Network Monitor API

ユーザーガイド

バージョン R91

日本語

June 10, 2015

Agreement

The purchase and use of all Software and Services is subject to the Agreement as defined in Kaseya's "Click-Accept" EULATOS as updated from time to time by Kaseya at http://www.kaseya.com/legal.aspx. If Customer does not agree with the Agreement, please do not install, use or purchase any Software and Services from Kaseya as continued use of the Software or Services indicates Customer's acceptance of the Agreement."

©2015 Kaseya. All rights reserved. | www.kaseya.com

目次

Network Monitor Lua API

このドキュメントは **Network Monitor** の Lua API について説明します。**Network Monitor** では Lua 5.0 を使用します。

Lua

Lua は、アプリケーションを拡張するために設計された、強力で非常に高速なプログラミング言語です。また、Lua は汎用独立型言語としてもよく使用されています。Lua は無償のソフトウェアです。Lua は、簡単な手続き型構文を、連想配列と拡張可能なセマンティクスに基づく強力なデータ記述構造体と組み合わせるものです。Lua はバイトコードから動的に入力と変換が行われます。また、ガベージコレクションによる自動メモリ管理機能があるため、構成、スクリプト作成、ラピッドプロトタイピングに最適です。

注：このドキュメントでは、Lua 言語自体については説明しません。Lua 言語の詳細については、**http://www.lua.org** 『**http://www.lua.org.**』を参照してください。

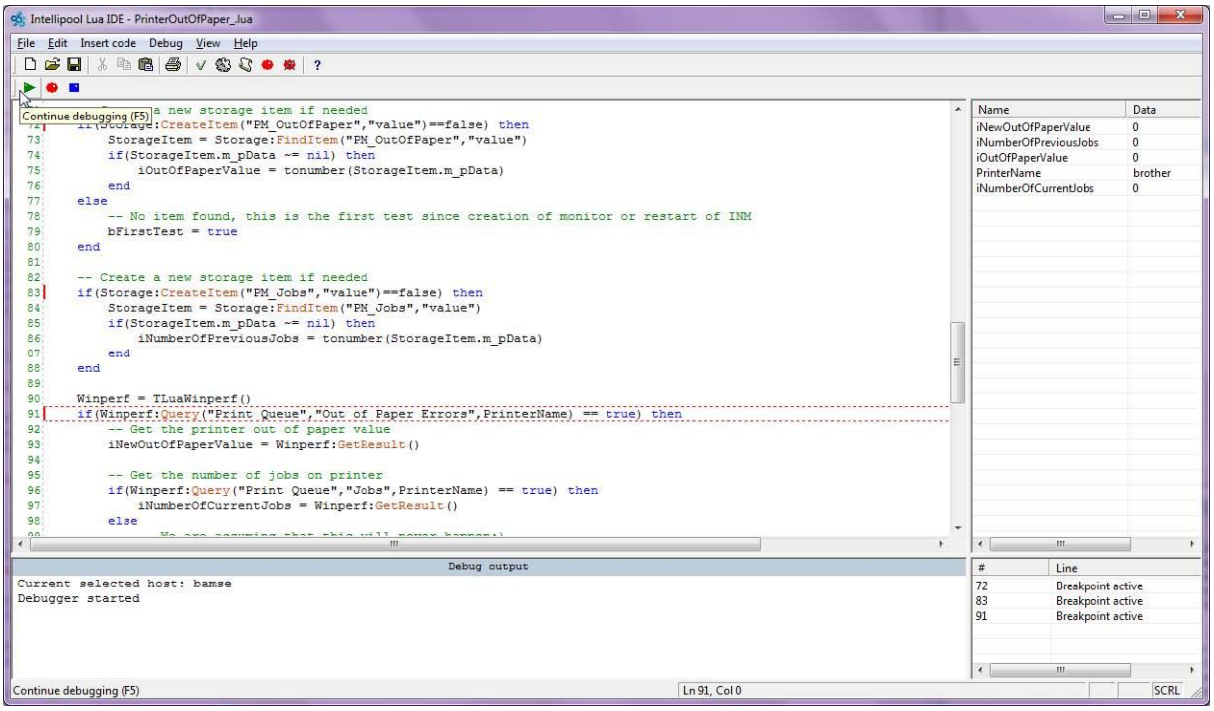
Network Monitor および Lua

顧客は、Lua スクリプト言語を使用してカスタムモニターを作成し、現在のモニタリングソリューションではサポートされていないシステムや機器をテストできます。

Kaseya が提供する開発環境で、新しいモニター、アクション、およびイベントの作成とテストを行ってから、Kaseya Network Monitor にエクスポートして使用できます。

SFTP クライアント、HTTP クライアント、ファイル管理などの作成済みのクラスをもつ総合ライブラリが利用可能です。

開発環境には、デバッガ、キーワードハイライト、統合型ヘルプ、および最新の開発ツールで利用可能なその他の機能が含まれています。開発環境は、弊社のホームページ **http://www.kaseya.com** 『**http://www.kaseya.com**』からダウンロードできます。



Lua IDE V3

チャプター 1

プログラミングモデル

Kaseya Network Monitor で使用するカスタム Lua スクリプトを作成するときには、スクリプトが Kaseya Network Monitor で正常に動作するために満たす必要がある要件がいくつかあります。

この章で

高度なスクリプト

高度なスクリプトモデルにより、スクリプト作成者は新しい強力なツールを使用して、スクリプトに引数として指定するパラメータをコントロールできます。これにより、ネイティブのモニタータイプと同様な外観をもつ Lua スクリプトを作成できます。

予約されている関数名

Network Monitor が情報のクエリーに使用するために予約済みの関数名が 2 つあります。これらの関数名は他の目的に使用することはできません。

OnConfigure

この関数は **Network Monitor** により呼び出され、スクリプトから LuaScriptConfigurator クラスインスタンスを事前設定します。この情報は、スクリプトのユーザーインターフェースを作成するために使用されます。関数が返したこのインスタンス名を **Network Monitor** が Lua スタック内で見つけることができるように、このインスタンス名は"Config"である必要があります*（大文字で始まることに注意）*。

OnEnumerate

ユーザーインターフェースの各フィールドは列挙可能であり、**Network Monitor** は OnEnumerate 関数を呼び出して、スクリプトからデータ構造 LuaScriptEnumResult にユーザーが選択可能な値を事前設定します。OnEnumerate には、sFieldToEnum というパラメータがあり、スクリプトはこのパラメータを使用してどのフィールド/引数に列挙結果を渡すかを決定します。返されるインスタンスには、"Enum"という名前を付ける必要があります（大文字で始めることに注意）。

エントリポイント

この高度なスクリプトモデルには、エントリポイント関数の名前を設定する OnConfigure 関数が必要です。この関数は **Network Monitor** によって呼び出され、スクリプトの実行を開始します。エントリポイントのデフォルトの名前は"main"ですが、予約済みの関数名以外の任意の名前に設定できます。

例

```
--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 specificbuild version is
required.
   Config:SetMinBuildVersion(0)

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

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

   --Add another parameter, a select box with 3 values.
   Config:AddArgument("Argument 2","This is the description of thesecond
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 asset
   return Config
end

--This is the entry point
```

```
function main()

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

  SetExitStatus("OK",true)

end
```

単純なスクリプト

この単純なスクリプトモデルは、最初のリリース以来 **Network Monitor** で使用されていますが、現在は非推奨です。これは古いスクリプトとの互換性を保つために維持されています。

アセットコンテキスト

関数はアセットコンテキストからの相対パスで指定します。

リソースにアクセスするすべての呼び出しは、親アセットからの相対パスで指定します。たとえば、スクリプトがファイルを開く場合、Open 関数に指定するパスはアセットからの相対パスになります。

