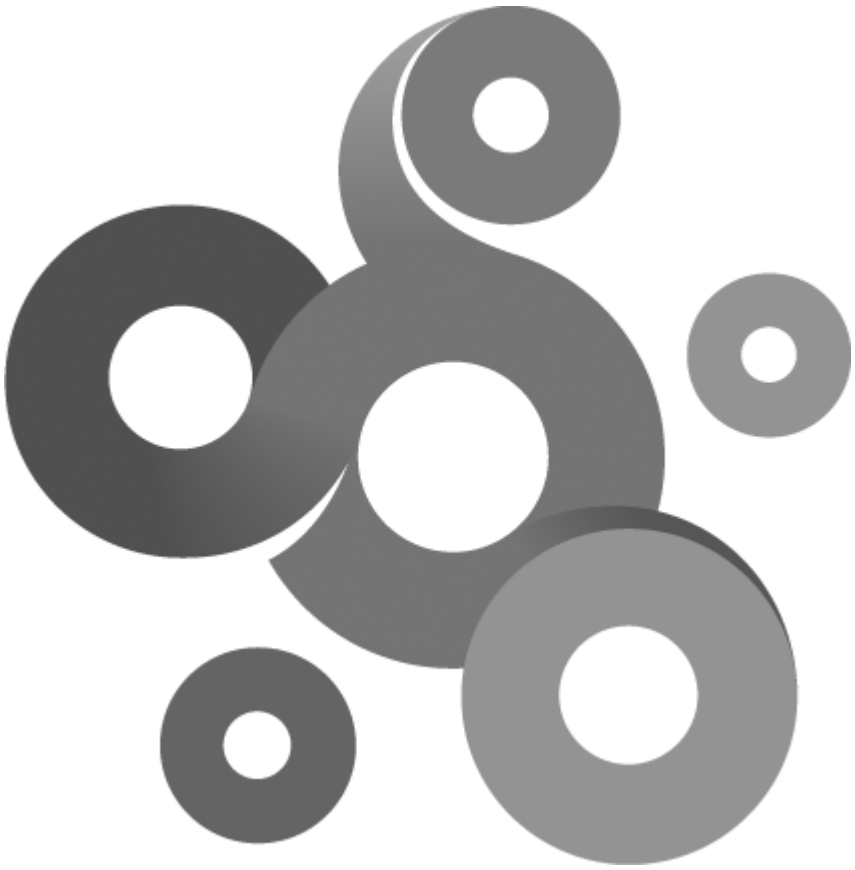
Return values

Returns true if a executed SQL statement returned any result.

Remarks

By design does not insert, update or delete statements return any result back after execution. The return

value from the Execute function should be used to determine if an statement was successfully executed or not before this function is called.



Section X

10 TLuaDNS

The class provides functions for querying DNS servers.

Example

```
DNS = TLuaDNS();
DNS:Begin(true);

if DNS:Query("microsoft.com",TLuaDNS.LuaDNS_TYPE_MX,false) then

      Record = TLuaDNS_MXRecord();

      while (DNS:Next(Record)) do
            print(Record.m_sNameExchange);
      end

      SetExitStatus("Test ok",true);

else
      SetExitStatus("Test failed",false);
end
```

10.1 Begin

bool Begin(bool bUselocalhost=false)

Return values

Non zero if the function is successful; otherwise 0.

Parameters

bUselocahost

Set to true if you wish to query a DNS server on the KNM host machine.

Remarks

The function starts a DNS query. Calling Query before Begin can lead to unknown result.

10.2 End

void End()

Remarks

The function ends a transaction and resets the object so that a new query can be executed. If this function is not called, the result of the next query is unpredictable.

10.3 GetErrorDescription

string GetErrorDescription(void)

Return values

Returns a the latest error description generated by the TLuaDNS API.

10.4 Next

bool Next(TLuaDNS_NSRecord Record);

bool Next(TLuaDNS_CNAMERecord Record);

bool Next(TLuaDNS_ARecord Record);

bool Next(TLuaDNS_PTRRecord Record);

bool Next(TLuaDNS_SOARecord Record);

bool Next(TLuaDNS_MXRecord Record);

bool Next(TLuaDNS_TXTRecord Record);

Return values

Non zero if the function is successful; otherwise 0.

Parameters

Record

DA DNS record type recieving the information queried from the DNS server.

Remarks

The function returns true if it successfully retrieved a new record, if the function returns false there is no more records to be retrieved.

10.5 Query

bool Query(string sDomainName,int iRecordType,bool bBypassCache=false);

Return values

Non zero if the function is successful; otherwise 0.

Parameters

sDomainName

Domain to query

iRecordType

One of the record types listed in the remarks section.

bBypassCache

Default false, if set to true the query will bypass the local resolver and ask the DNS server directly.

Remarks

The function sends a query to the DNS server. The result can be extracted with one or more calls to the Next() function.

The following record types can be queried:

LuaDNS_TYPE_PTR

LuaDNS_TYPE_TEXT

LuaDNS_TYPE_SOA

LuaDNS_TYPE_CNAME

LuaDNS_TYPE_MX

LuaDNS_TYPE_NS

DNS record types reference can be found here:

http://www.iana.org/assignments/dns-parameters

10.6 TLuaDNS_ARecord

Class members

int m_iTTL

string m_sIPAddress;

10.7 TLuaDNS_CNAMERecord

Class members

int m_iTTL

string m_sHostname

10.8 TLuaDNS_MXRecord

Class members

int m_iPreference

string m_sNameExchange

10.9 TLuaDNS_NSRecord

Class members

int m_iTTL

string m_sHostname

10.10 TLuaDNS_PTRRecord

Class members

int m_iTTL

string m_sHostname

10.11 TLuaDNS_SOARecord

Class members

int m_iTTL

string m_sPrimaryServer

string m_sAdministrator

int m_iSerialNo

int m_iRefresh

int m_iRetry

int m_iExpire

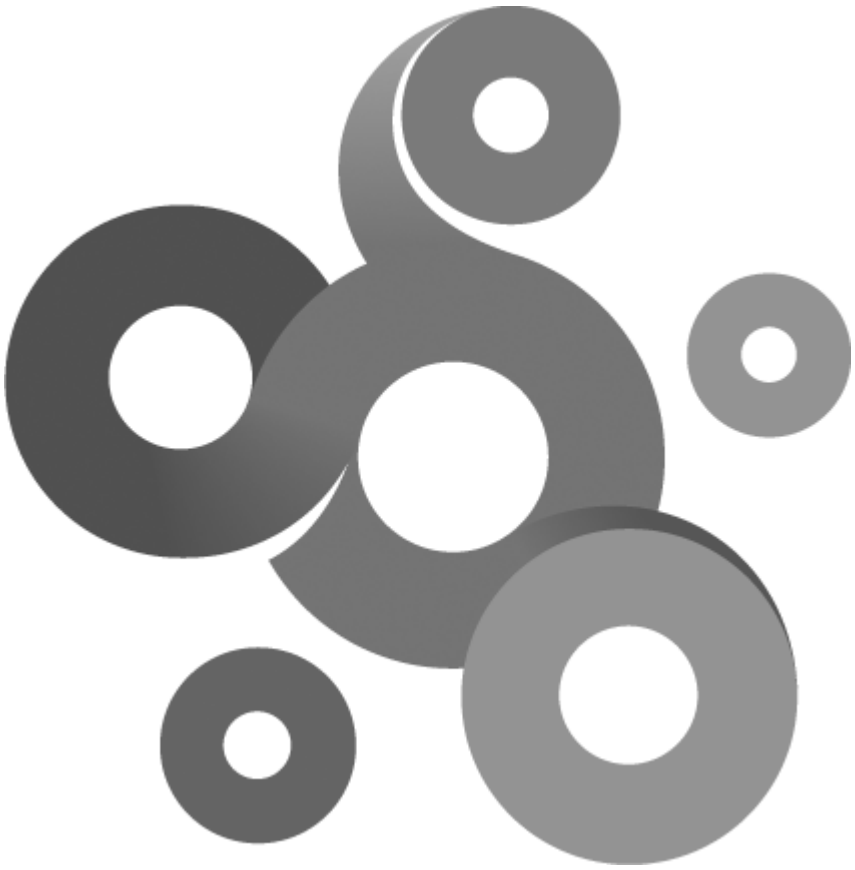
int m_iDefaultTTL

10.12 TLuaDNS_TXTRecord

Class members

int StringCount(void)

string GetString(int iPos)



Section XI

11 TLuaFile

The class provides basic file handling routines and support both binary and text files. All file operations are relative to the context that the script is executed in. For example if the script is executed by an object with the address \\myserver the file path, c:\test.txt, will be translated to \\myserver\C$\test.txt.

There is one exception to this and its when the class is initialized with true, then all file operations are relative the KNM host machine. For more information, see example 3.

Example 1

```
-- Script creates a new text file, writes a string to it and close it.
file = TLuaFile()
-- Open a text file
iRet = file:Open("c:\\test.txt",true)
-- Check if it could be created, it might exist and be write protected
if iRet==0 then
 sErrString = "Failed to create file, error code:"..GetLastError().."\n"
 SetExitStatus(sErrString,false)
else
 print("File created\n")
 -- Write a string to the file
 sString = "Hello world!"
 file:Write(sString,string.len(sString))
 -- Close the file
 file:Close()
 SetExitStatus("Test ok",true)
end
```

Example 2

```
-- Script demonstrates:
-- File enumration
-- Directory enumration
-- Create and delete a directory

-- Construct a new file object
file = TLuaFile();
-- Scan the directory c:\temp for file using the wildcard *.*
sResult = file:GetFileList("c:\\temp","*.*")
print(sResult)
-- Scans the directory c:\temp for sub directories
sResult = file:GetDirectoryList("c:\\temp")
print(sResult)
bResult = false
-- Create a directory called "temp20" on the c: harddisk
if file:CreateDirectory("c:\\temp20") then
 print("Created directory");
 bResult = true
else
 print("Failed to create directory")
end
-- Delete the directory we created above
if file:DeleteDirectory("c:\\temp20") then
 bResult = true
 print("Deleted directory");
else
 print("Failed to delete directory")
end
if bResult then
 SetExitStatus("Test ok",true)
else
 SetExitStatus("Test failed",false
end
```

Example 3

```
------------------------------------------------------------------------
----------------------
```

```
-- KNM Lua API example (C) 2006 Kaseya AB
-- Script creates a new text file, writes a string to it and close it.
-----------------------------------------------------------------------
----------------------
-- Switch the context so that files are opened on the KNM host machine,
not translating the paths
file = TLuaFile(true)
-- Open a text file
iRet = file:Open("c:\\test.txt",true)
-- Check if it could be created, it might exist and be write protected
if iRet==0 then
 sErrString = "Failed to create file, error code:"..GetLastError().."\n"
 SetExitStatus(sErrString,false)
else
 print("File created\n")
 -- Write a string to the file
 sString = "Hello world!"
 file:Write(sString,string.len(sString))
 -- Close the file
 file:Close()
 SetExitStatus("Test ok",true)
end
```

11.1 Close

int Close()

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Remarks

Closes the file and makes the file unavailable for reading or writing. If you have not closed the file before the object is destroyed the destructor will do it for you.

11.2 CopyFile

int CopyFile(string sSource,string sDest)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sSource

Path and name of the file to be copied.

sDest

New path and name of the new file.

Remarks

Function copies a file. Directories can not be copied with this function.

11.3 CreateDirectory

int CreateDirectory(string sPath)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sPath

Path of the directory to be created.

Remarks

The parent directory of the directory to be created must exist otherwise this function will fail.

11.4 DeleteDirectory

int DeleteDirectory(string sDirectory)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sDirectory

Path of the directory to be deleted.

Remarks

The directory must be empty and cannot be a root director or the current working directory.

11.5 DeleteFile

int DeleteFile(string sFileName)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sFileName

Path and name of file to delete.

Remarks

Function deletes a file. Directories can not be deleted with this function.

11.6 DoesFileExist

int DoesFileExist(string sFile);

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sFile

Path and name of the file.

Remarks

Returns a non zero value if the file exist.

11.7 GetDirectoryList

string GetDirectoryList(string sDirectory)

Return values

Returns a string with the sub directories separated with a carriage return; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sDirectory

Path of the directory to be searched.

Remarks

The return string contains all sub directories in the specified directory. Each directory is returned with its full path. Directories in string is separated with a carriage return sign.

11.8 GetFileAccessedTime

TLuaDateTime GetFileCreatedTime(string sFilename)

Return values

Returns a TLuaDateTime object contaning the time the file was last accessed; otherwise a TLuaDateTime contaning 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sFilename

Path and name of the file.

Remarks

Use TLuaDateTime object to construct a string representing the time stored in the object or compare it with another TLuaDateTime object.

11.9 GetFileCreatedTime

TLuaDateTime GetFileCreatedTime(string sFilename)

Return values

Returns a TLuaDateTime object contaning the time the file was created; otherwise a TLuaDateTime contaning 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

pFilename

Path and name of the file.

Remarks

Use TLuaDateTime object to construct a string representing the time stored in the object or compare it with another TLuaDateTime object.

11.10 GetFileList

string GetFileList(string sDirectory,string sWildCard)

Return values

Returns a string with the listed files separated with a carriage return; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sDirectory

Path of the directory.

sWildCard

Wild card to filter out wanted files. To retrieve all files in the directory use the *.* wild card.

Remarks

The return string contains all files in the directory matching the supplied wild card. Each file is returned with its full path. Files in string is separated with a carriage return sign.

11.11 GetFileModifiedTime

TLuaDateTime GetFileCreatedTime(string sFilename)

Return values

Returns a TLuaDateTime object contaning the time the file was last modified; otherwise a TLuaDateTime containing 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sFilename

Path and name of the file.

Remarks

Use TLuaDateTime object to construct a string representing the time stored in the object or compare it with another TLuaDateTime object.

11.12 GetFileSize

int GetFileSize(string sFile)

Return values

Size of file in number of bytes if the function is successful; if the file cannot be accessed the return value is -1.

Parameters

sFile

Path and name of the file.

Remarks

The function is limited to files smaller then 2^31 bytes.

11.13 GetFileSizeMB

int GetFileSizeMB(string sFile)

Return values

Size of file in number of mega bytes if the function is successful; if the file cannot be accessed the return value is -1.

Parameters

sFile

Path and name of the file.