例

ホストを Windows コンピュータである"domainserver"のアドレスに設定します。

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

この関数を呼び出すと、スクリプトは、コンピュータ"domainserver"の C: ハードディスク上にある test.txt ファイルを開きます。

このため、TLuaFTPClient、TLuaHTTPClient、TLUASocket などのすべての通信関係のクラスはポート番号の引数のみを受け取ります。IP アドレスはフレームワークにより現在のアセットにハードコード化されています。

結果

スクリプトが存在する場合、スクリプトは **Network Monitor** にテストが成功したかどうかを通知する必要があります。この目的で、グローバル関数 SetExitStatus が用意されています。SetExitStatus は必須であり、スクリプトが終了する前に呼び出す必要があります。

チャプター 2

グローバル関数

グローバル関数はアセットと関連付けられていない関数です。**Network Monitor** の Lua API には、いくつかのグローバル関数があります。その一部は、スクリプトの終了時に呼び出す必要があります。

この章で

ConvertFromUTF16

string ConvertFromUTF16(local UTF16data,int iSize)

戻り値

UTF16 文字列から変換された 8 ビット文字列。

パラメータ

- UTF16data - TLuaFile::ReadData によって読み出される UTF16（2 バイト）文字列。
- iSize - 文字列の長さ。

注釈

この関数が受け取るデータは、TLuaFile::ReadData 関数が作成したデータのみです。

FormatErrorString

string FormatErrorString(int iError)

戻り値

エラーコードである iError を説明する文字列。

パラメータ

- iError - 以前に GetLastError 関数を呼び出して取得した Windows のエラーコード。

注釈

この関数を使用して、エラーコードの代わりに意味があるテキストをユーザーに提供できます。

GetArgument

string GetArgument(int iNumber)

戻り値

呼び出し側のアプリケーションから渡される引数。

パラメータ

- iNumber - 取得する引数のゼロから始まるインデックス。引数の最大数は、GetArgumentCount を呼び出すことで分かります。

注釈

呼び出し側のアプリケーションは、動作をカスタマイズするために Lua スクリプトに複数の引数を渡すことができます。この関数および関連する GetArgumentCount を使用して、引数を取り出すことができます。

GetArgumentCount

int GetArgumentCount()

戻り値

呼び出し側アプリケーションからプログラムに渡される引数の数

注釈

呼び出し側のアプリケーションは、動作をカスタマイズするために Lua スクリプトに複数の引数を渡すことができます。この関数を使用すると、取り出す引数の数を知ることができます。

GetLastError

int GetLastError()

戻り値

ライブラリ関数の呼び出しにより生成された最後のエラーコード。このエラーコードは、Windows の標準エラーコードです。

注釈

function.sds を呼び出す前に、SetLastError を使用して現在の Windows エラーコードをクリアできます。

GetDeviceAddress

string GetDeviceAddress()

戻り値

デバイスのアドレスフィールドに入力されたアドレス。

注釈

データを TLuaStorage に保存するときに、この文字列を一意の識別子として使用できます。

IsIDE

bool IsIDE()

戻り値

スクリプトが IDE によって実行された場合はブール値の true、スクリプトが **Network Monitor** によって実行された場合は false です。

注釈

この関数は、スクリプトが **Network Monitor** または IDE により実行される場合に使用できます。

MessageBox

MessageBox(string sText)

パラメータ

- sText - メッセージボックスに表示するテキスト。

注釈

この関数は、文字列を表示するために OS の標準 OS メッセージボックスを呼び出します。この関数は IDE でのみ利用可能です。メッセージボックスが表示されてから閉じられるまで、スクリプトの実行が停止することに注意してください。

print

print(string sText)

パラメータ

- sText - 出力ウィンドウに出力するテキスト

注釈

この関数は、デバッグ目的で出力ウィンドウにテキストを出力するために使用できます。このスクリプトが **Network Monitor** によって実行された場合、この関数で出力されたテキストは何の役にも立ちません。

SetExitStatus

SetExitStatus(string sString,bool bSuccess)

パラメータ

- sString - スクリプトの実行結果を説明する文字列。
- bSuccess - 非ゼロ（ブール値の true）の場合、スクリプトはフレームワークにより正常に実行されたと考えられます。この値がゼロ（ブール値の false）に設定された場合は、エラーのステータスを説明する文字列を指定して SetErrorString 関数も呼び出す必要があります。

注釈

この関数はスクリプトが終了するときに呼び出される必要があり、スクリプトが正常に終了したかどうかを **Network Monitor** に通知します。スクリプトがエージェントのコンテキストで実行される場合、関数に指定したテキストは **Network Monitor** によって使用され、最終ステータスのテキストをインターフェースに設定します。

SetLastError

SetLastError(int iErrorCode)

パラメータ

- iErrorCode - Windows 固有のエラーコードに対応する整数。

注釈

この関数は最終エラーコードを設定し、GetLastError が後でこれを取得できます。

StoreStatisticalData

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

戻り値

データが正常に統計データベースに格納された場合は true、パラメータエラーがあった場合は false。

パラメータ

- iRecordSetIndex - データ情報を格納する統計チャンネルのゼロから始まるインデックス。有効な定数に関しては注釈を参照してください。
- fData - スクリプトがサンプリングした浮動小数点データ。
- fThreshold - サンプルデータに対するオプションの閾値で、この値はすべての呼び出しで一定である必要があります。
- Unit - データ単位を説明するオプションの文字列で、この値はすべての呼び出しで一定である必要があります。文字列の最大長は 16 文字で、この長さを超えると呼び出しは失敗します。

注釈

この関数を使用すると、スクリプトは統計データを格納できます。現在、この目的で 8 チャンネルを使用できます。iRecordSetIndex パラメータには、次の定数のいずれかを使用できます。

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

StoreStatisticalData

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

戻り値

データが正常に統計データベースに格納された場合は true、パラメータエラーがあった場合は false。

パラメータ

- iRecordSetIndex - データ情報を格納する統計チャンネルのゼロから始まるインデックス。有効な定数に関しては注釈を参照してください。

- fData - スクリプトがサンプリングした浮動小数点データ。
- fThreshold - サンプルデータに対するオプションの閾値で、この値はすべての呼び出しで一定である必要があります。
- iVirtualType - 格納するデータ型。
- iVirtualUnit - 格納する型について選択した単位。型と単位の有効な組み合わせについては注釈を参照してください。
- Unit - データ単位を説明するオプションの文字列で、この値はすべての呼び出しで一定である必要があります。文字列の最大長は 16 文字で、この長さを超えると呼び出しは失敗します。

注釈

この関数は高度なスクリプトでのみ利用可能です。この関数と、同じ名前をもつ古い関数との違いは、データと共に型の情報を格納できることです。

iVirtualType と iVirtualUnit は、次の組み合わせで使用できます。

```
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
```

レコードを設定するインデックスパラメータには、次の定数のいずれかを使用できます。

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

Wait

Wait(int iMs)

パラメータ

- iMs - スクリプトが実行を待機する時間（単位: ms）。

注釈

この関数は OS の"Sleep"関数を呼び出し、スクリプトが実行されるスレッドの実行を中断します。

チャプター 3

LuaScriptEnumResult

このクラスは、OnEnumeration のコールバック関数に列挙結果を入力するインターフェースを提供します。

この章で

サンプルスクリプト:OnEnumerate

```
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
```

追加

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

パラメータ

- sDisplayValue - 選択オプションとして表示する値。
- sUsageValue - （オプション）表示値の代わりに使用する値。

注釈

オプションの sUsageValue は、非常に複雑で長い値があり、オプションの簡単な表示方法が必要な場合に使用できます。この関数を使用すると、sDisplayValue はユーザーに提示する値になり、sUsageValue は **Network Monitor** が使用する値になります。

チャプター 4

LuaScriptConfigurator

このクラスは構成情報を作成するインターフェースを提供し、**Network Monitor** はこの構成情報を使用してスクリプトのユーザーインターフェースを提示します。

この章で

サンプルスクリプト:OnConfigure

```
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 specificbuild 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 thefirst
argument",LuaScriptConfigurator.CHECK_NOT_EMPTY)

  --Add another parameter, a select box with 3 values.
  Config:AddArgument("Argument 2","This is the description of thesecond
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 asset
  return Config
end
```

AddArgument

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

戻り値

後続の呼び出しでこの引数を参照するときに使用できるハンドル。

パラメータ

- sName - 引数フィールドの名前。
- sDescription - 検証のコントロールを行うフラグである iFlags フィールドの説明。フラグについては注釈を参照してください。

注釈

有効なフラグを示します。一部のフラグは組み合わせが可能です。

CHECK_NOTHING	テキストのない値も含めて、あらゆる型のデフォルト値が受け付けられます。
CHECK_NOT_EMPTY	引数が空でないかチェックします。CHECK_NOTHING と組み合わせることはできません。
CHECK_RANGE_LOW	CHECK_NUMERIC と共に使用する必要があります。数値が範囲内にあるかどうかを検証します（下限）。
CHECK_RANGE_HIGH	CHECK_NUMERIC と共に使用する必要があります。数値が範囲内にあるかどうかを検証します（上限）。
CHECK_NUMERIC	値が数値であるかどうかを検証します（実数または整数）。
ENUM_AVAIL	このフィールドで利用可能な事前定義値をもつ列挙型コールバックがあることを示します。

SetCharacterLimits

SetCharacterLimits(int iArgIndex,int iMaxCharacters,int iMaxVisibleCharacters)

パラメータ

- iArgIndex - AddArgument が返すハンドル。
- iMaxCharacters - 引数の最大入力文字数。
- iMaxVisibleCharacters - 表示される最大文字数で、iMaxCharacters 以下の値である必要があります。

注釈

この関数は、ある引数の最大長、およびインターフェースで表示する文字数（入力フィールドの長さ）を設定します。

SetNumericLimits

SetNumericLimits(int iArgIndex,float fLow,float fHigh)

パラメータ

- iArgIndex - **AddArgument** 『18 ページ 』が返すハンドル。
- Low - 下限
- High - 上限

注釈

この関数は、フィールドに入力される実数値と整数値の受け取り可能な範囲を設定します。引数には、CHECK_RANGE_LOW フラグと CHECK_RANGE_HIGH フラグを設定しておく必要があります。

SetEntryPoint

SetEntryPoint*(string sName)*

パラメータ

- sName - エントリポイント関数の名前。

注釈

この関数は、エントリポイント関数の名前を登録します。これは **Network Monitor** が実行の開始点として呼び出す関数です。デフォルト値は"main"です。

SetAuthor

SetAuthor(string sName)

パラメータ

- sName - スクリプトの作成者の名前。

注釈

この関数は、スクリプトの作成者を設定します。これは、自分以外が作成したスクリプトをロードしたユーザーに、スクリプトの作成者を通知する説明の目的で使用されます。

SetDescription

SetDescription(string sDescription)

パラメータ

- sDescription - スクリプトの機能の説明。

注釈

スクリプトの説明は、このスクリプトが実行する内容、スクリプトに関する既知の制約の有無を数行でユーザーに数行で伝える必要があります。テキストの上限はありませんが、簡潔にまとめる必要があります。

SetMinBuildVersion

SetMinBuildVersion(int iMinBuildNumber)

パラメータ

- iMinBuildNumber - スクリプトに必要な **Network Monitor** の最小ビルド番号。

注釈

最小ビルド番号は非常に重要な設定フィールドです。これにより **Network Monitor** は、**Network Monitor** の現在のバージョンでスクリプトを実行できるかどうかが分かります。デフォルトでは、スクリプト作成者がスクリプトのテストに使用したビルド番号に設定する必要があります。

SetScriptVersion

SetScriptVersion(int iMajor,int iMinor)

パラメータ

- iMajor - スクリプトのメジャーバージョン番号。
- iMinor - スクリプトのマイナーバージョン番号。

注釈

スクリプトの作成者は、スクリプトのバージョン番号を設定する必要があります。メジャーバージョンが 0 の場合、スクリプトは"ベータ"段階であり、他のユーザーは開発用にのみ使用する必要があることを示します。スクリプトを修正するたびに、バージョン番号を増加する必要があります。メジャーバージョン番号の変更は大規模な書き直しまたは修正を表し、マイナーバージョン番号の変更は小規模な修正を意味します。

チャプター 5

TLuaDateTime

TLuaDateTime は日付と時刻の関数を提供します。時刻は、ローカル時刻を 1970 年 1 月 1 日からの経過秒数で示します。

この章で

追加

Add(TLuaDateTime DateTime)

パラメータ

- DateTime - 他のクラス関数から取得した、または構成して得られた TLuaDateTime インスタンス。

注釈

この関数は、DateTime パラメータに含まれる時刻をアセットに加えます。

作成する

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

パラメータ

- iYear - 年。例: 1972。
- iMonth - 月の番号。例: 10。
- iDay - 月内の日。例: 2。
- iHour - 使用する時で、ゼロでもかまいません。
- iHour - 使用する分で、ゼロでもかまいません。
- iHour - 使用する秒で、ゼロでもかまいません。

注釈

この関数は、絶対時刻を含む TLuaDateTime を作成します。

CreateSpan

CreateSpan(int iHour,int iMinute,int iSecond)

パラメータ

- iHour - 使用する時間で、ゼロでもかまいません。
- iMinute - 使用する分数で、ゼロでもかまいません。
- iSecond - 使用する秒数で、ゼロでもかまいません。

注釈

この関数は、絶対時刻ではなく、他の TLuaDateTime アセットから加算または減算する目的で使用できる時間の長さをもつ TLuaDateTime を作成します。

等しい

bool Equal(TLuaDateTime DateTime)

戻り値

DateTime が等しい場合は true、等しくない場合は false。

パラメータ

- DateTime - 他のクラス関数から取得した、または構成して得られた TLuaDateTime インスタンス。

Get

int Get()

戻り値

このインスタンスに含まれる秒数。

注釈

この関数は、インスタンスが含む絶対時刻の秒数を取得するために使用できます。

GetDate

string GetDate(string sFormat=NULL)

戻り値

現在の時刻を文字列で返します。フォーマットは sFormat パラメータで指定したものか、デフォルトのフォーマットです。

パラメータ

- sFormat - 返された時刻の別フォーマットをもつオプション文字列。デフォルトのフォーマットは、YY-MM-DD です。使用できるフラグについては、注釈を参照してください。

注釈

インスタンスに含まれる時刻を文字列で返します。デフォルトのフォーマットは YY-MM-DD です。独自のフォーマットを指定して、時刻を返す形式を変更できます。

フォーマットのフラグ

- %a - 省略した曜日名
- %A - 省略しない曜日名
- %b - 省略した月名
- %B - 省略しない月名
- %c - ロケールに適する日時表現
- %d - 月内の日の 10 進数表記（01〜31）
- %H - 24 時間形式の時（00〜23）
- %I - 12 時間形式の時（01〜12）
- %j - 年内の日の 10 進数表記（001〜366）
- %m - 月の 10 進数表記（01〜12）
- %M - 分の 10 進数表記（00〜59）

- %p - 現在のロケールにおける 12 時間時計の午前/午後インジケータ
- %S - 秒の 10 進数表記（00〜59）
- %U - 日曜日を週の先頭とした年内の週番号（00〜53）
- %w - 数字で表した曜日（0〜6、日曜日が 0）
- %W - 月曜日を週の先頭とした年内の週番号（00〜53）
- %x - 現在のロケールの日付表現
- %X - 現在のロケールの時刻表現
- %y - 西暦年の下 2 桁（00〜99）
- %Y - 4 桁の西暦年
- %z、%Z - タイムゾーン名またはその省略形、タイムゾーンが不明の場合は文字列なし