11.14 GetFileStatus

int GetFileStatus(string sFile)

Return values

A value describing the current state of the file if the function is successful; if the file cannot be accessed the return value is -1.

Parameters

sFile

Path and name of the file.

Remarks

The function returns a combination of the following flags to describe the state of the file.

FILE_NORMAL = 0x00

Normal file.

FILE_READONLY = 0x01

File is read only.

FILE_HIDDEN = 0x02

File is hidden.

FILE_SYSTEM = 0x04

File is a system file.

FILE_VOLUME = 0x08

File is a volume.

FILE_DIR = 0x10

File is a directory.

FILE_ARCHIV = 0x20

File have the archive bit set.

11.15 MoveFile

int MoveFile(string sSource,string sDest)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sSource

Path and name of the file to be moved.

sDest

New path and name of the file.

Remarks

Function moves a file. The directory where the file will be moved to must exist or this function will fail. Directories can not be moved with this function.

11.16 Open

int Open(string sFileName, bool bCreate=false)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sFileName

Name and path of the file to open or create.

bCreate

If set to non zero the function will create a file. If the given file already exist its content will be destroyed.

Remarks

If bCreate is set to zero and the file do not exist this function will fail.

11.17 Read

string,int Read(int iSize)

Return values

A string if the function is successful and iSize is set to the size of the returned data; otherwise nil, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

iSize

Size of data to be read in bytes.

Remarks

The function will read the specified size of data from the file and move the file pointer forward the same amount. If end of file is reached the read will stop and return the data read until end of file was reached.

11.18 ReadData

local data,int ReadData(int iSize)

Return values

The function returns a special data type that can only be used in conjunction with the ConvertFromUTF16. If function is successful iSize is set to the size of the returned data; otherwise nil, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

iSize

Size of data to be read in bytes.

Remarks

The function will read the specified size of data from the file and move the file pointer forward the same amount. If end of file is reached the read will stop and return the data read until end of file was reached.

11.19 RenameFile

int RenameFile(string sOrgFile,string sNewFile)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sOrgFile

Path and name of the file to be renamed.

sNewFile

New path and name of the file.

Remarks

Function renames a file. Directories can not be renamed with this function.

11.20 SeekFromCurrent

int SeekFromCurrent(int iNumberofBytes)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

iNumberofBytes

Number of bytes to reposition the file pointer by, relative to its current position.

Remarks

The function will move the file pointer relative its current position. Both negative and positive values can be specified. The pointer can be positioned beyond the end of the file, it will then clear the end of file marker.

11.21 SeekFromEnd

int SeekFromEnd(int iNumberofBytes)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

iNumberofBytes

Number of bytes to reposition the file pointer by, counted from the end of the file.

Remarks

The function will move the current position of the file pointer to iNumberofBytes from the end of the file. Note that the iNumberofBytes must be negative in order to move the file pointer "upwards" in the file.

11.22 SeekFromStart

int SeekFromStart(int iNumberofBytes)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

iNumberofBytes

Number of bytes to reposition the file pointer by, counted from the start of the file.

Remarks

The function will move the current position of the file pointer to iNumberofBytes from the start of the file. The pointer can be positioned beyond the end of the file, it will then clear the end of file marker.

11.23 Write

int Write(string sData,int iSize)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sData

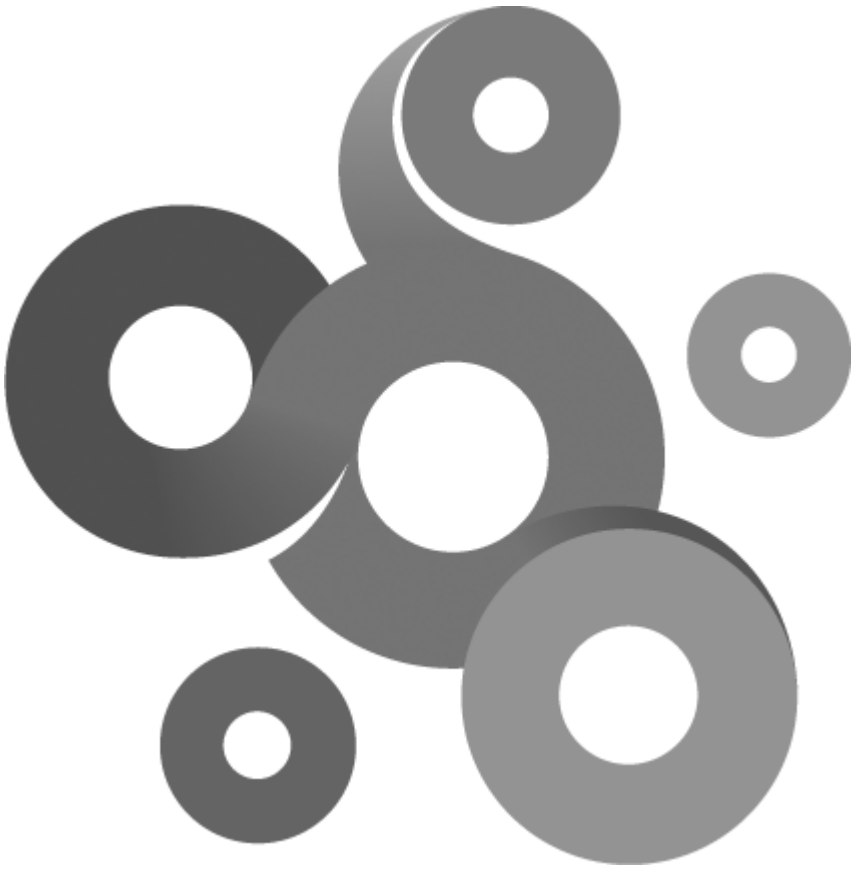
Array of data to be written to the file.

iSize

Size of data to be written.

Remarks

The function will write from the current position of the file pointer, if needed it will extend the file when passing the current end of file. This function will fail if the opened file is write protected.



Section XII

12 TLuaFTPClient

The class implements a FTP client capable of basic FTP operations.

Example

```
-- Script connects to a FTP server and download content of file
ftp = TLuaFTPClient();
-- Enter the username and password for the session
sUsername = "myusername"
sPassword = "mypassword"
-- Connect to FTP server using username and password
iRet = ftp:Connect(sUsername,sPassword,21)
-- Check return value from server
if iRet == 0 then
 -- Failed to connect, print why
 iRet = GetLastError()
 sErrorString = FormatErrorString(iRet)
 sErrString = "Error when connecting to FTP server, error: "..sErrorString
 SetExitStatus(sErrString,false)
else
 -- Open a file on the FTP server that we know exist
 sFilename = "update.vcf"
 -- Open file, do not create it, use text mode
 iRet = ftp:OpenFile(sFilename,false,true)
 if iRet == 0 then
  sErrString = "Cannot open file "..sFilename
  SetExitStatus(sErrString,false)
 else
  iMaxSize = 1024*16
  -- Read a number of bytes from the file
  -- Note here that we are using the special lua return value convention
  -- Read returns one string and the size of the string
  sFilecontent, iMaxSize = ftp:Read(iMaxSize)
  print("Size of content: "..iMaxSize.."\n")
  print(sFilecontent)
  -- Close file so we can open a new file later or close the session
  ftp:CloseFile()
  SetExitStatus("Test ok",true)
 end
end
-- Close FTP session
ftp:Close()
```

12.1 ChangeDirectory

int ChangeDirectory(string sDir)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sDir

Path of new current directory.

Remarks

This function will fail if the directory do not exist.

12.2 Close

Close()

Remarks

The function closes the current connection to the FTP server. The function must be called to close the current connection.

12.3 CloseFile

void CloseFile()

Remarks

Closes an file opened with OpenFile.

12.4 Connect

int Connect(string sUsername,string sPassword,unsigned int iPort=21)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sUsername

String contaning the username,

sPassword

String contaning the password.

iPort

Default port 21 (standard FTP port).

Remarks

The function connects to a FTP server using the provided username and password. The port can be changed from the default port 21 if the FTP server is bound on another port.

12.5 CreateDirectory

int CreateDirectory(string sDir)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sDir

Path of directory to create.

Remarks

The directory which the new directory is to be created in must exist or this function will fail.

12.6 DeleteDirectory

int DeleteDirectory(string sDir)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sDir

Path of directory to delete.

Remarks

If the directory is not empty the function will fail.

12.7 DeleteFile

int DeleteFile(string sFilename)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sFilename

Path and name of file to be deleted.

Remarks

This function will fail if the file do not exist.

12.8 FindDirectory

string FindDirectory()

Return values

Returns a string with the listed directories separated with a carriage return; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Remarks

The return string contains all sub directories of the current directory. Each directory is returned with its full path. Directories in the string are separated with a carriage return sign.

12.9 FindFile

string FindFile(string sWildcard)

Return values

Returns a string with the listed files separated with a carriage return; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sWildCard

Wild card to filter out wanted files. To retrieve all files in the directory use the *.* wild card.

Remarks

The return string contains all files in current directory matching the supplied wild card. Each file is returned with its full path. Files in string is separated with a carriage return sign.

12.10 GetCurrentDirectory

int ChangeDirectory(string sDir)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sDir

Path of new current directory.

Remarks

This function will fail if the directory do not exist.

12.11 GetFileModifiedTime

TLuaDateTime GetFileModifiedTime(string sFilename)

Return values

A TLuaDateTime object containing the time of the last modification of the file if the function is successful; otherwise a TLuaDateTime object set to 0,

Parameters

sFilename

Name of file in current directory.

Remarks

The file must be in the current directory for this function to success, relative paths will not work.

12.12 GetFileSize

int GetFileSize(string sFilename)

Return values

Size of file in number of bytes if the function is successful; if the file cannot be accessed the return value is -1.

Parameters

sFilename

Path and name of file.

Remarks

The function is limited to files smaller then 2^31 bytes.

12.13 OpenFile

int OpenFile(string sFilename,bool bWrite,bool bText)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sFilename

Name of file to open.

bWrite

If non zero the file is opened in write mode. If zero the file is opened for read only.

bText

If non zero the file is opened in text translation mode; otherwise it is opened in binary mode.

Remarks

The function opens a file on the FTP server, after the file is opened calls to Write and Read function can

be made. The CloseFile function is used to close the file.

12.14 Read

string Read(int iSize)

Return values

An array with the data read from the file; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

iSize

Size to read from the file, when function returns it will contain how much was read that can be less then the requested size.

Remarks

This function will fail if a file have not been opened.

12.15 RenameFile

bool RenameFile(string sOldname,string sNewname)

Return values

True if successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sOldname

Old name of file to rename.

sNewname

New name of the file.

Remarks

The function will fail if there exists a file with the same name as supplied in the second parameter.

12.16 Write

int Write(string sData,int iSize)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sData

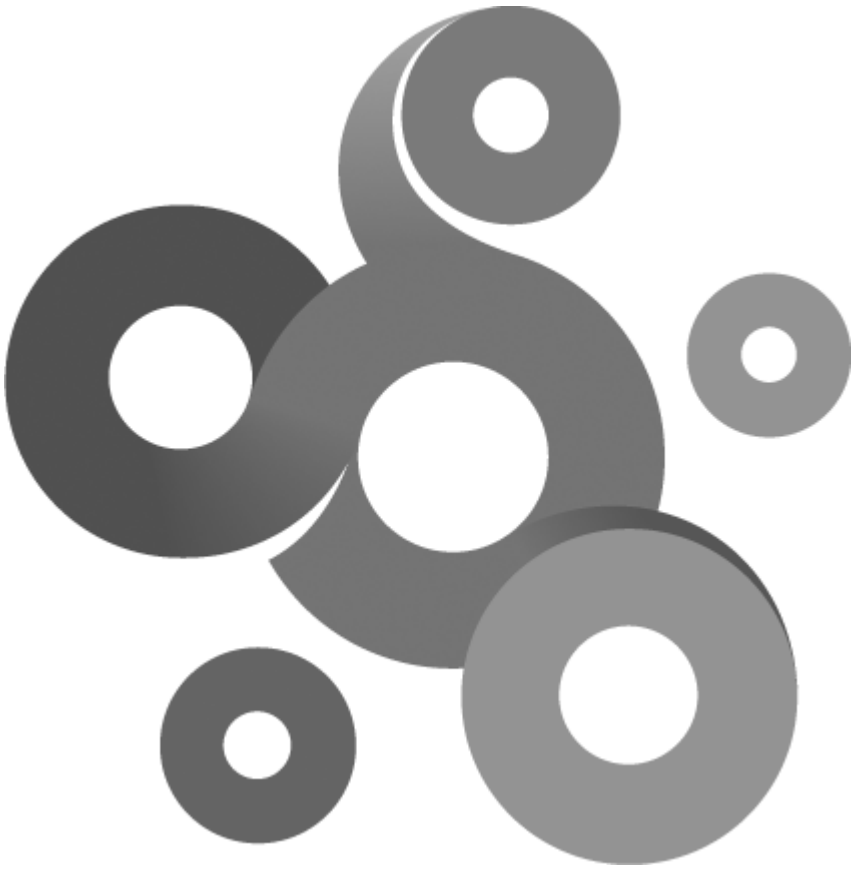
Array of data to be written to file.

iSize

Size of array to be written.

Remarks

This function will fail if a file have not been opened or if the file is opened in read only mode. The function can also fail if the storage on the FTP server is exhausted.



Section XIII

13 TLuaHTTPClient

The class implements a basic HTTP client.

Example

```
-------------------------------------------------------------------------------
----------------------
-- Script to download a html page from a web server
-------------------------------------------------------------------------------
----------------------
http = TLuaHTTPClient()
-- Connect using the default parameters
iRet = http:Connect()
if iRet ~= 0 then
 -- Make a GET request to default document
 iRet = http:Get("/")
 -- Print returned code from HTTP server
 print("Code:"..iRet)
 -- Extract content length
 iRet = http:GetHeaderContentLength()
 print("Content length:"..iRet)
 -- Print content
 string,iRet = http:GetContent(iRet)
 print(string)
 -- Print raw headers
 string = http:GetHeadersRaw()
 print("headers:\n"..string)
 -- Print cookies
 string = http:GetHeaderCookies()
 print("Cookies:\n"..string)
 -- Extract and print cookies one by one
 iNumber = http:GetHeaderCookieCount()
 for count = 0, iNumber-1 do
  string = http:GetHeaderCookie(count)
  print("Coookie #"..count.." "..string.."\n")
 end
 -- Extract location header
 string = http:GetHeaderLocation();
 print("location:\n"..string)
 SetExitStatus("Test ok",true)
else
 print("Connect failed")
 SetExitStatus("Test failed",false)
end
```

13.1 Connect

int Connect(unsigned int iPort=80,bool bSecure=false,string sUsername=NULL,string sPassword=NULL)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

iPort

Port number to connect to, default port 80

bSecure

Set to non zero if connection is to be established using HTTPS

sUsername

Optional username for servers requiring authentication

sPassword

Optional password for servers requiring authentication

Remarks

The function connects to a HTTP server with the supplied parameters. This function must be called before any other function in this class is called.