GetTime

string GetTime(string sFormat=NULL)

戻り値

現在の時刻を文字列で返します。フォーマットは sFormat パラメータで指定したものか、デフォルトのフォーマットです。

パラメータ

- sFormat - 返された時刻の別フォーマットをもつオプション文字列。デフォルトのフォーマットは、HH:MM:SS です。使用できるフラグについては、注釈を参照してください。

注釈

インスタンスに含まれる時刻を文字列で返します。デフォルトのフォーマットは SS-MM-HH です。独自のフォーマットを指定して、時刻を返す形式を変更できます。

フォーマットのフラグ

- %a - 省略した曜日名
- %A - 省略しない曜日名
- %b - 省略した月名
- %B - 省略しない月名
- %c - ロケールに適する日時表現
- %d - 月内の日の 10 進数表記（01〜31）
- %H - 24 時間形式の時（00〜23）
- %I - 12 時間形式の時（01〜12）
- %j - 年内の日の 10 進数表記（001〜366）
- %m - 月の 10 進数表記（01〜12）
- %M - 分の 10 進数表記（00〜59）
- %p - 現在のロケールにおける 12 時間時計の午前/午後インジケータ
- %S - 秒の 10 進数表記（00〜59）
- %U - 日曜日を週の先頭とした年内の週番号（00〜53）
- %w - 数字で表した曜日（0〜6、日曜日が 0）
- %W - 月曜日を週の先頭とした年内の週番号（00〜53）

- %x - 現在のロケールの日付表現
- %X - 現在のロケールの時刻表現
- %y - 西暦年の下 2 桁（00〜99）
- %Y - 4 桁の西暦年
- %z、%Z - タイムゾーン名またはその省略形、タイムゾーンが不明の場合は文字列なし

Greater

bool Greater(TLuaDateTime DateTime)

戻り値

DateTime の方が小さい場合は true、そうでない場合は false。

パラメータ

- DateTime - 他のクラス関数から取得した、または構成して得られた TLuaDateTime インスタンス。

GreaterOrEqual

bool GreaterOrEqual(TLuaDateTime DateTime)

戻り値

DateTime の方が小さいかまたは等しい場合は true、そうでない場合は false。

パラメータ

- DateTime - 他のクラス関数から取得した、または構成して得られた TLuaDateTime インスタンス。

Less

bool Less(TLuaDateTime DateTime)

戻り値

DateTime の方が大きい場合は true、そうでない場合は false。

パラメータ

- DateTime - 他のクラス関数から取得した、または構成して得られた TLuaDateTime インスタンス。

LessOrEqual

bool LessOrEqual(TLuaDateTime DateTime)

戻り値

DateTime の方が大きいかまたは等しい場合は true、そうでない場合は false。

パラメータ

- DateTime - 他のクラス関数から取得した、または構成して得られた TLuaDateTime インスタンス。

NotEqual

bool NotEqual(TLuaDateTime DateTime)

戻り値

DateTime と等しくない場合は true、等しい場合は false。

パラメータ

- DateTime - 他のクラス関数から取得した、または構成して得られた TLuaDateTime インスタンス。

設定

Set(int iNewTime)

パラメータ

- iNew - 秒単位での時刻のオフセット、または 1970 年 1 月 1 日からの秒単位での絶対時刻。

注釈

この関数は TLuaDateTime インスタンスを作成するために使用できます。このインスタンスを後で別の TLuaDateTime インスタンスと加算、減算、比較することができます。

サブ

Sub(TLuaDateTime DateTime)

パラメータ

- DateTime - 他のクラス関数から取得した、または構成して得られた TLuaDateTime インスタンス。

注釈

この関数は、アセットに格納されている時間から、DateTime パラメータがもつ時間を減算します。

チャプター 6

TLuaDB

このクラスは、SQL データベースを照会する関数を提供します。

この章で

サンプルスクリプト:TLuaDB

```
--Create new DB asset
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
```

ColCount

int ColCount()

戻り値

正常終了したクエリーの結果として列数を返します。

接続

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

戻り値

接続が正常に実行された場合は true、エラーが発生した場合は false です。

パラメータ

- sConnectString - クライアント固有の接続文字列。詳細については注釈を参照してください。
- iClientType - 通信を行うクライアントのタイプ。後述のオプションを参照してください。

注釈

接続文字列はクライアントに固有のものです。現在サポートしているクライアントを示します。

CLIENT _ODBC

接続文字列の例:

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

この接続文字列は"test"という名前のデータソースを使用し、ユーザー名（UID）"test"とパスワード（PWD）"test"を DSN に渡します。

ユーザー名とパスワードが不要の場合には、接続は次のように作成できます。

```
sConnectString = "DSN=test;";
```

Network Monitor はサービスとして動作し、データソースはシステムのデータソースである必要があります。これはユーザーの DSN も活用できる IDE とは異なります。

CLIENT _SQLSERVER

接続文字列の例:

```
sConnectString = "myserver@mydatabase";
```

SQL Server 2000 の名前付きインスタンスに接続する場合

```
sConnectString = "myserver\\instance_name@mydatabase";
```

CLIENT_MYSQL

接続文字列の例（デフォルトのポート番号 3306 を使用）:

```
sConnectString =  myserver@mydatabase
```

カスタムポートをリスンするサーバーに接続

```
sConnectString = "myserver:portnumber@mydatabase";
```

MySQL クライアントライブラリをインストールする必要があり（"MySQL Workbench"または "Connector/C (libmysql)"）、**Network Monitor** と IDE が検出できるように、そのライブラリをデフォルトの Windows システム PATH 変数に追加する必要があります。

Connect(2)

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

戻り値

接続が正常に実行された場合は true、エラーが発生した場合は false です。

パラメータ

- sConnectString - クライアント固有の接続文字列。詳細については注釈を参照してください。
- sUsername - 接続で使用する資格情報。
- sPassword - 接続で使用する資格情報。ユーザー名を指定する場合、空白にすることはできません。
- iClientType - 通信するクライアントのタイプ。現在の唯一のオプションは CLIENT_ODBC です。

注釈

接続文字列はクライアントに固有のものです。詳細については、**Connect** 『31 ページ 』を参照してください。

実行する

bool Execute*(string sSQL)*

戻り値

クエリーが正常に実行された場合は true、エラーが発生した場合は false です。

パラメータ

- sSQL - SQL ステートメント。

注釈

SQL ステートメントの実行時にエラーが発生した場合、GetErrorDescription()関数はエラーを説明する文字列を返します。

GetCol

string GetCol(int iIndex)

戻り値

列から取得したデータ文字列を返します。

パラメータ

- iIndex - 1 から始まる列インデックス。

注釈

列インデックスは 1 から始まることに注意してください。ColCount()が 10 を返した場合、有効なインデックス範囲は 1～10 です。データの型を取得するには、GetColType()を呼び出します。

GetCol_AsDateTime

TLuaDateTime GetCol_AsDateTime(int iIndex)

戻り値

列から取得した日付と時間をもつ TLuaDateTIme 構造を返します。

パラメータ

- iIndex - 1 から始まる列インデックス。

注釈

列インデックスは 1 から始まることに注意してください。ColCount()が 10 を返した場合、有効なインデックス範囲は 1〜10 です。列が日付と時刻の型でない場合、この関数を呼び出す必要はありません。

GetColType

int GetColType(int iIndex)

戻り値

列が含むデータの型を表す整数を返します。

パラメータ

- iIndex - 1 から始まる列インデックス。

注釈

GetCol()関数は常にデータを文字列で返します。GetColType 関数を使用して、抽出後の文字列を変換する列の型を特定できます。次の型が存在します。

- TYPE_BOOL - ブール値
- TYPE_NUMERIC - 数値
- TYPE_SHORT - 単精度
- TYPE_LONG - 長精度
- TYPE_DOUBLE - 倍精度（実数）
- TYPE_DATETIME - 日付/時刻
- TYPE_STRING - 文字列
- TYPE_UNKNOWN - 不明なフィールド型/未サポート
- TYPE_BYTES - バイト
- TYPE_LONG_BINARY - 長精度バイナリ
- TYPE_LONG_CHAR - 長精度 CHAR
- TYPE_BLOB - バイナリデバイス
- TYPE_DBMS_SPECIFIC - クライアント固有データ（変換なし）

GetErrorDescription

string GetErrorDescription(void)

戻り値

TLuaDB API が生成した最後のエラー説明を返します。

NextRow

bool NextRow()

戻り値

新しい結果が取得された場合は true、クエリーの結果がこれ以上存在しない場合は false。

注釈

この関数は、以前の Execute 関数の呼び出しで生成された新しい結果を取得します。この関数は、最初の ColCount、GetCol、または GetColType の関数呼び出しを行う前に呼び出す必要があります。

ResultAvailable

bool ResultAvailable()

戻り値

実行した SQL ステートメントが何らかの結果を返した場合は true を返します。

注釈

実行後に結果を返すステートメントの挿入、更新、または削除は行わない設計になっています。ステートメントが正常に実行されたかどうかを調べるには、この関数を呼び出す前に、Execute 関数の戻り値を使用する必要があります。

チャプター 7

TLuaDNS

このクラスは、DNS サーバーを照会する関数を提供します。

この章で

サンプルスクリプト:TLuaDNS

```
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
```

開始

bool Begin(bool bUselocalhost=false)

戻り値

関数が正常に終了した場合は非ゼロの値を返し、それ以外の場合は 0 を返します。

パラメータ

- bUselocahost - **Network Monitor** のホストマシン上の DNS サーバーを照会する場合は true を設定します。

注釈

この関数は DNS へのクエリーを開始します。Begin の前に Query を呼び出すと、予期しない結果になる可能性があります。

終了

void End()

注釈

この関数はトランザクションを終了し、アセットをリセットして新規のクエリーを実行できるようにします。この関数を呼び出さない場合、次のクエリー結果は予測できません。

GetErrorDescription

string GetErrorDescription(void)

戻り値

TLuaDNS API が生成した最後のエラー説明を返します。

次

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);

戻り値

関数が正常に終了した場合は非ゼロの値を返し、それ以外の場合は 0 を返します。

パラメータ

Record - DNS サーバーに照会した情報を受け取る DA DNS レコードタイプ。

注釈

この関数は新しいレコードを正常に取得した場合に true を返し、取得するレコードがもう存在しない場合に false をかえします。

Query

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

戻り値

関数が正常に終了した場合は非ゼロの値を返し、それ以外の場合は 0 を返します。

パラメータ

- sDomainName - 照会先のドメイン。
- iRecordType - 注釈に記載しているいずれかのレコードタイプ。
- bBypassCache - デフォルトは false です。true に設定した場合には、問い合わせはローカルリゾルバを迂回して DNS サーバーに直接に照会します。

注釈

この関数は DNS サーバーにクエリーを送信します。結果は、Next()関数を 1 回以上呼び出すことで抽出できます。

次のレコードタイプが照会可能です。

- LuaDNS_TYPE_PTR
- LuaDNS_TYPE_TEXT
- LuaDNS_TYPE_SOA
- LuaDNS_TYPE_CNAME
- LuaDNS_TYPE_MX
- LuaDNS_TYPE_NS

- LuaDNS_TYPE_A

DNS レコードタイプのリファレンスについては、次のサイトを参照してください。
http://www.iana.org/assignments/dns-parameters 『http://www.iana.org/assignments/dns-parameters』

TLuaDNS_ARecord

クラスメンバー

- int m_iTTL
- string m_sIPAddress;

TLuaDNS_CNAMERecord

クラスメンバー

- int m_iTTL
- string m_sHostname

TLuaDNS_MXRecord

クラスメンバー

- int m_iPreference
- string m_sNameExchange

TLuaDNS_NSRecord

クラスメンバー

- int m_iTTL
- string m_sHostname

TLuaDNS_PTRRecord

クラスメンバー

- int m_iTTL
- string m_sHostname

TLuaDNS_SOARecord

クラスメンバー

- int m_iTTL
- string m_sPrimaryServer

- string m_sAdministrator
- int m_iSerialNo
- int m_iRefresh
- int m_iRetry
- int m_iExpire
- int m_iDefaultTTL

TLuaDNS_TXTRecord

クラスメンバー

- int StringCount*(void)*
- string GetString(int iPos)

チャプター 8

TLuaFile

このクラスは基本的なファイル処理ルーチンを提供し、バイナリファイルとテキストファイルの両方をサポートします。すべてのファイル操作はスクリプトが実行されるコンテキストを基準にします。たとえば、スクリプトを実行するアセットのアドレスが`\\myserver`の場合、ファイルパス`c:\test.txt`は`\\myserver\C$\test.txt`に変換されます。

これには1つ例外があり、それはクラスがtrueで初期化される場合です。この場合には、すべてのファイル操作は **Network Monitor** のホストマシンを基準にします。詳細については、このクラスのサンプルスクリプトの例 3 を参照してください。

この章で

サンプルスクリプト:TLuaFile

例 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
```

例 2

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

--Construct a new file device
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
```