13.2 Close

Close()

Remarks

Closes an open connection.

13.3 Get

int Get(string sUrl,string sHeaders=NULL)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sUrl

URL relative to the base url of the site.

sHeaders

Optional header string containing headers to be sent with the request.

Remarks

The connection is always opened in the context of the object, therefore the URL supplied to the function must be relative the base URL.

13.4 Post

int Post(string sUrl,string sHeaders=NULL,string sData=NULL)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sUrl

URL relative to the base URL of the site.

sHeaders

Optional header string containing headers to be sent with the request.

sData

Optional data to include in post request.

Remarks

The connection is always opened in the context of the object, therefore the URL supplied to the function must be relative the base URL. Each header supplied must be ended with a CR/LF pair.

13.5 GetContent

string GetContent(int iSize)

Return value

If successful a string containing the content part of a GET request; otherwise nil, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

iSize - Set to size of the returned string when function returns.

Remarks

The content refers to the data returned by a request that follows the header.

13.6 GetHeadersRaw

string GetHeadersRaw()

Return values

If successful a string containing the headers returned by the request; otherwise an empty string.

Remarks

The headers are returned exactly the way they are sent by the server.

13.7 GetHeaderLocation

string GetHeaderLocation()

Return values

If successful a string containing the "Location:" header; otherwise an empty string.

13.8 GetHeaderContentLength

int GetHeaderContentLength()

Return values

The length in bytes of the content part of the request.

13.9 GetHeaderContentType

string GetHeaderContentType()

Return value

If successful a string containing the "Content-Type:" header; otherwise an empty string.

13.10 GetHeaderContentTransferEncoding

string GetHeaderContentTransferEncoding()

Return value

If successful a string containing the "Transfer-Encoding:" header; otherwise an empty string.

13.11 GetHeaderCookies

string GetHeaderCookies()

Return value

If successful a string containing all cookies returned by the server separated with carriage return;

otherwise an empty string.

13.12 GetHeaderCookie

string GetHeaderCookie(int iIndex)

Return value

A string with the requested cookie string.

Parameters

iIndex

A zero based index specifying the cookie to return.

Remarks

If a negative number or a index out of range is specified an empty string will be returned.

13.13 GetHeaderCookieCount

int GetHeaderCookieCount()

Return value

The number of cookies returned by the request.

13.14 GetHeaderDate

string GetHeaderDate()

Return value

If successful a string containing the "Date:" header; otherwise an empty string.

13.15 GetHeaderExpires

string GetHeaderExpires()

Return value

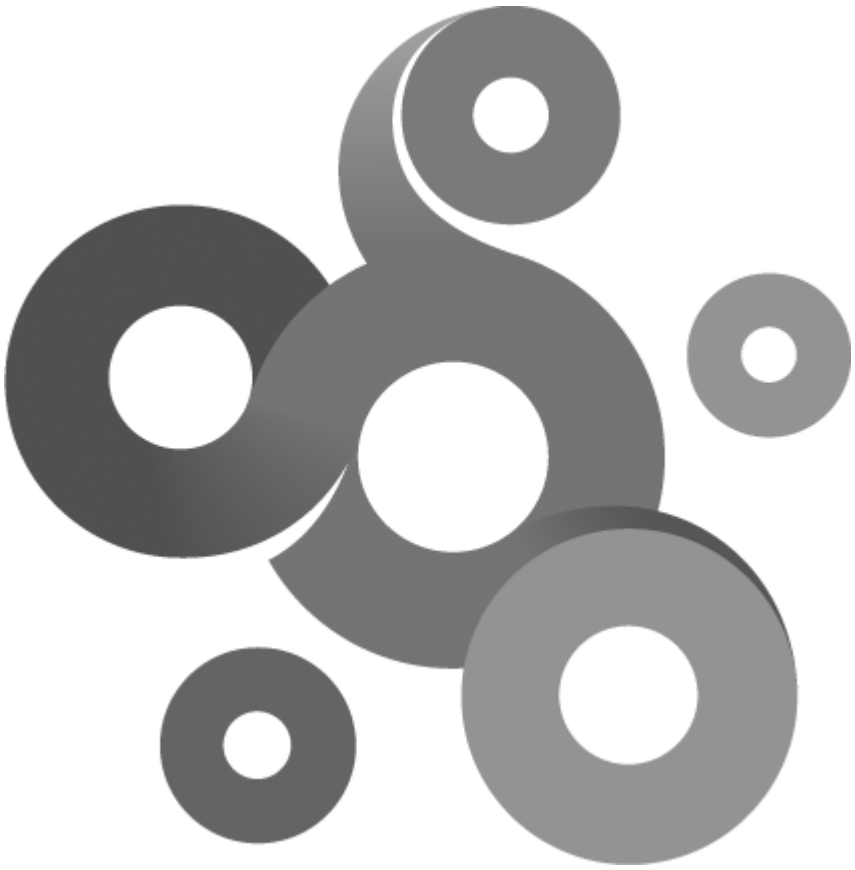
If successful a string containing the "Expires:" header; otherwise an empty string.

13.16 GetHeaderHost

string GetHeaderExpires()

Return value

If successful a string containing the "Host:" header; otherwise an empty string.



Section XIV

14 TLuaICMP

The class provides ping and trace route functions that can be used to diagnose a network connection.

```
-------------------------------------------------------------------------------
--------------------------------
-- Description: Trace route example
-------------------------------------------------------------------------------
--------------------------------
icmp = TLuaICMP()

iPacketSize = 32              -- packet size in bytes, excluding the
header
bNoFragment = false           -- Set to true to inhibit fragmentation of
packet sent
iMaxHops = 255                -- Max number of hops in route
-- Begin trace
bok = icmp:BeginTrace(iPacketSize,bNoFragment,iMaxHops)
if bok ~= true then
      SetExitStatus("Trace failed!",false);
end

-- Print the result
iCount = 1;
Result = TLuaICMPTraceResult()
while icmp:NextTraceResult(Result) do

      print("Hop: "..iCount)
      print(Result.m_Name)
      print(Result.m_iRoundTripTimeMs)
      iCount = iCount + 1
end

-- Clean up resources
icmp:EndTrace()

SetExitStatus("Trace ok!",true);
```

14.1 BeginTrace

bool BeginTrace(int iPacketsToSend,int iPacketSize,bool bNoFragment,int iTimeoutMs)

Return values

The function returns true if the trace was successful, or false if it failed.

Parameters

iPacketsToSend

A positive integer between 1 and 255

iPacketSize

Size of packets to send, a integer between 0 and

bNoFragment

Set to true to stop sent packets from being fragmented, function will fail and return false if option is set and packet is fragmented.

iTimeoutMs

Max time in ms that the function will wait for packet to be returned.

Remarks

By setting the bNoFragment to true its possible to test the largest frame size for a route, adjust the iPacketSize until the function fails due to fragmentation.

14.2 EndTrace

EndTrace()

Remarks

The function performs clean up of used resources. Must be called for each BeginTrace() call.

14.3 NextTraceResult

bool NextTraceResult(TLuaICMPTraceResult Result)

Return values

The function returns true while a result is available.

Parameters

Result

A TLuaICMPTraceResult 69 variable that receives the result for the current hop.

Remarks

To iterate over the result set, call the function until null is returned.

14.4 Ping

bool Ping(TLuaICMPPingResult Result,int iPacketsToSend,int iPacketSize,bool bNoFragment,int iTimeoutMs)

Return values

The function returns a TLuaICMPPingResult object containing the result of the operation.

Parameters

iPacketsToSend

A positive integer between 1 and 255

iPacketSize

Size of packets to send, a integer between 0 and

bNoFragment

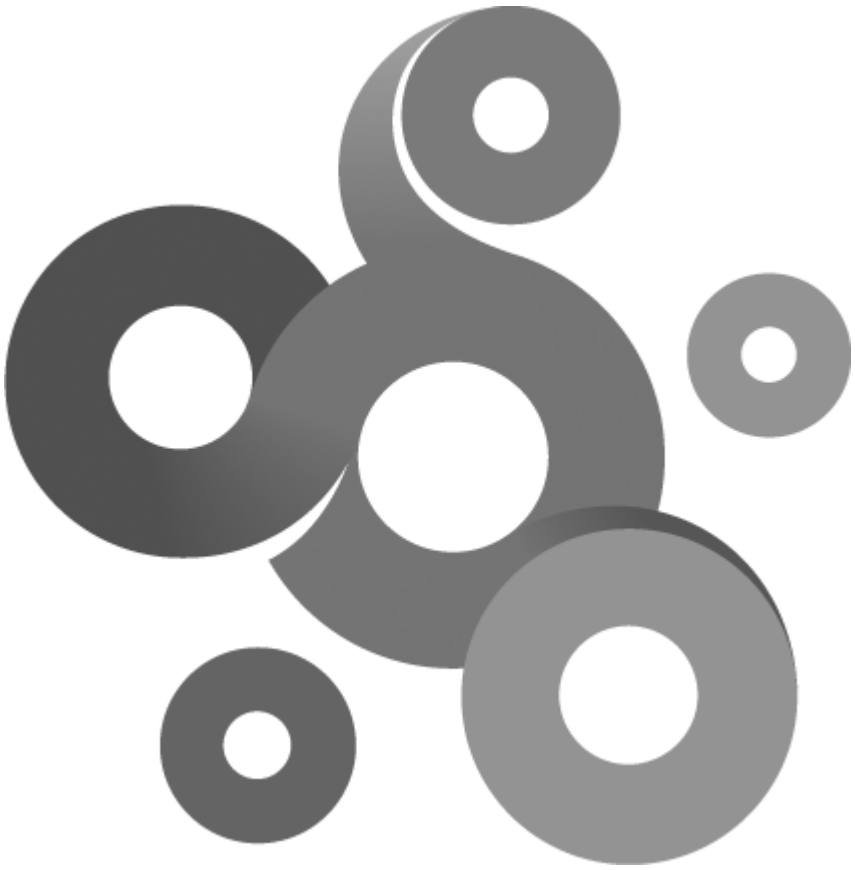
Set to true to stop sent packets from being fragmented, function will fail and return false if option is set and packet is fragmented.

iTimeoutMs

Max time in ms that the function will wait for packet to be returned.

Remarks

By setting the bNoFragment argument makes it possible to test the biggest possible frame size for a route.



Section XV

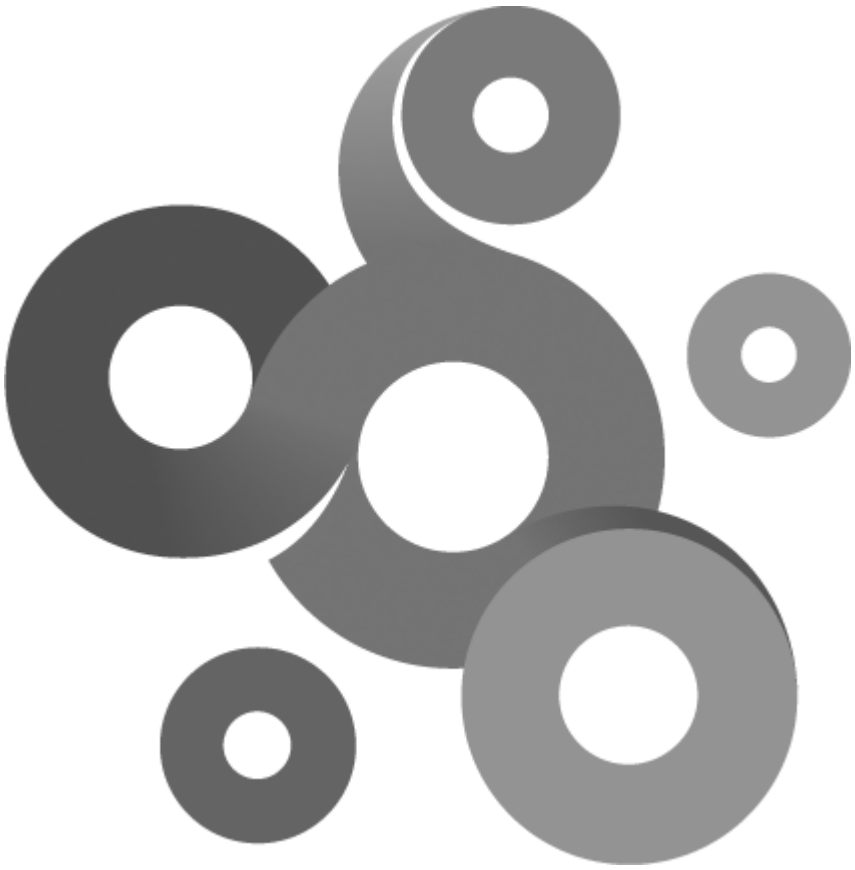
15 TLuaICMPPingResult

The TLuaICMPPingResult is a read only class.