例 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
```

閉じる

int Close()

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

注釈

ファイルを閉じ、ファイルの読み書きを不可にします。アセットを破棄する前にファイルを閉じていない場合、デストラクタがファイルを閉じます。

CopyFile

int CopyFile(string sSource,string sDest)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sSource - コピーするファイルのパスと名前。
- sDest - 新しいファイルの新しいパスと名前。

注釈

この関数はファイルをコピーします。この関数ではディレクトリをコピーできません。

CreateDirectory

int CreateDirectory(string sPath)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sPath - 作成するディレクトリのパス。

注釈

作成するディレクトリの親ディレクトリが存在している必要があり、存在していない場合、この関数は失敗します。

DeleteDirectory

int DeleteDirectory(string sDirectory)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sDirectory - 削除するディレクトリのパス。

注釈

ディレクトリは空である必要があり、ルートディレクトリまたは現在の作業ディレクトリを指定することはできません。

DeleteFile

int DeleteFile(string sFileName)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sFileName - 削除するファイルのパスと名前。

注釈

この関数はファイルを削除します。この関数ではディレクトリを削除できません。

DoesFileExist

int DoesFileExist(string sFile);

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sFile - ファイルのパスと名前。

注釈

ファイルが存在する場合は非ゼロ値を返します。

GetDirectoryList

string GetDirectoryList(string sDirectory)

戻り値

サブディレクトリを改行文字で区切った文字列を返します。ディレクトリが見つからない場合は 0 を返し、特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sDirectory - 検索するディレクトリのパス。

注釈

返された文字列には、指定したディレクトリのすべてのサブディレクトリが含まれます。各ディレクトリはフルパス付きで返されます。文字列内のディレクトリは、改行文字で区切られます。

GetFileAccessedTime

TLuaDateTime GetFileAccessedTime(string sFilename)

戻り値

ファイルが最後にアクセスされた時刻をもつ TLuaDateTime アセットを返します。その他の場合は 0 をもつ TLuaDateTime を返し、特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sFilename - ファイルのパスと名前。

注釈

TLuaDateTime アセットを使ってアセットに格納されている時刻を表す文字列を構成するか、その時刻を他の TLuaDateTime アセットと比較します。

GetFileCreatedTime

TLuaDateTime GetFileCreatedTime(string sFilename)

戻り値

ファイルが作成された時刻を含む TLuaDateTime アセットを返します。その他の場合は 0 をもつ TLuaDateTime を返し、特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- pFilename - ファイルのパスと名前。

注釈

TLuaDateTime アセットを使ってアセットに格納されている時刻を表す文字列を構成するか、その時刻を他の TLuaDateTime アセットと比較します。

GetFileList

string GetFileList(string sDirectory,string sWildCard)

戻り値

リストのファイルを改行文字で区切った文字列を返します。ディレクトリが見つからない場合は 0 を返し、特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sDirectory - ディレクトリのパス。
- sWildCard - 目的のファイルをフィルター処理するワイルドカード。ディレクトリ内のすべてのファイルを取得するには、*.*のようにワイルドカードを使用します。

注釈

返された文字列には、指定したワイルドカードに一致するディレクトリ内のすべてのファイルが含まれます。各ファイルはフルパス付きで返されます。文字列内のファイルは、改行文字で区切られます。

GetFileModifiedTime

TLuaDateTime GetFileModifiedTime(string sFilename)

戻り値

ファイルが最後に変更された時刻をもつTLuaDateTimeアセットを返します。その他の場合は0をもつ TLuaDateTime を返し、特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sFilename - ファイルのパスと名前。

注釈

TLuaDateTimeアセットを使ってアセットに格納されている時刻を表す文字列を構成するか、その時刻を他の TLuaDateTime アセットと比較します。

GetFileSize

int GetFileSize(string sFile)

戻り値

関数が正常に実行された場合はファイルサイズをバイト単位で返します。ファイルにアクセスできなかった場合の戻り値は-1 です。

パラメータ

- sFile - ファイルのパスと名前。

注釈

この関数は 2^31 バイト未満のファイルに限定されます。

GetFileSizeMB

int GetFileSizeMB(string sFile)

戻り値

関数が正常に実行された場合はファイルサイズをMB単位で返します。ファイルにアクセスできなかった場合の戻り値は-1 です。

パラメータ

- sFile - ファイルのパスと名前。

GetFileStatus

int GetFileStatus(string sFile)

戻り値

関数が正常に実行された場合はファイルの現在の状態を表す値を返します。ファイルにアクセスできなかった場合の戻り値は-1 です。

パラメータ

- sFile - ファイルのパスと名前。

注釈

この関数は、ファイルの状態を表す次のフラグの組み合わせを返します。

FILE_NORMAL = 0x00	通常のファイル。
FILE_READONLY = 0x01	ファイルは読み取り専用です。
FILE_HIDDEN = 0x02	ファイルは非表示です。
FILE_SYSTEM = 0x04	ファイルはシステムファイルです。
FILE_VOLUME = 0x08	ファイルはボリュームです。
FILE_DIR = 0x10	ファイルはディレクトリです。
FILE_ARCHIV = 0x20	ファイルにはアーカイブビットが設定されています。

MoveFile

int MoveFile(string sSource,string sDest)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sSource - 移動するファイルのパスと名前。sDest - ファイルの新しいパスと名前。

注釈

この関数はファイルを移動します。ファイルの移動先のディレクトリが存在している必要があります。存在していない場合、関数は失敗します。この関数ではディレクトリを移動できません。

開く

int Open(string sFileName, bool bCreate=false)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sFileName - 開くファイルまたは作成するファイルの名前とパス。bCreate - 非ゼロを設定した場合、ファイルが作成されます。指定したファイルが既に存在している場合、ファイルの内容が破棄されます。

注釈

bCreate にゼロが設定され、かつファイルが存在しない場合、この関数は失敗します。

読み取り

string,int Read(int iSize)

戻り値

関数が正常に実行された場合は文字列を返し、iSize には返されるデータの大きさが設定されます。正常に実行されなかった場合は nil を返します。特定のエラーコードはグローバル関数 GetLastError を呼び出すことにより取得できます。

パラメータ

- iSize - 読み取るデータのバイト数。

注釈

この関数はファイルから指定された大きさのデータを読み出し、ファイルポインタを同じ大きさだけ前に移動します。ファイルの終わりに到達すると読み取りは停止し、そこまでに読み取ったデータを返します。

ReadData

local data,int ReadData(int iSize)

戻り値

この関数は、ConvertFromUTF16 と組み合わせてのみ使用できる特殊なデータタイプを返します。関数が正常に実行された場合、iSize には返されるデータの大きさが設定されます。正常に実行されなかった場合は nil を返します。特定のエラーコードはグローバル関数 GetLastError を呼び出すことにより取得できます。

パラメータ

- iSize - 読み取るデータのバイト数。

注釈

この関数はファイルから指定された大きさのデータを読み出し、ファイルポインタを同じ大きさだけ前に移動します。ファイルの終わりに到達すると読み取りは停止し、そこまでに読み取ったデータを返します。

RenameFile

int RenameFile(string sOrgFile,string sNewFile)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sOrgFile - 名前を変更するファイルのパスと名前。
- sNewFile- ファイルの新しいパスと名前。

注釈

この関数はファイルの名前を変更します。この関数ではディレクトリの名前を変更できません。

SeekFromCurrent

int SeekFromCurrent(int iNumberofBytes)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- iNumberofBytes - 現在位置を基準にして、ファイルポインタの移動量を示すバイト数。

注釈

この関数は、現在位置を基準にしてファイルポインタを移動します。負の値と正の値の両方を指定できます。このポインタはファイルの終わりを超えた位置に移動可能で、その場合にはファイルの終わりのマーカーがクリアされます。

SeekFromEnd

int SeekFromEnd(int iNumberofBytes)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- iNumberofBytes - ファイルの終わりの位置を基準にして、ファイルポインタの移動量を示すバイト数。

注釈

この関数は、ファイルポインタの現在位置をファイルの終わりから iNumberofBytes の量だけ移動します。ファイルポインタをファイル内の前方に移動するには、iNumberofBytes を負にする必要があ

ることに注意してください。

SeekFromStart

int SeekFromStart(int iNumberofBytes)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- iNumberofBytes - ファイルの先頭位置を基準にして、ファイルポインタの移動量を示すバイト数。

注釈

この関数は、ファイルポインタの現在位置をファイルの先頭から iNumberofBytes の量だけ移動します。このポインタはファイルの終わりを超えた位置に移動可能で、その場合にはファイルの終わりのマーカーがクリアされます。

書き込み

int Write(string sData,int iSize)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sData - ファイルに書き込むデータの配列。
- iSize - 書き込むデータの大きさ。

注釈

この関数は、ファイルポインタの現在位置から書き込みを実行し、現在のファイルの終わりを超えた場合はファイルを拡張します。開いたファイルが書き込み保護されている場合、関数は失敗します。

チャプター 9

TLuaFTPClient

このクラスは、基本的な FTP 操作が可能な FTP クライアントを実装します。

この章で

サンプルスクリプト:TLuaFTPClient