Class members

int m_iRoundTripTimeMs

float m_fPacketloss



Section XVI

16 TLuaICMPTraceResult

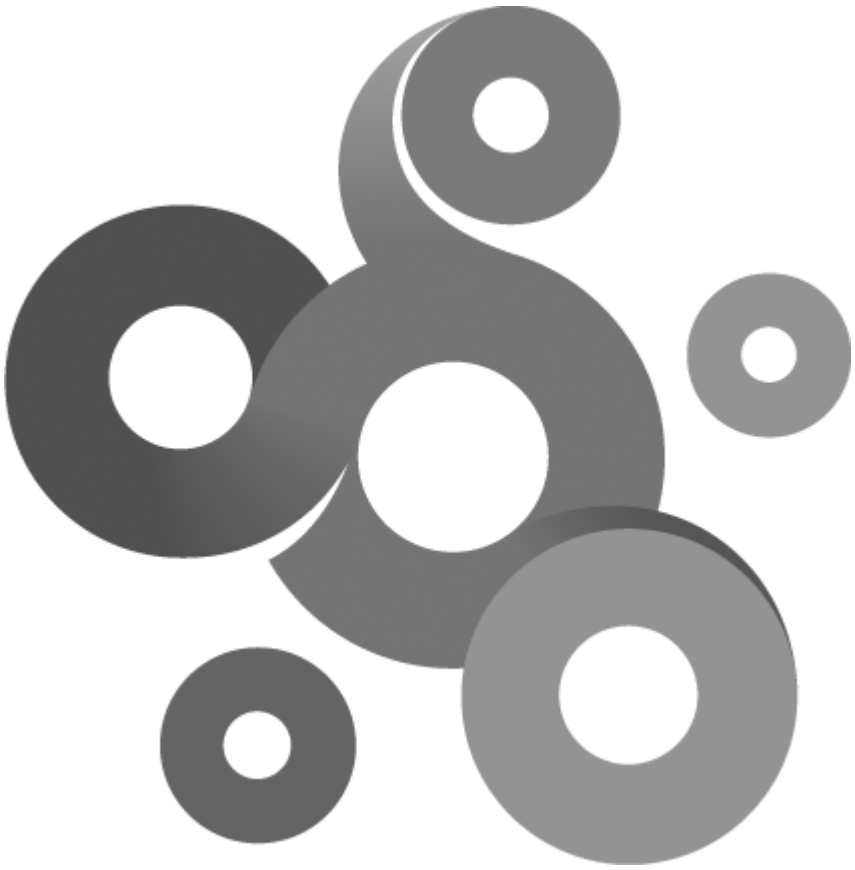
The TLuaICMPTraceResult is a read only class.

Class members

string m_IP

string m_Hostname

int m_iRoundTripTimeMs



Section XVII

17 TLuaRegistry

The class provides access to the Windows registry. When working with the registry there is two important terms that are used in the documentation.

Key - A entity in the registry hive that can contain child keys and values.

Value - A entity without child entries that contains data. The data can be of different types, types supported by this implementation is string, integer and binary data.

All registry operations are relative to the context that the script is executed in. There is one exception to this and its when the class is initialized with true, then all operations are relative the KNM host machine. For more information, see example 2.

Example 1

```
--------------------------------------------------------------------------------
-------------------------------
-- Demonstrates the Lua Windows Registry interface
--------------------------------------------------------------------------------
-------------------------------

-- Open the registry on the host determined by the current context
Reg = TLuaRegistry();

if Reg:Open(Reg.LOCAL_MACHINE,"SOFTWARE\\Kaseya") == true then
      sValue = "";
      bOK,sValue = Reg:ReadValue("test",sValue);
else
      Reg:SetValue("test","a test value");
end
```

Example 2

```
--------------------------------------------------------------------------------
-------------------------------
-- Demonstrates the Lua Windows Registry interface
--------------------------------------------------------------------------------
-------------------------------

-- Open the registry on the localhost
Reg = TLuaRegistry(true);

if Reg:Open(Reg.LOCAL_MACHINE,"SOFTWARE\\Kaseya") == true then
      sValue = "";
      bOK,sValue = Reg:ReadValue("test",sValue);
else
      Reg:SetValue("test","a test value");
end
```

17.1 BeginEnumValue

BeginEnumValue()

Remarks

The function should be called before the first call to EnumValue(). The function ensures that EnumValue () starts at the top of the value list. Failure to call this function before EnumValue() will give unpredictable results.

17.2 Close

Close()

Remarks

The function closes the current open registry connection.

17.3 Create

bool Create(string sKey)

Return values

Non zero if the function is successful; otherwise 0, and a error description can be retrieved by calling GetErrorDescription().

Parameters

sKey

A key that will be created.

Remarks

The Create() function creates the specified registry key. If the key already exists in the registry, the function opens it. The function can be used to create several keys at once. For example, a script can create a sub-key three levels deep by specifying a string in the following form:

subkey1\subkey2\subkey3

17.4 DeleteValue

bool DeleteValue(string sValueName)

Return values

Non zero if the function is successful; otherwise 0, and a error description can be retrieved by calling GetErrorDescription().

Parameters

sValueName

Name of a value that will be deleted.

Remarks

The function deletes a value in the current key, if the value does not exist this function will fail.

17.5 EnumValue

bool EnumValue(string &sValueName)

Return values

Non zero if the function is successful; otherwise 0, and a error description can be retrieved by calling GetErrorDescription().

Parameters

sValueName

Name of the next enumerated value in the current key.

Remarks

Call this function until the it returns false to enumerate all values in the current key. Before this function is called the first time, a call to BeginEnumValue() must be made.

Example 1

```
-----------------------------------------------------------------------
-------------------------------
-- KNM Lua API example (C) 2007 Kaseya AB
```

```
-- Demonstrates the Lua Windows Registry interface
-------------------------------------------------------------------------------

Reg = TLuaRegistry();
-- Open the key to enumerate
if Reg:Open(Reg.LOCAL_MACHINE,"SOFTWARE\\Kaseya") == true then
      Reg:BeginEnumValue();
      bOk = true;
      repeat
            sValue = "";
            bOk,sValue = Reg:EnumValue(sValue);
            if bOk then print(sValue); end;
      until bOk == false;
else
      print("Failed to open the key");
end
```

17.6 GetErrorDescription

string GetErrorDescription()

Return values

Returns a string describing the latest error encountered when calling any function in the class.

17.7 Open

bool Open(int iKey,string sKey)

Return values

Non zero if the function is successful; otherwise 0, and a error description can be retrieved by calling GetErrorDescription().

Parameters

iKey

A key that represents one of the registry hives.

bCreate

A sub-key in the selected registry hive.

Remarks

The function opens a registry key in the selected registry hive. Note that the credentials of the process (either the IDE or KNM) can restrict access to certain keys and hives.

The following constants are defined for iKey:

CLASSES_ROOT

CURRENT_CONFIG

CURRENT_USER

LOCAL_MACHINE

PERFORMANCE_DATA

USERS

17.8 ReadValue

bool ReadValue(string sValueName,string &sData)

Return values

Non zero if the function is successful; otherwise 0, and a error description can be retrieved by calling GetErrorDescription().

Parameters

sValueName

Name of value to retrieve.

sData

Data returned by the function.

Remarks

The function returns the data of the value with the specified name. If the value type is not a string this function will fail.

17.9 ReadValue

bool ReadValue(string sValueName,int &iData)

Return values

Non zero if the function is successful; otherwise 0, and a error description can be retrieved by calling GetErrorDescription().

Parameters

sValueName

Name of value to retrieve.

iData

Data returned by the function.

Remarks

The function returns the data of the value with the specified name. If the value type is not a integer this function will fail.

17.10 ReadValue

string ReadValue(string sValueName,int &iSize)

Return values

Data stored in the registry value, if the function fails an empty string is returned. A error description can be retrieved by calling GetErrorDescription().

Parameters

sValueName

Name of value to retrieve.

iSize

Size of data returned by the function, in bytes.

Remarks

The function returns the data of the value with the specified name. If the value type is not a integer this function will fail. The size of the data returned is stored in the iSize parameter. If this function fails the iSize parameter is set to zero.

17.11 SetValue

bool SetValue(string sValueName,string sData,int iDataSize)

Return values

Non zero if the function is successful; otherwise 0, and a error description can be retrieved by calling GetErrorDescription().

Parameters

sValueName

Name of value to write.

sData

Data to be written to the value.

iSize

Size of data to write, in bytes.

Remarks

The function writes the specified data to the value.

17.12 SetValue

bool SetValue(string sValueName,string sString)

Return values

Non zero if the function is successful; otherwise 0, and a error description can be retrieved by calling GetErrorDescription().

Parameters

sValueName

Name of value to write.

sString

String to be written to the value.

iSize

Size of data to write, in bytes.

Remarks

The function writes the specified string to the value. If the value does not exist this function will fail.

17.13 SetValue

bool SetValue(string sValueName,int iValue)

Return values

Non zero if the function is successful; otherwise 0, and a error description can be retrieved by calling GetErrorDescription().

Parameters

sValueName

Name of value to write.

iValue

Integer to be written to the value.

Remarks

The function writes the specified integer to the value. If the value does not exist this function will fail.

17.14 SetValueExpandedString

bool SetValueExpandedString(string sValueName,string sString)

Return values

Non zero if the function is successful; otherwise 0, and a error description can be retrieved by calling GetErrorDescription().

Parameters

sValueName

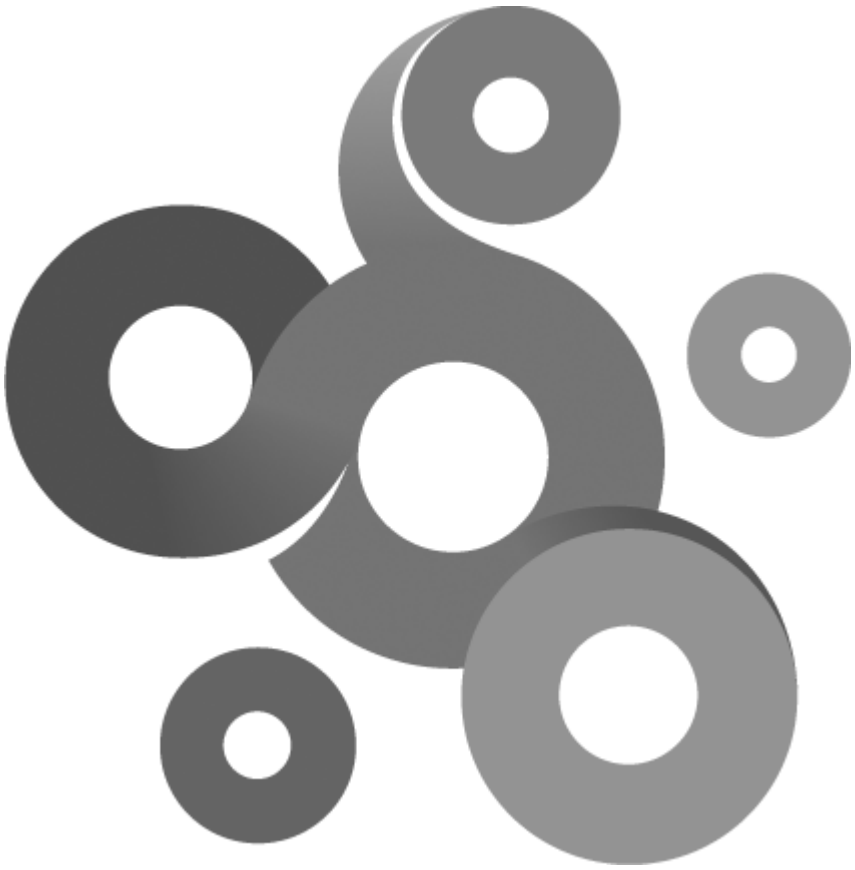
Name of value to write.

sString

String to be written to the value.

Remarks

The function works like the normal SetValue function with one exception. The string written can contain unexpanded references to environment variables (for example, "%PATH%").



Section XVIII

18 TLuaSFTPClient

The class implements a basic SFTP client class.

Example

```
-------------------------------------------------------------------------------
-------------------------------
-- Demonstrates the Lua SFTP client class
-------------------------------------------------------------------------------
-------------------------------

-- Create the client object
sftp = TLuaSFTPClient()

-- Connect to the remote SFTP server
if sftp:Connect("username","password") == false then
      SetExitStatus("No respons",false)
      return
end

-- Create a directory handle and open the current directory
hHandle = TLuaSFTPClientDirectoryHandle()
bOk = sftp:OpenDir(".",hHandle)
if bOk == true then

      -- List the directory
      bOk = sftp:ListDir(hHandle)
      if  bOk == false then
            SetExitStatus("Cannot list directory",false)
            sftp:CloseDir(hHandle)
            return
      end

      -- Loop over the entries in the directory
      File = TLuaSFTPClientFile()
      while hHandle:Next(File) ~= false do
            -- Print the file name
            print(File.m_sFilename)
      end
end
-- Close handle
sftp:CloseDir(hHandle)
```

18.1 Close

bool Close(TLuaSFTPClientHandle FileHandle)

Return value

Returns true if the operation was successful or false if otherwise.

Parameters

FileHandle

Handle of previously opened file.

18.2 CloseDir

bool CloseDir(TLuaSFTPClientDirectoryHandle Handle);

Return value

Returns true if the operation was successful or false if otherwise. On a successful operation the TLuaSFTPClientDirectoryHandle handle is closed.

Parameters

Handle

Handle opened by the OpenDir 80 function.

18.3 Connect

bool Connect(string sUsername,string sPassword,int iPort=22,int iTimeout=25000)

Return value

Returns true if the connect operation succeeded or false otherwise.

Parameters

sUsername

Username

sPassword

Password

iPort

Port number where the server listens. Default value 22.

iTimeout

Timeout in milliseconds to wait for server to respond. Default value 25000 (25 seconds).

18.4 CreateFile

bool CreateFile(string sPath,TLuaSFTPClientHandle hHandle)

Return value

Returns true if the file was created or false if the operation failed. The TLuaSFTPClientHandle contains a reference to the open if the operation succeeded.

Parameters

sPath

Full path of file to create. Directories included in path must exist or the operation fails

hHandle

Handle to create file.

Remarks

The newly created file have read write access rights.

18.5 ListDir

bool ListDir(TLuaSFTPClientDirectoryHandle Handle);

Return value

Returns true if the operation was successful or false if otherwise. On a successful operation data is ready

for retrieval in the TLuaSFTPClientDirectoryHandle 88 class.

Parameters

Handle

Handle opended by the OpenDir 80 function.

18.6 MkDir

bool MkDir(string sPath)

Return value

Returns true if the operation was successful or false if otherwise.

Parameters

sPath

Path to directory to create, including name of new directory.

Remarks

This function is not able to recursively create new directives, all parent directories of the last directory in the path must exists.

18.7 OpenDir

bool OpenDir(string sPath,TLuaSFTPClientDirectoryHandle &Handle)

Return value

Returns true if the operation was successful or false if otherwise.

Parameters

sPath

Path to directory to open.

Handle

Handle returned to be used in subsequent operations.

Remarks

This function "opens" a directory for the purpose of list its content with the ListDir 79 function.

18.8 Open_ForRead

bool Open_ForRead(string _sPath,TLuaSFTPClientHandle hHandle)

Return value

Returns true if the file was opened successfully or false if the operation failed.

Parameters

sPath

Full path of file.

hHandle

Handle to file that is used in subsequent operations.

18.9 Open_ForWrite

bool Open_ForWrite(string _sPath,TLuaSFTPClientHandle hHandle)

Return value

Returns true if the file was opened successfully or false if the operation failed.

Parameters

sPath

Full path of file.

hHandle

Handle to file that is used in subsequent operations.

18.10 Open_ForAppend

bool Open_ForAppend(string _sPath,TLuaSFTPClientHandle hHandle)

Return value

Returns true if the file was opened successfully or false if the operation failed.

Parameters

sPath

Full path of file.

hHandle

Handle to file that is used in subsequent operations.

Remarks

Open_ForAppend opens the file in write mode, the difference between this function and the Open_ForWrite is that all data is written to the end of the file even if the file pointer would been repositioned between two writes.

18.11 Read

bool Read(TLuaSFTPClientHandle FileHandle,int iOffset,int iLen,string &sData)

Return value

Returns true if the operation was successful or false if otherwise.

Parameters

FileHandle

Handle of previously opened file.

iOffset

Offset in bytes where to read in file.

iLen

Length of data to read

sData

Variable to put data.

Remarks

Only text files can be read with this function.

Example

```
-------------------------------------------------------------------------------
--------------------------------
-- KNM Lua API example (C) 2010 Kaseya AB
-- Demonstrates the Lua SFTP client class
-------------------------------------------------------------------------------
```

```
--------------------------------

sftp = TLuaSFTPClient()
hFileHandle = TLuaSFTPClientHandle()
-- Open the file
bOk = sftp:Open_ForRead("test.txt",hFileHandle)
if bOk == false then
      SetExitStatus("Open failed",false)
      return
end

sTemp = ""
-- Read the first 20 bytes
bOk,sTemp = sftp:Read(hFileHandle,0,20,sTemp)
if bOk == false then
      SetExitStatus("Read failed",false)
      return
end
print(sTemp)
```

18.12 Remove

bool Remove(string sPath);

Return value

Returns true if the operation was successful or false if otherwise.

Parameters

sPath

Path of file to be removed.

18.13 Rename

bool Rename(string sPath,string sNewPath);

Return value

Returns true if the operation was successful or false if otherwise.

Parameters

sPath

Path of existing file to be renamed.

sNewPath

Path with new file name.

18.14 RmDir

bool RmDir(string sPath)

Return value

Returns true if the operation was successful or false if otherwise.

Parameters

sPath

Path to directory to delete.

Remarks

This function can only delete empty directories.

18.15 Write

```
bool Write(TLuaSFTPClientHandle FileHandle,const int iOffset,string vData)
```

Return value

Returns true if the operation was successful or false if otherwise.

Parameters

FileHandle

Handle of previously opened file.

iOffset

Offset in bytes where to write in file.

sData

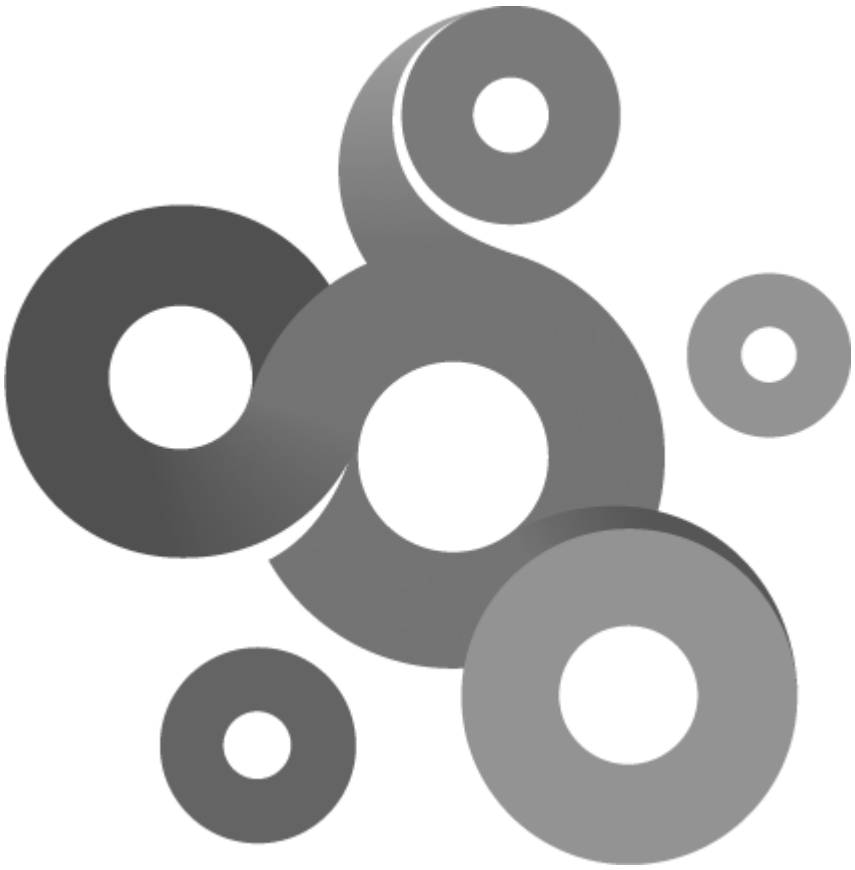
String of text to write.

Example

```
-------------------------------------------------------------------------------
-------------------------------
-- KNM Lua API example (C) 2010 Kaseya AB
-- Demonstrates the Lua SFTP client class
-------------------------------------------------------------------------------
-------------------------------

sftp = TLuaSFTPClient()
hFileHandle = TLuaSFTPClientHandle()
-- Open the file
hFileHandle = TLuaSFTPClientHandle()
if sftp:Open_ForWrite("test.txt",hFileHandle) == false then
      SetExitStatus("Open of file failed",false)
      return
end

-- Create a string and write it to the begining of the file
sString = [[ test text  ]];
if sftp:Write(hFileHandle,0,sString) == false then
      SetExitStatus("Write failed",false)
      return
end
-- Close the file
sftp:Close(hFileHandle)
```



Section XIX

19 TLuaSFTPClientAttributes

The class contains attributes describing a directory or file retrieved by the ListDir 79 function.

19.1 AccessedTime

bool AccessedTime(TLuaDateTime &Time)

Return value

Returns true if the value is present or false if otherwise.

Parameters

Time

Contains the time the file was last accessed.

19.2 CreatedTime

bool CreatedTime(TLuaDateTime &Time)

Return value

Returns true if the value is present or false if otherwise.

Parameters

Time

Contains the time the file was created.

19.3 Group

bool Group(string &sGroup)

Return value

Returns true if the value is present or false if otherwise.

Parameters

sOwner

Contains name of the group of the file or directory.

19.4 ModifiedTime

bool ModifiedTime(TLuaDateTime &Time)

Return value

Returns true if the value is present or false if otherwise.

Parameters

Time

Contains the time the file was last modified.

19.5 Owner

bool Owner(string &sOwner)

Return value

Returns true if the value is present or false if otherwise.

Parameters

sOwner

Contains name of the owner of the file or directory.

19.6 PermissionBits

bool PermissionBits(int &iPermissionsBits)

Return value

Returns true if the value is present or false if otherwise.

Parameters

iPermissionsBits

Contains an decimal value representing the permission of the file or directory.

19.7 Size

bool Size(int &iBytesHighDWord,int &iBytesLowDWord);

Return value

Returns true if the value is present or false if otherwise.

Parameters

iBytesHighDWord

Contains the high dword portion of the 64 bit integer.

iBytesLowDWord

Contains the low dword portion of the 64 bit integer.

Remarks

Size of the file is reported in bytes as a 64 bit integer. Since Lua lacks a 64 integer data type the information have been split into two 32 bit integers. If the file is less the 2 GB in size, iBytesHighDWord will always be zero.

19.8 SizeMB

bool SizeMB(unsigned int &iSizeMB);

Return value

Returns true if the value is present or false if otherwise.

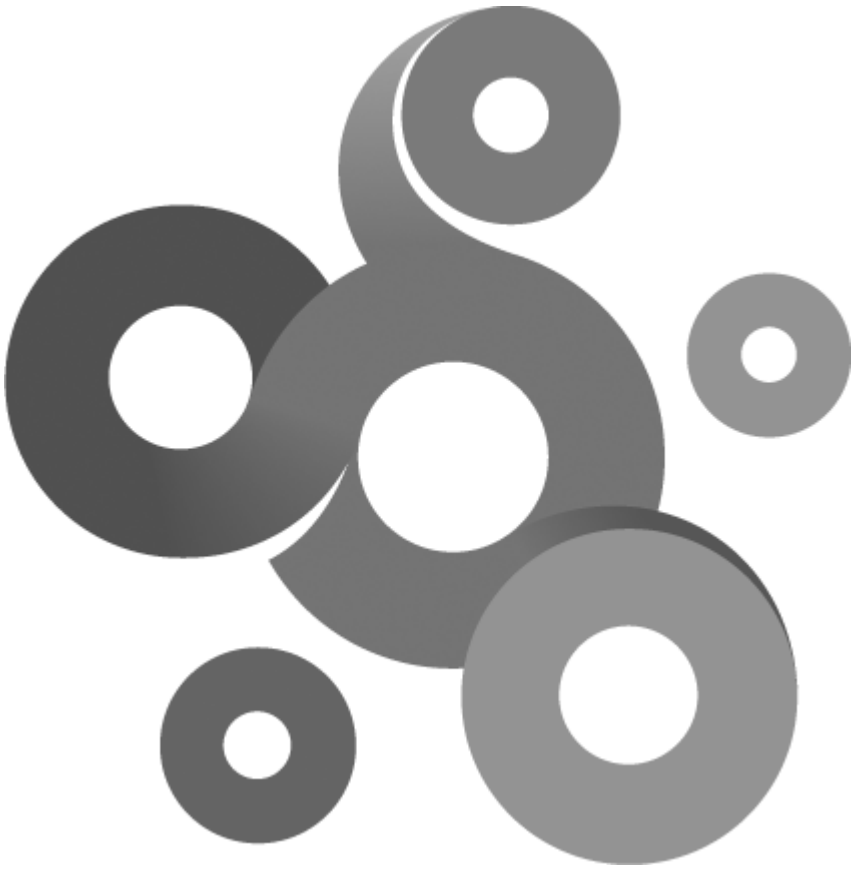
Parameters

iSizeMB

Contains size of the file in megabytes.

Remarks

Provided as an easy to use alternative to the Size() function, returns the size of the file rounded down.



Section XX

20 TLuaSFTPClientDirectoryHandle

This class is used in conjunction with the OpenDir 80, ListDir 79 and CloseDir 79 functions.

20.1 Next

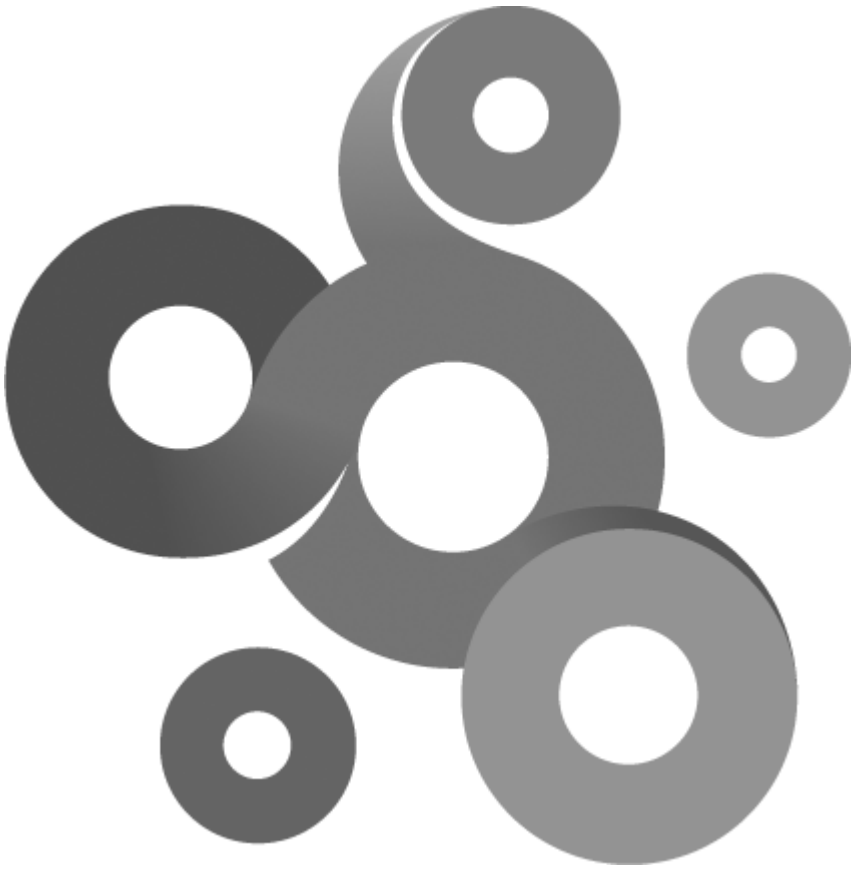
bool Next(TLuaSFTPClientFile &hFile)

Return value

Returns true if the supplied TLuaSfPTClientFile contains data.

Remarks

Loop over the function until it returns false to retrieve all returned information from the ListDir 79 function.