```
--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()
```

ChangeDirectory

int ChangeDirectory(string sDir)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル

関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sDir - 新しい現在のディレクトリのパス。

注釈

ディレクトリが存在しない場合、この関数は失敗します。

閉じる

Close()

注釈

この関数は、FTP サーバーへの現在の接続を閉じます。現在の接続を閉じるには、この関数を呼び出す必要があります。

CloseFile

void CloseFile()

注釈

OpenFile で開いたファイルを閉じます。

接続

bool Connect(unsigned int iPort=21)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sUsername - ユーザー名をもつ文字列。
- sPassword - パスワードをもつ文字列。
- iPort - デフォルトはポート 21（標準 FTP ポート）。

注釈

この関数は、指定されたユーザー名とパスワードを使って FTP サーバーに接続します。FTP サーバーが別のポートにバインドされている場合は、ポートをデフォルトのポート 21 から変更できます。

CreateDirectory

int CreateDirectory(string sDir)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sDir - 作成するディレクトリのパス。

注釈

作成する新しいディレクトリの親ディレクトリが存在している必要があります。存在していない場合、この関数は失敗します。

DeleteDirectory

int DeleteDirectory(string sDir)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sDir - 削除するディレクトリのパス。

注釈

ディレクトリが空ではない場合、関数は失敗します。

DeleteFile

int DeleteFile(string sFilename)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sFilename - 削除するファイルのパスと名前。

注釈

ファイルが存在しない場合、この関数は失敗します。

FindDirectory

string FindDirectory()

戻り値

リストしたディレクトリを改行文字で区切った文字列を返します。ディレクトリが見つからない場

合は0を返し、特定のエラーコードはグローバル関数のGetLastErrorを呼び出すことにより取得できます。

注釈

返された文字列には、現在のディレクトリのすべてのサブディレクトリが含まれます。各ディレクトリはフルパス付きで返されます。文字列内のディレクトリは、改行文字で区切られます。

FindFile

string FindFile(string sWildcard)

戻り値

リストのファイルを改行文字で区切った文字列を返します。ディレクトリが見つからない場合は0を返し、特定のエラーコードはグローバル関数のGetLastErrorを呼び出すことにより取得できます。

パラメータ

- sWildCard - 目的のファイルをフィルター処理するワイルドカード。ディレクトリ内のすべてのファイルを取得するには、*.*のようにワイルドカードを使用します。

注釈

返された文字列には、指定したワイルドカードに一致する現在のディレクトリ内のすべてのファイルが含まれます。各ファイルはフルパス付きで返されます。文字列内のファイルは、改行文字で区切られます。

GetCurrentDirectory

int GetCurrentDirectory(string sDir)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は0です。特定のエラーコードはグローバル関数のGetLastErrorを呼び出すことにより取得できます。

パラメータ

- sDir - 新しい現在のディレクトリのパス。

注釈

ディレクトリが存在しない場合、この関数は失敗します。

GetFileModifiedTime

TLuaDateTime GetFileModifiedTime(string sFilename)

戻り値

関数が正常に実行された場合は、ファイルを最後に変更した時刻をもつTLuaDateTimeアセットを

返します。その他の場合は、0 に設定した TLuaDateTime アセットを返します。

パラメータ

- sFilename - 現在のディレクトリにあるファイルの名前。

注釈

この関数が正常に実行されるには、ファイルが現在のディレクトリに存在している必要があります。相対パスは機能しません。

GetFileSize

int GetFileSize(string sFilename)

戻り値

関数が正常に実行された場合はファイルサイズをバイト単位で返します。ファイルにアクセスできなかった場合の戻り値は-1 です。

パラメータ

- sFilename - ファイルのパスと名前。

注釈

この関数は 2^31 バイト未満のファイルに限定されます。

OpenFile

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

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sFilename - 開くファイルの名前。
- bWrite - 非ゼロの場合、ファイルは書き込みモードで開きます。ゼロの場合、ファイルは読み取り専用で開きます。
- bText - 非ゼロの場合、ファイルはテキスト変換モードで開きます。その他の場合、バイナリモードで開きます。

注釈

この関数はファイルを FTP サーバー上で開きます。ファイルを開いた後、Write 関数と Read 関数を呼び出すことができます。ファイルを閉じるには、CloseFile 関数を使用します。

読み取り

string Read(int iSize)

戻り値

関数が正常に実行された場合はファイルから読み取ったデータ配列、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- iSize - ファイルから読み出す大きさです。関数が戻ると、このパラメータには読み出したデータの大きさが入りますが、指定した大きさ未満であることがあります。

注釈

ファイルが開いていない場合、この関数は失敗します。

RenameFile

bool RenameFile(string sOldname,string sNewname)

戻り値

関数が正常に実行された場合は true、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sOldname - 名前を変更するファイルの古い名前。sNewname - ファイルの新しい名前。

注釈

2 番目のパラメータに指定されたものと同じ名前のファイルが存在する場合、この関数は失敗します。

書き込み

int Write(string sData,int iSize)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sData - ファイルに書き込むデータの配列。
- iSize - 書き込む配列の大きさ。

注釈

ファイルが開いていない場合、またはファイルが読み取りモードで開いている場合は、この関数は失敗します。FTP サーバー上のストレージが不足している場合も、この関数は失敗します。

チャプター 10

TLuaHTTPClient

このクラスは基本的な HTTP クライアントを実装します。

この章で

サンプルスクリプト:TLuaHTTPClient

```
--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("Cookie #"..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
```

接続

bool Connect(bool bSecure,unsigned short iPort)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- iPort - 接続先のポート番号で、デフォルトはポート 80 です。
- bSecure - HTTPS を使って接続する場合は、非ゼロに設定します。
- sUsername - 認証が必要なサーバー用のオプションのユーザー名。
- sPassword - 認証が必要なサーバー用のオプションのパスワード。

注釈

この関数は、指定したパラメータを使用して HTTP サーバーに接続します。この関数は、このクラスの他の関数の前に呼び出す必要があります。

閉じる

Close()

注釈

開いている接続を閉じます。

Get

int Get(string sUrl,string sHeaders=NULL)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sUrl - サイトのベース URL からの相対 URL。
- sHeaders - 要求と共に送信するヘッダーを含むオプションのヘッダー文字列。

注釈

接続は常にアセットのコンテキストで開かれるので、この関数に指定する URL はベース URL からの相対 URL である必要があります。

Post

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

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sUrl - サイトのベース URL からの相対 URL。
- sHeaders - 要求と共に送信するヘッダーを含むオプションのヘッダー文字列。
- sData - Post 要求に含めるオプションデータ。

注釈

接続は常にアセットのコンテキストで開かれるので、この関数に指定する URL はベース URL からの相対 URL である必要があります。指定するヘッダーはすべて、CR/LF のペアで終了する必要があります。

GetContent

string GetContent(int iSize)

戻り値

正常に実行された場合は、GET 要求の内容部分をもつ文字列を返し、その他の場合は nil を返します。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- iSize - 関数が返す文字列の大きさを設定します。

注釈

内容とは、要求が返したデータ（ヘッダーの後にある）を指します。

GetHeadersRaw

string GetHeadersRaw()

戻り値

正常に実行された場合は、要求が返したヘッダーを含む文字列です。その他の場合は、空の文字列です。

注釈

ヘッダーはサーバーから送信されたとおりに返されます。

GetHeaderLocation

string GetHeaderLocation()

戻り値

正常に実行された場合は、"Location:"ヘッダーを含む文字列です。その他の場合は、空の文字列で

す。

GetHeaderContentLength

int GetHeaderContentLength()

戻り値

要求の内容部分のバイト数です。

GetHeaderContentType

string GetHeaderContentType()

戻り値

正常に実行された場合は、"Content-Type:"ヘッダーを含む文字列です。その他の場合は、空の文字列です。

GetHeaderContentTransferEncoding

string GetHeaderContentTransferEncoding()

戻り値

正常に実行された場合は、"Transfer-Encoding:"ヘッダーを含む文字列です。その他の場合は、空の文字列です。

GetHeaderCookies

string GetHeaderCookies()

戻り値

正常に実行された場合は、サーバーが返したすべての Cookie を改行文字で区切った文字列を返します。その他の場合は、空の文字列を返します。

GetHeaderCookie

string GetHeaderCookie(int iIndex)

戻り値

要求された Cookie 文字列を含む文字列。

パラメータ

- iIndex - 返す Cookie を指定するゼロから始まるインデックス。

注釈

負数または範囲外のインデックスを指定した場合は、空の文字列が返されます。

GetHeaderCookieCount

int GetHeaderCookieCount()

戻り値

要求が返した Cookie の数。

GetHeaderDate

string GetHeaderDate()

戻り値

正常に実行された場合は、"Date:"ヘッダーを含む文字列です。その他の場合は、空の文字列です。

GetHeaderExpires

string GetHeaderExpires()

戻り値

正常に実行された場合は、"Expires:"ヘッダーを含む文字列です。その他の場合は、空の文字列です。

GetHeaderHost

string GetHeaderHost()

戻り値

正常に実行された場合は、"Host:"ヘッダーを含む文字列です。その他の場合は、空の文字列です。

チャプター 11

TLuaICMP

このクラスは、ネットワーク接続の診断に使用できる ping 関数およびルートのトレース関数を提供します。

この章で

サンプルスクリプト:TLuaICMP