Section XXI

21 TLuaSFTPClientFile

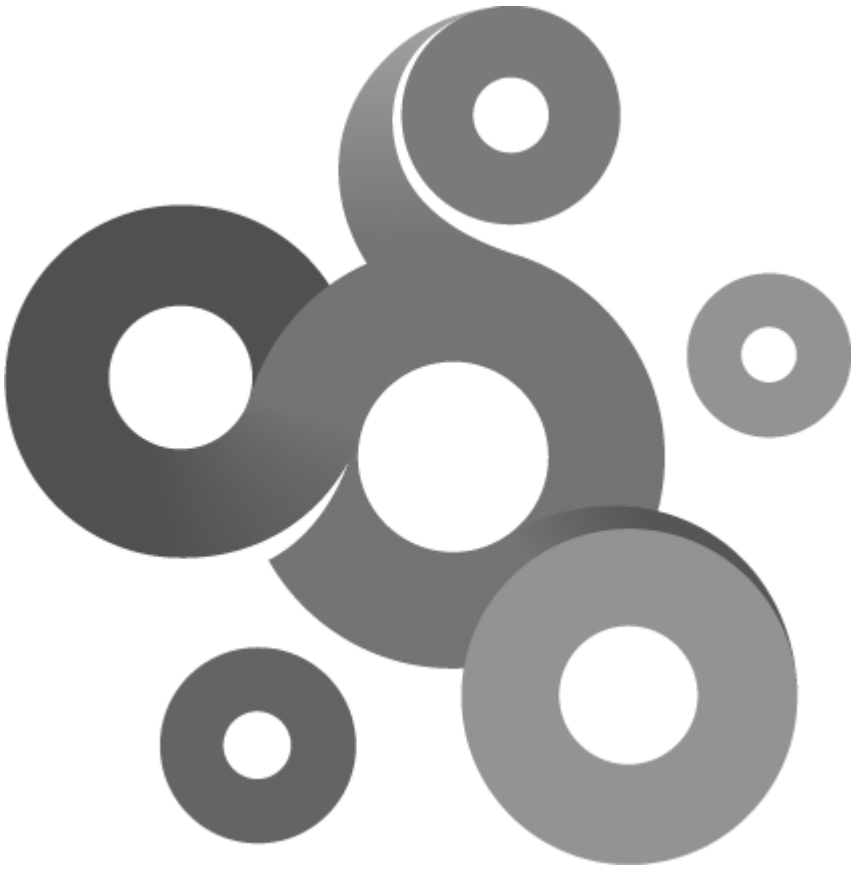
The TLuaSFTPClientFile is a read only class.

Class members

string m_sFilename

string m_sLongFilename

TLuaSFTPClientAttributes m_Attribs



Section XXII

22 TLuaSNMP

The class implements a basic SNMP client that can perform set and get operations.

Example

```
--------------------------------------------------------------------------------
--------------------------------
-- Simple example of SNMP interface
--------------------------------------------------------------------------------
--------------------------------
SNMP = TLuaSNMP();
SNMP:Open("public");

iSyntax =1

sData = SNMP:Get("iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.
ifEntry.ifInOctets.1",iSyntax);

if sData ~= "" then

      print(sData);
      SetExitStatus("Got sample value: "..sData.." bytes received",true);

else

      SetExitStatus("Get failed",false);
end
```

22.1 BeginWalk

BeginWalk(string sOID)

Parameters

sOID

OID representing the start of an OID tree walk.

Example of OID

iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable

Remarks

Before the first call to the Walk function the program must call the BeginWalk function to set the start of the Walk. Walk will retrieve all the child and sibling object identifiers of the start OID set by the BeginWalk functions.

22.2 Close

Close()

Remarks

Closes the SNMP connection.

22.3 Get

string Get(string sOID,int iSyntax)

Return values

A string with the value fetched from the remote SNMP agent.

Parameters

sOID

OID to use in Get operation. When querying an interface the @ operator can be used to specify the interface index.

Example of usage of @ operator:

iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifInOctets@NVIDIA nForce Networking Controller

Example of normal OID

iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifInOctets.1

iSyntax

Specifies the format of the returned data. Can be one of the following constants.

SNMP_NOSYNTAX

SNMP_IPADDRESS

SNMP_INTEGER

SNMP_UNSIGNED32

SNMP_COUNTER32

SNMP_GAUGE32

SNMP_TIMETICKS

SNMP_OPAQUE

SNMP_OCTETSTRING

SNMP_DATA_AS_HEXSTRING

Reading binary values

Some OID's may return binary data instead of for example a string or integer, this can be a problem since the Get function returns a null terminated string. A solution for this problem is to settings the iSyntax variable to SNMP_DATA_AS_HEXSTRING. The function will then return the binary data hexadecimal encoded.

Example of three hexadecimal encoded bytes

49 4E 4D

22.4 Open

bool Open(string sCommunity, int iPort=161)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sCommunity

Name of the community, usually public

iPort

(Optional) Specify the port number if you need to use a port other then the standard port (port 161).

22.5 Set

bool Set(string sOID,string sData,int iSyntax)

Return values

Non zero if the function is successful; otherwise 0.

Parameters

sOID

OID to use in set operation.

sData

Textual data used in set operation.

iSyntax

Specifies the format of the sData parameter. Can be one of the following constants.

SNMP_NOSYNTAX

SNMP_IPADDRESS

SNMP_INTEGER

SNMP_UNSIGNED32

SNMP_COUNTER32

SNMP_GAUGE32

SNMP_TIMETICKS

SNMP_OPAQUE

SNMP_OCTETSTRING

22.6 Walk

TLuaSNMPResult Walk(string sOID)

Return values

A data structure containing the result of the walk operation. When the end is reached the m_sOID member of the TLuaSNMPResult structure will be empty.

Parameters

sOID

OID to walk. This OID should be the OID last returned by the previous call to Walk, or if this is the first call to Walk it should be the same OID as used in BeginWalk.

Remarks

Before the first call to the Walk function the program must call the BeginWalk function to set the start of the Walk. Walk will retrieve all the child and sibling object identifiers of the start OID set by the BeginWalk functions.

Example

```
--------------------------------------------------------------------------------
--------------------------------
-- KNM Lua API example (C) 2007 Kaseya AB
-- Simple example of SNMP interface
--------------------------------------------------------------------------------
--------------------------------
SNMP = TLuaSNMP();
SNMP:Open("public");
```

```
sOID = "iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry";


-- A repeat ... until loop
Result = TLuaSNMPResult();
SNMP:BeginWalk(sOID);
repeat
      Result = SNMP:Walk(sOID);
      sOID = Result.m_sOID;
      print("OID "..sOID);
      print("Data "..Result.m_sData);
      print("Syntax "..Result.m_iSyntax);
until Result.m_sOID == "";
```

22.7 TLuaSNMPResult

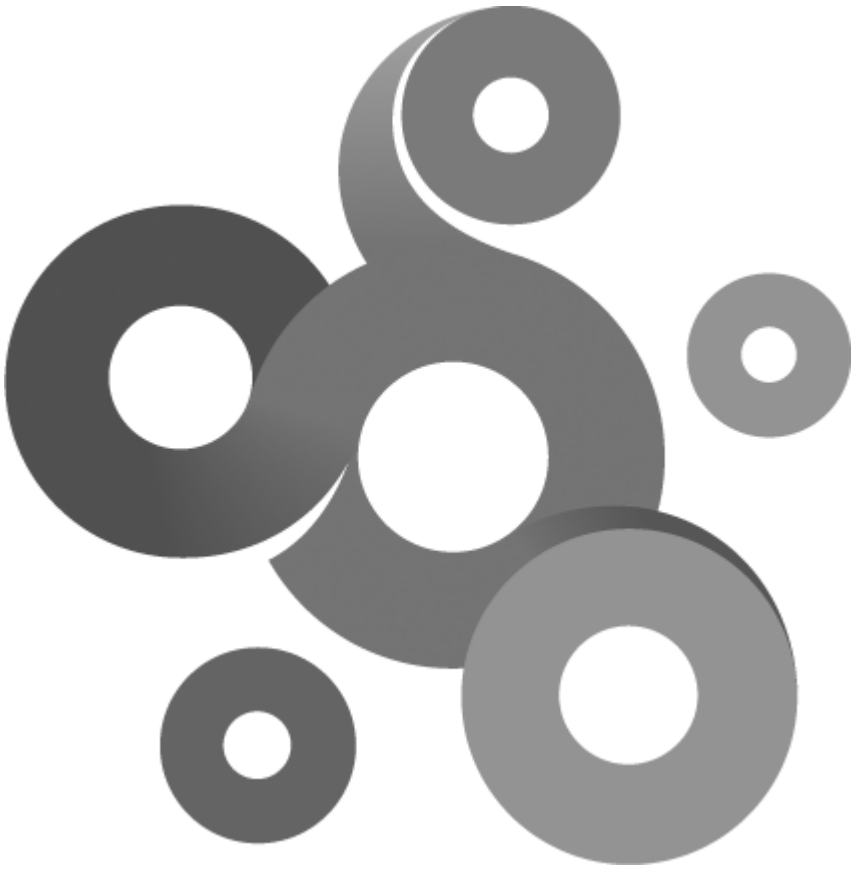
The TLuaSNMPResult is a read only class returned by the Walk function. If modified the an exception will be thrown.

Class members

string m_sOID

string m_sData

int m_iSyntax



Section XXIII

23 TLuaSSH2Client

The class implements a SSH 2.0 client that can execute commands on a remote server.

Example

```
SSHClient = TLuaSSH2Client();
SSHClient:Open(23,"testuser","testpassword");

if SSHClient:ExecuteCommand("shutdown") == true then
      print(SSHClient:GetStdOut());
      SetExitStatus("Exec ok",true);
else
      print(SSHClient:GetStdErr());
      print(SSHClient:GetErrorDescription());
      SetExitStatus("Exec failed",true);
end
```

23.1 ExecuteCommand

bool ExecuteCommand(string sCommand,DWORD dwWait/* =2500*/)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sCommand

String with the command to execute on the remote host.

dwWait

(Optional) Time to wait for execution to finish, default 25 seconds.

23.2 GetErrorDescription

string GetErrorDescription(void)

Return values

Returns a the latest error description generated by the client as a string.

23.3 GetStdErr

string GetStdErr(void)

Return values

Returns the std error output from the remote host.

23.4 GetStdOut

string GetStdOut(void)

Return values

Returns the std output from the remote host.

23.5 Open

bool Open(int iPort,string sUsername,string sPassword)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling function GetErrorDescription. If the failure is a result of the command more information can be retrieved by calling GetStdErr.

Parameters

iPort

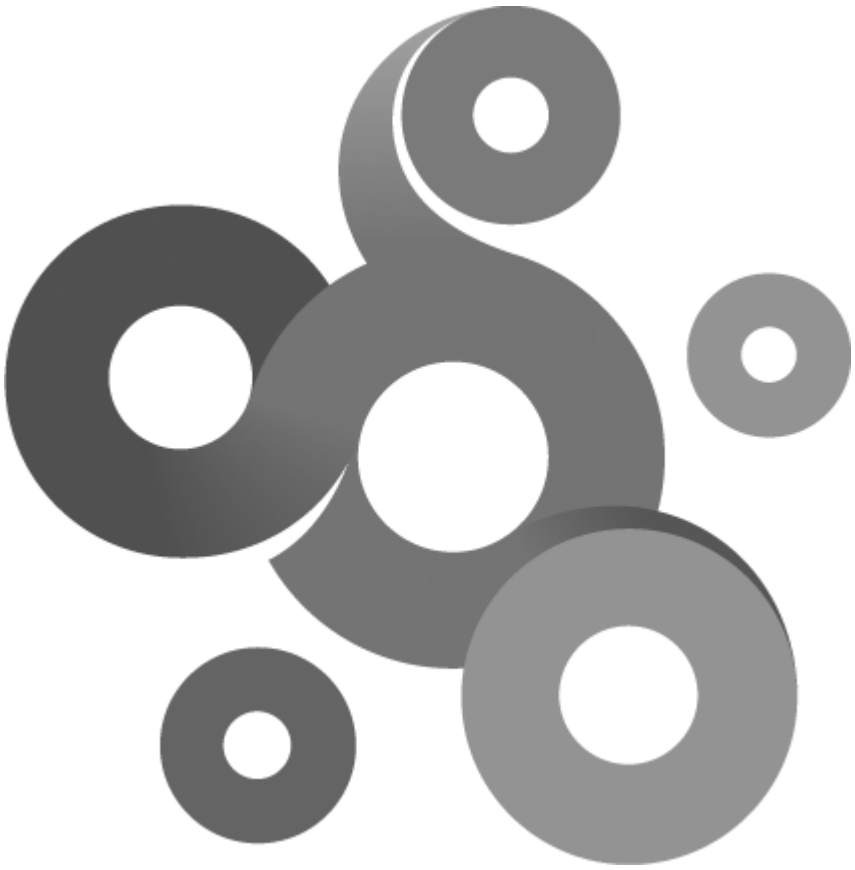
SSH port, default 23.

sUsername

Username

sPassword

Password



Section XXIV

24 TLuaSocket

This class provide basic socket operations. Sockets can be opened in either UDP or TCP mode.

Example

```
-- Construct a new socket object
socket = TLuaSocket()
iPortNumber = 8080
-- Open a TCP socket
iRet = socket:OpenTCP(iPortNumber)
-- If OpenTCP returned a 0 (boolean FALSE) then the open failed
if iRet==0 then
 print("Cannot open port "..iPortNumber.." Error code:"..GetLastError())
else
 -- Read some data (max 1024 bytes) from the socket
 iReadSize = 1024
 data = socket:Read(iReadSize)
 -- Print the content
 if iReadSize > 0 then
  print("Data received from server:\n\n")
  print(data)
 else
  print("No data received from server")
 end
end
socket:Close()
```

24.1 Close

int Close()

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Remarks

Closes the socket previously opened with OpenTCP or OpenUDP.

24.2 OpenTCP

int OpenTCP(int iPort)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

iPort

The port to open.

Remarks

Opens a TCP socket using the specified port number.

24.3 OpenUDP

int OpenUDP(int iPort)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

iPort

A particular port to use with the socket.

Remarks

Opens a UDP socket using the specified port number.

24.4 Read

string Read(int iSize,int iTimeout=1)

Return values

A array of data if the function is successful; otherwise nil if no data could be read, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

iSize

When call to function returns the variable is set to the size of the data read. If no data was read this value will be zero.

iTimeout

The amount of time in seconds to wait for data to arrive to the socket. Default value is one second.

Remarks

The function only blocks execution for the amount of time specified by the timeout value, if no data is received during this period the function will return a nil value.

24.5 Write

int Write(string Data,int iSize)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

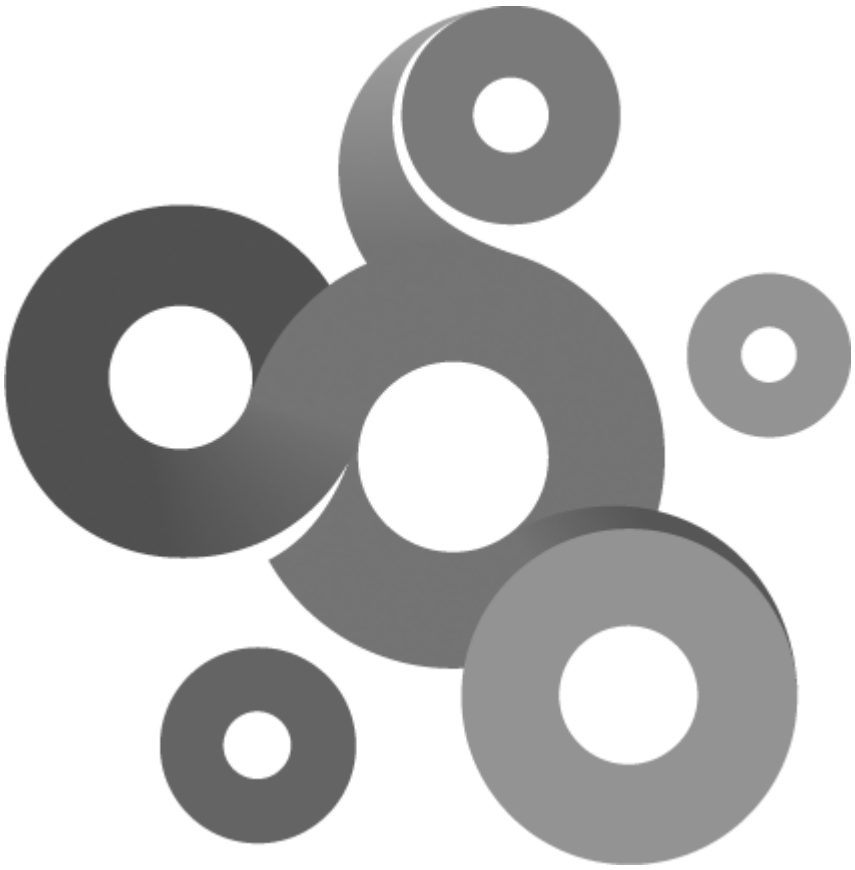
Parameters

sData

An array with data to send.

iSize

Size of the data in the array.



Section XXV

25 TLuaSocketSecure

This class provide basic secure socket operations, commonly referred to as Transport Layer Security (TLS) or its predecessor Secure Sockets Layer (SSL)

Example

```
-- This function is called by KNM when enumerating a field
function OnEnumerate(sFieldToEnum)

	Enum = LuaScriptEnumResult()

	if sFieldToEnum == "Ignore connection problems" then
		Enum:Add("Yes")
		Enum:Add("No")
	end

	return Enum
end

-- This function is called by KNM to retrieve a script configuration
function OnConfigure()

	Config = LuaScriptConfigurator()
	Config:SetAuthor("Robert Aronsson, Kaseya AB")
	Config:SetDescription("The script check if a certificate is about to
expire within the configured number of days.")
	Config:SetMinBuildVersion(5280)
	Config:SetScriptVersion(1,0)

	Config:AddArgument("Port number","Port number to connect on",
LuaScriptConfigurator.CHECK_NOT_EMPTY)
	Config:AddArgument("Number of days","Check if certificate expres
within this period",LuaScriptConfigurator.CHECK_NOT_EMPTY)
	Config:AddArgument("Ignore connection problems","Do you want the
script to report connection problems as well ?",LuaScriptConfigurator.
ENUM_AVAIL + LuaScriptConfigurator.CHECK_NOT_EMPTY)

	Config:SetEntryPoint("main")

	return Config
end

-- This is the entry point
function main()

	local iPort = GetArgument(0)
	local iNumDays = GetArgument(1)
	local bReportConnectionProblem = false;
	if GetArgument(2) == "Yes" then
		bReportConnectionProblem = true
	end

	-- Timeperiod that the certificate should be valid within
	local iOffsetTime = (60 * 60 * 24) * iNumDays
```

```
	-- Default values for test eval
	local bTestOk = true;
	local sText = "Certificate ok";

	-- Open socket
	Socket = TLuaSocketSecure()
	if Socket:Open(iPort) ~= 0 then

		CurrentTime = TLuaDateTime();

		-- The time was retrived during the connect
		Time = Socket:GetCertificateExpiryDate();

		print("Certificate expires ("..Time:GetDate() .." " .. Time:
GetTime()..")");

		-- Check time
		iExpiryTime = Time:Get() - iOffsetTime;
		if Time:Get() < CurrentTime:Get() then
			bTestOk = false;
			sText = "Certificate have already expired ("..Time:
GetDate() .." " .. Time:GetTime()..")";
		else
			if iExpiryTime < CurrentTime:Get() then
				bTestOk = false;
				sText = "Certificate is about to expire in less
than "..iNumDays.." days"
			end
		end
	else
		-- Failed to open the socket, server down ?
		if bReportConnectionProblem == true then
			bTestOk = false;
		end
		sText = "Cannot connect to host.";
	end

	-- Report status and exit
	SetExitStatus(sText,bTestOk);
end
```

25.1 Open

int Open(int iPort)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

iPort

The port to open

25.2 Close

int Close()

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

25.3 Read

string Read(int iSize)

Return values

A array of data if the function is successful; otherwise nil if no data could be read, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

iSize

When call to function returns the variable is set to the size of the data read. If no data was read this value will be zero.

25.4 Write

int Write(string Data,int iSize)

Return values

Non zero if the function is successful; otherwise 0, and a specific error code can be retrieved by calling global function GetLastError.

Parameters

sData

An array with data to send.

iSize

Size of the data in the array.

25.5 GetCertificateExpiryDate

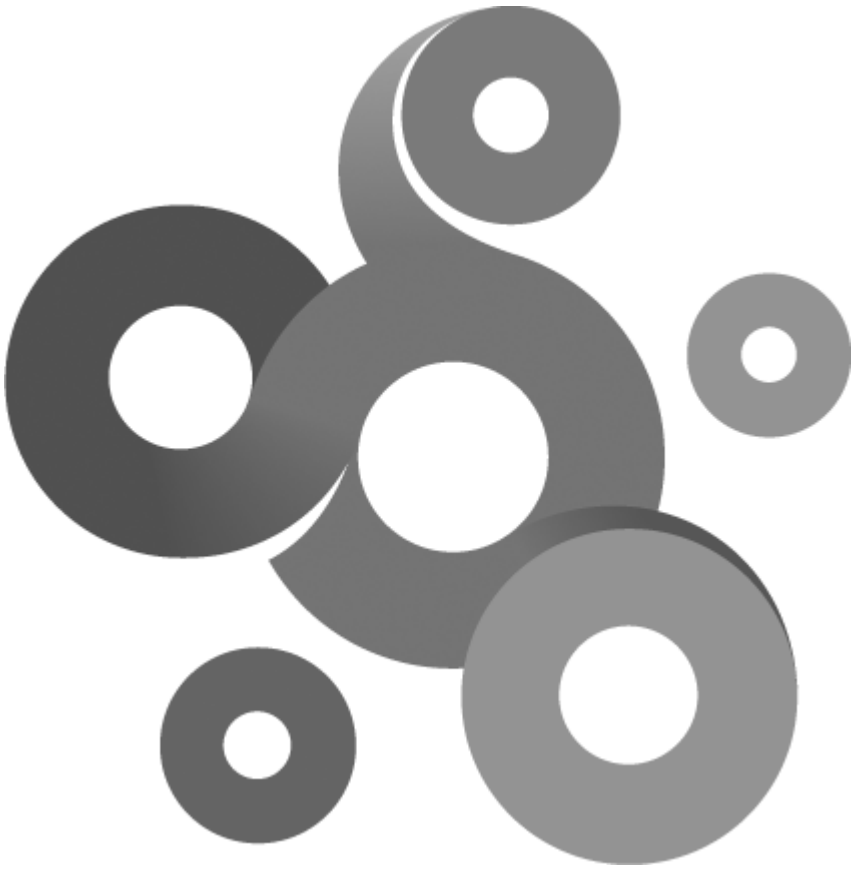
TLuaDateTime GetCertificateExpiryDate()

Return values

A TLuaDateTime structure containing the date when the certificate of the remote host expires. If the Connect() call failed, the structure will contain a zero date.

Remarks

This function can be used to determine if a certificate is about to expire or have expired already.



Section XXVI

26 TLuaStorage

The class provides to save textual data between script sessions. It can be useful when you want to base the current script iteration on a previous result or communicate between two unrelated scripts.

26.1 CreateItem

bool CreateItem(string sName,string sKey,string sData=NULL,int iSize=0)

Return values

Non zero if the function is successful; otherwise 0.

Parameters

sName

Unique name of the item, if the name is already created this function fail.

sKey

Key name of the item, must be unique if it already exist this function will fail.

sData

Optional data that will be associated with the item

iSize

Size of the data, only needed if data is supplied with function.

Remarks

The function creates an item and an sub item called a "key", the user can associate data with this key. The data can later be acquired by called the function FindItem.

26.2 UpdateItem

boo UpdateItem(string sName,string Key,string Data=NULL,int iSize=0)

Return values

Non zero if the function is successful; otherwise 0.

Parameters

sName

Unique name of the item, an item with this name must already exist.

sKey

Key name of the item, a key with name must already exist.

sData

Optional data that will be associated with the item, the data will replace the current data stored in the item (if any).

iSize

Size of the data, only needed if data is supplied with function.

Remarks

The function updates an already created item, if the item/key combination does not exist this function will fail.

26.3 DeleteItem

void DeleteItem(string sName,string sKey);

Parameters

sName

Name of item.

sKey

Name of key to delete.

Remarks

The function deletes an item/key combination, data associated with the key will also be deleted.

26.4 FindItem

TLuaStorageItem FindItem(string sName,string sKey);

Return values

Non zero if the function is successful; otherwise 0.

Parameters

sName

Unique name of the item, an item with this name must already exist.

sKey

Key name of the item, a key with name must already exist.

Remarks

The function retrieves an stored item, the returned class contains the item/key names as well as the data associated with the item.

26.5 TLuaStorageItem

The TLuaStorageItem is a read only class. If modified an exception will be thrown.

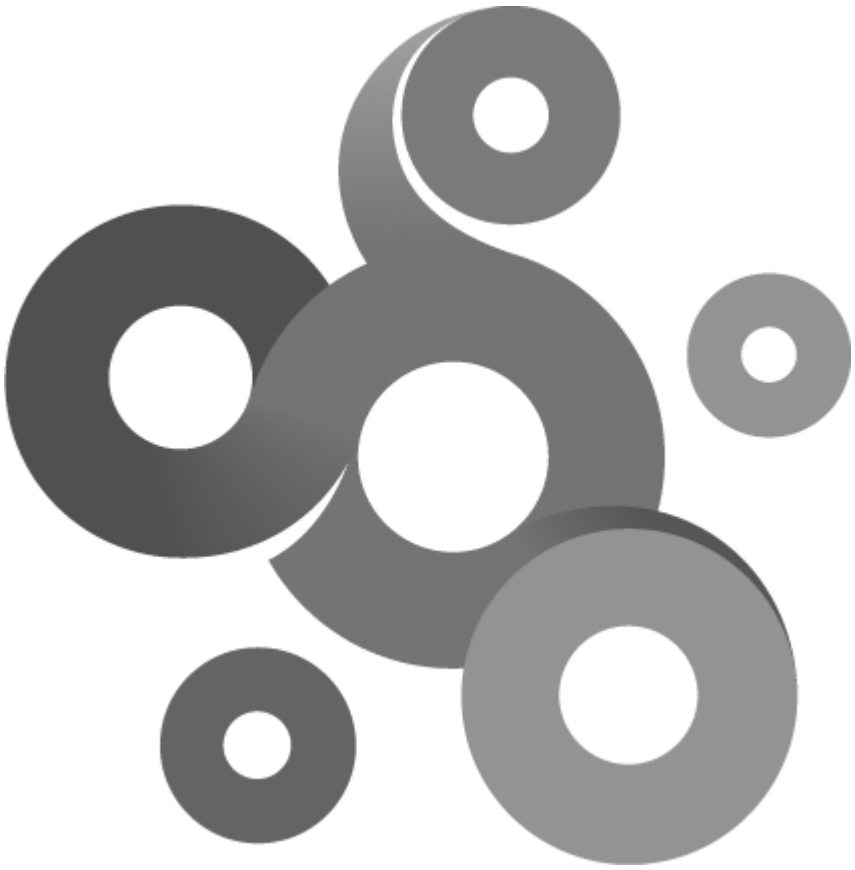
Class members

string m_Key

string m_Name

string m_pData

int m_iSize



Section XXVII

27 TLuaTimer

The class provides a timer with milisecond precision.

Example

```
-------------------------------------------------------------------------------
-------------------------------
-- Demonstrates the Lua timer interface
-------------------------------------------------------------------------------
-------------------------------
Timer = TLuaTImer();
Timer:Start()
print("Timer started");
Wait(1000);
print("Operation took "..Timer:Stop().." ms";
```

27.1 Start

Start()

Remarks

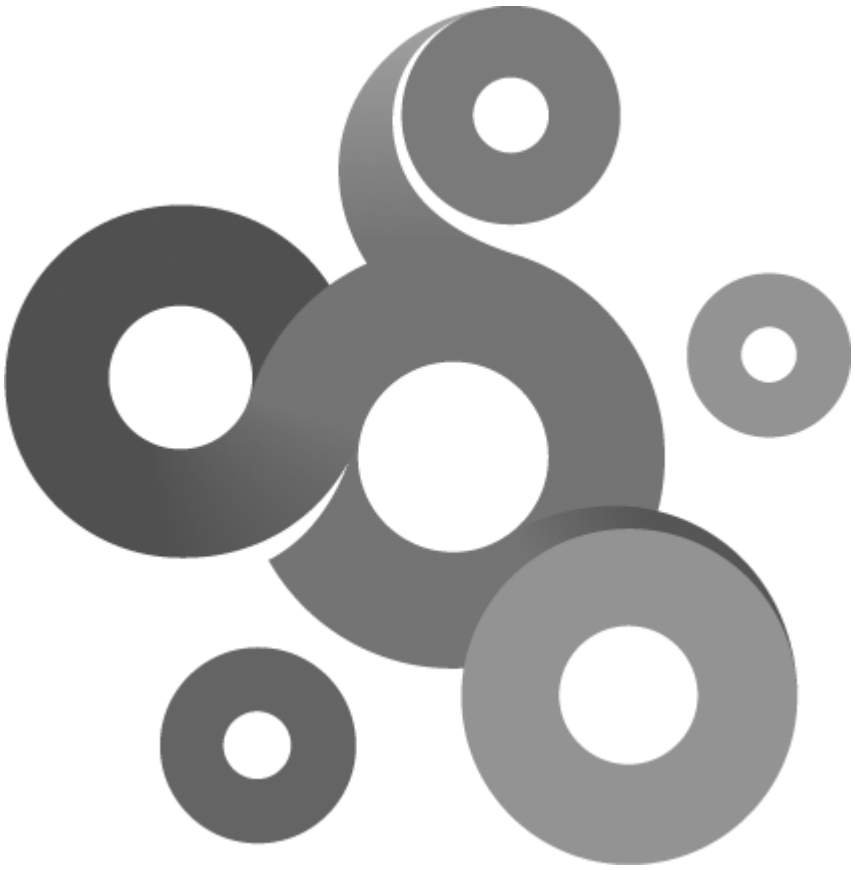
The function starts a time and a subsequent call to the Stop() function will return the time between the calls to Start and Stop. After Stop is called, call Start again to reset the timer and start a new period.

27.2 Stop

int Stop()

Return values

Returns the number of miliseconds since the call to the Start() function.



Section XXVIII

28 TLuaWinperf

The class provides functions to query numeric values in the Windows performance register. Its provided as an easy to use alternative to the more advanced TLuaWMIQuery class. The class is executing in the security context of the process or thread that launched the script. In the IDE the security context is inherited from the desktop. When executed by the Lua script monitor the security context can be set by selecting an default account in the monitor property page.

Example

```
--------------------------------------------------------------------------------
-- Prints the number of private bytes the notepad.exe application have
allocated
--------------------------------------------------------------------------------

Perf = TLuaWinperf()
if Perf:Query("Process","Private Bytes","notepad") then

      Value = Perf:GetResult();
      print(Value);
else
      print(Perf:GetErrorDescription())
end
```

28.1 GetErrorDescription

string GetErrorDescription()

Return values

Returns a string describing the latest error encountered when calling any function in the class.

28.2 GetResult

double GetResult()

Return values

Returns a numeric counter value, if the previous call to Query() failed, this function will return zero.

28.3 Query

bool Query(string sObjectName,string sCounterName,string sInstanceName=NULL);

Return values

True if the query was successfully executed, false if an error occurred.

Parameters

sObjectName

A string with the name of the object containing the counter to query.

sCounterName

A string with the name of the counter to query

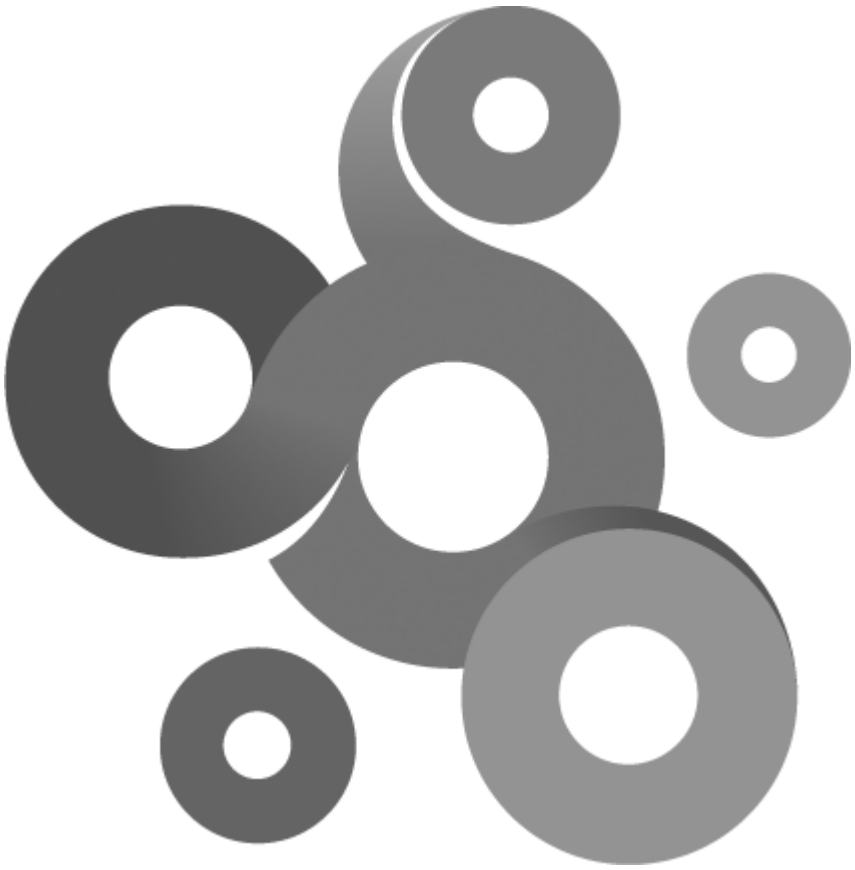
sInstanceName

(Optional) string with the name of the counter instance.

Remarks

Object, counter and instance names can be obtained either in the KNM Winperf monitor by clicking on the

enumeration button or by using the Windows perfmon.exe application. To retrieve the value call GetResult() after this function completed.



Section XXIX

29 TLuaWMIQuery

The class provides functions to query WMI properties. The class is executing in the security context of the process or thread that launched the script. In the IDE the security context is inherited from the desktop. When executed by the Lua script monitor the security context can be set by selecting an default account in the monitor property page. The account must be enabled for delegation.

Example

```
-------------------------------------------------------------------------------
-- Demonstrates the Lua WMI interface
-------------------------------------------------------------------------------
Query = TLuaWMIQuery();
Query:Execute("select Deviceid,Size,Freespace from win32_logicaldisk");
print(Query:GetErrorDescription());

while (Query:NextInstance()) do
      sDeviceID = "";
      bOk,sDeviceID = Query:GetProperty("Deviceid",sDeviceID);
      print(sDeviceID);
end
```

29.1 Execute

bool Execute(string sWQL)

Return values

True if the query was successfully executed, false if an error occurred.

Parameters

sWQL

A string containing a WQL query.

Remarks

Executes an WQL (WMI Query Language) query. Calls to Next() and GetProperty() can be used to retrieve the result.

29.2 GetErrorDescription

string GetErrorDescription()

Return values

Returns a string describing the latest error encountered when calling any function in the class. Useful when debugging WMI queries.

29.3 GetProperty

bool,string GetProperty(string sPropertyName,string sReturnValue);

Return values

Returns true and a value in a string if successful, false and an empty string if the function failed. More detailed information about the error can be retrieved by called GetError().

Parameters

sPropertyName

Name of the property to retrieve.

sReturnValue

Defined string receiving the return value. The return value is always a string, even if the property type is for example an integer or a real number.

Remarks

Retrieves a property value in the current result. To retrieve the next value of the same property call the NextInstance() function. If NextInstance() returns false, there are no more values.

29.4 NextInstance

bool NextInstance()

Return values

True if a new result was fetched, false if no more results of the query exists.

Remarks

The function retrieves a new result generated by a previous call to the Execute function. This function must be called before the first call to the GetProperty function.

29.5 SetNamespace

SetNamespace(string sNamespace)

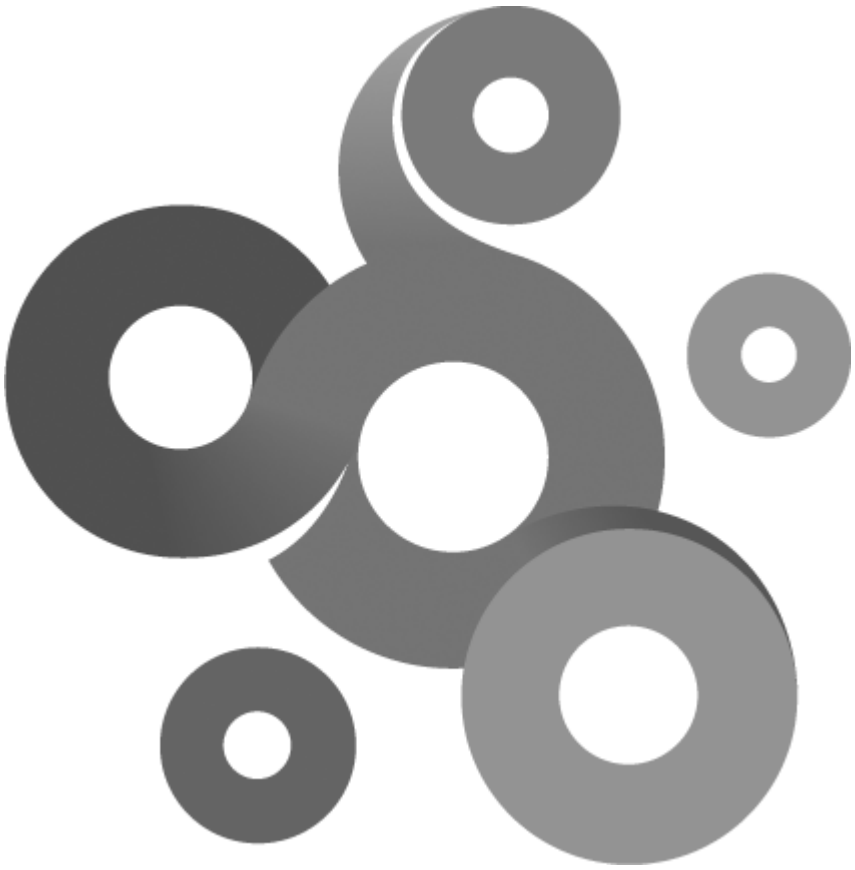
Parameters

sNamespace

String with WMI namespace to use in all future calls.

Remarks

The default namespace used by the TLuaWMIQuery class is root\cvmi2.



Section XXX

30 TLuaXMLNode

The class represents a XML element and can contain one or more child elements.

30.1 FindAttribute

string FindAttribute(string sName)

Return values

The function returns a string with the value of the attribute. If the attribute cannot be found, the returned string is empty.

Parameters

sName

The name of the attribute

30.2 FindChildNode

TLuaXMLNode FindChildNode(string sElementName, int iOffset)

Return values

The function returns a valid TLuaXMLNode object if the element was found.

Parameters

sElementName

The name of the element that is a child of this node

iOffset

A zero based index to retrieve child elements with the same name in the node.

Remarks

The function can be used to iterate over a number of child elements with the same name. Increment the offset parameter to retrieve the next element.

30.3 GetData

string GetData()

Return values

The function returns the data in the element.

30.4 GetTag

string GetTag()

Return values

The function returns the tag name of the element.

30.5 GetParentNode

TLuaXMLNode GetParentNode()

Return values

The function returns the parent of the current XML document element.

30.6 IsValid

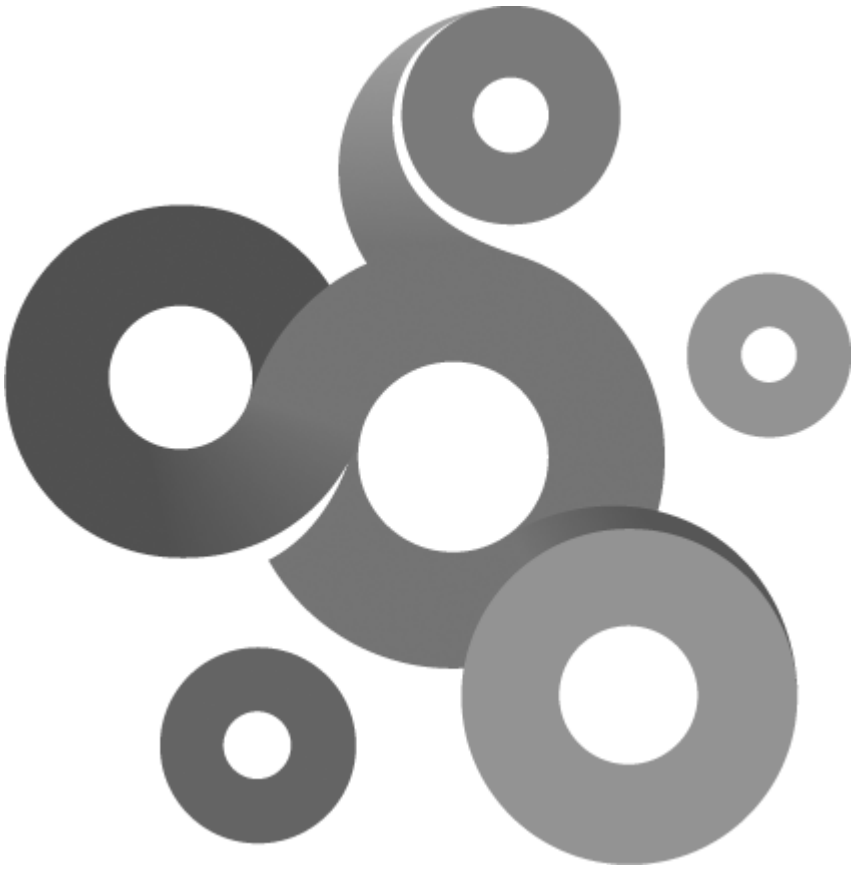
bool IsValid()

Return values

The function returns true if the node is valid and false if the node is invalid.

Remarks

All search functions returns a TLuaXMLNode object, the IsValid() function is used to determine if the search was successful.



Section XXXI

31 TLuaXMLReader

The class provides basic functionality to parse and traverse XML documents.

31.1 FindChildNode

TLuaXMLNode FindChildNode(string sElementName, TLuaXMLNode ParentNode)

Return values

The function returns a valid TLuaXMLNode object if the element was found.

Parameters

sElementName

The name of the element that is a child of ParentNode

ParentNode

The parent node to be searched

Remarks

Note that the function returns the first element with the specified name.

31.2 FindNode

TLuaXMLNode FindNode(string sElementName, TLuaXMLNode RootNode)

Return values

The function returns a valid TLuaXMLNode object if the element was found.

Parameters

sElementName

The name of the element that is a child of ParentNode

RootNode

The parent node to be the starting point of search.

Remarks

The function searches the XML document recursively with "RootNode" as starting point for the search.

31.3 FromXML

bool FromXML(string XML)

Return values

True if the operation was successfully executed, false if an error occurred.

Parameters

sXML

A XML document to parse

Remarks

Note that the parser do not validate the document a schema.

31.4 GetRootNode

TLuaXMLNode GetRootNode()

Return values

The function returns the XML document root element.

Index

- A -

- B -

- C -

- D -

- E -

- F -

- G -

- H -

- I -

- L -

- M -

- N -

- O -

- P -

- Q -

- R -

- S -

- T -

- U -

- W -