```
--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);
```

BeginTrace

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

戻り値

この関数はトレースが正常に実行された場合に true を返し、失敗した場合に false を返します。

パラメータ

- iPacketsToSend - 1～255 の整数。
- iPacketSize - 送信パケット長を表す 0～65500 の整数。
- bNoFragment - 送信パケットにフラグメンテーションが発生しないようにするには true を設定します。このオプションを設定し、かつパケットにフラグメンテーションが発生した場合、関数は失敗して false を返します。
- iTimeoutMs - パケットが返されるまで関数が待機する最大時間（単位: ms）。

注釈

bNoFragment を true に設定すると、あるルートの最大フレームサイズをテストできます。フラグメンテーションにより関数が失敗するまで、iPacketSize を調整できます。

EndTrace

EndTrace()

注釈

この関数は、使用したリソースのクリーンアップを実行します。個々の BeginTrace()呼び出しに対して、この関数を呼び出す必要があります。

NextTraceResult

bool NextTraceResult(TLuaICMPTraceResult Result)

戻り値

この関数は、結果が取得可能な間は true を返します。

パラメータ

- Result - 現在のホップの結果を受け取る **TLuaICMPTraceResult** 変数。

注釈

結果セット全体に対して繰り返し実行するには、null が返されるまでこの関数を呼び出します。

Ping

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

戻り値

この関数は、操作結果を含む TLuaICMPPingResult アセットを返します。

パラメータ

- iPacketsToSend - 1〜255 の整数。
- iPacketSize - 送信パケット長を表す 0〜65500 の整数。
- bNoFragment - 送信パケットにフラグメンテーションが発生しないようにするには true を設定します。このオプションを設定し、かつパケットにフラグメンテーションが発生した場合、関数は失敗して false を返します。
- iTimeoutMs - パケットが戻ってくるまで関数が待機する最大時間（単位: ms）。

注釈

bNoFragment 引数を設定すると、あるルートに対して可能な最大のフレームワーク長をテストできます。

チャプター 12

TLuaICMPPingResult

TLuaICMPPingResult は読み取り専用のクラスです。

クラスメンバー

- int m_iRoundTripTimeMs
- float m_fPacketloss

チャプター 13

TLuaICMPTraceResult

TLuaICMPTraceResult は読み取り専用のクラスです。

クラスメンバー

- string m_IP
- string m_Hostname
- int m_iRoundTripTimeMs

チャプター 14

TLuaPowershell

Powershell のコマンド文字列を実行します。標準の Powershell コマンド出力、エラー出力、エラーコード、およびエラーの説明を返します。

必要条件

TLuaPowershell ライブラリは、WinRS（Windows Remote Shell）を使用して Windows リモート管理（WinRM）対応アセットに接続します。

WinRM/WinRS に関する Microsoft からの情報 『 http://msdn.microsoft.com/en-us/library/aa384426%28v=vs.85%29.aspx を見て 』:

"WinRM（Windows リモート管理: Windows Remote Management）は、標準の SOAP（簡易オブジェクトアクセスプロトコル: Simple Object Access Protocol）ベースのファイアウォール対応プロトコルである WS 管理プロトコルを Microsoft が実装したものであり、各種ベンダーのハードウェアとソフトウェアの相互運用を可能にします。

WinRM スクリプトオブジェクト、WinRM コマンドラインツール、または Windows のリモートシェルコマンドラインツールである WinRS を使用すると、BMC（ベースボード管理コントローラ: Baseboard Management Controller）を備えているローカルコンピュータやリモートコンピュータから管理データを取得することができます。コンピュータで WinRM を含む Windows ベースの OS バージョンを実行している場合、管理データは WMI（Windows Management Instrumentation）が提供します。"

アセット上で WinRM を有効にするには、コマンドプロンプトに次のコマンドを入力します。

```
winrm /quickconfig
```

Powershell コマンドやスクリプトを実行できますが、WinRS を使って実行するコマンドまたはスクリプトには、ユーザーインターフェースとの依存関係があってはならないことに注意してください。たとえば、ローカルコンソールに"任意のキーを押してください”のようなプロンプトを表示したり、その他の何らかの対話的な応答を要求するコマンドは実行できません。

この章で

サンプルスクリプト:TLuaPowershell

```
--executes a Powershell command string

bExitStatus = false
PS = TLuaPowershell()
bStatus = PS:Open(5985, false)

if bStatus == true then
   sCmd = "Get-Date\n"
   bExec = PS:ExecuteCommand(sCmd)
   if bExec == true then
          sStatus = PS:GetStdOut()
          bExitStatus = true
   else
          sStatus = "Execute failed. " .. PS:GetStdErr()
   end
else
   sStatus = "Failed to open connection. "  .. PS:GetErrorDescription()
end

SetExitStatus(sStatus, bExitStatus)
```

サンプルスクリプト:TLuaPowershell（Windows スクリプト作成）

```
--Create a script file named test.vbs in your C:\temp folder
--(on the remote asset) for this sample to work.
--The file should look like this:

strFile = WScript.Arguments(0)
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objFile = objFSO.GetFile(strFile)
strFileSize = objFile.Size
WScript.Echo strFile & " is " & strFileSize & " bytes."

--The script takes one parameter. That is the path to a file
--used for checking the file size.

bExitStatus = false
PS = TLuaPowershell()
bStatus = PS:Open(5985, false)

if bStatus == true then
   sCmd = "cscript /Nologo C:\\temp\\test.vbs \n"
   bExec = PS:ExecuteCommand(sCmd)
   if bExec == true then
          sStatus = PS:GetStdOut()
          bExitStatus = true
   else
          sStatus = "Execute failed. " .. PS:GetStdErr()
   end
else
   sStatus = "Failed to open connection. "  .. PS:GetErrorDescription()
end

SetExitStatus(sStatus, bExitStatus)
```

開く

bool Open(unsigned short _iPort,bool bSecure=true,int dwWait=2500,const char *_pWorkingDir=nullptr)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードは GetStdErr() を呼び出すことにより取得できます。

パラメータ

- iPort - WinRM（Windows リモート管理）が使用する TCP ポート。
- bSecure - SSL を使用する場合は true を設定し（デフォルト）、非 SSL 接続の場合は false を設定します。
- dwWait - 接続のタイムアウト値。

ExecuteCommand

ExecuteCommand(const char *pCommand)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードは GetStdErr を呼び出すことにより取得できます。

パラメータ

- pCommand - Powershell のコマンド文字列。

GetStdOut

string GetStdOut(void)

戻り値

リモートホストからの標準出力を返します。

GetStdErr

string GetStdErr(void)

戻り値

リモートホストからの標準エラー出力を返します。

GetErrorDescription

string GetErrorDescription(void)

戻り値

リモートホストが生成した最後のエラー説明を文字列として返します。

GetErrorCode

dWord GetErrorCode(void)

戻り値

リモートホストが生成した最後のエラーコードを文字列として返します。

チャプター 15

TLuaRegistry

このクラスは Windows のレジストリへのアクセスを提供します。レジストリを操作するときに、このドキュメントで使用する重要な用語が 2 つあります。

- キー - レジストリハイブ内のエンティティで、サブキーと値をもつことができます。
- 値 - サブエントリをもたない、データを含むエンティティ。データには各種の型があります。この実装でサポートする型は、文字列、整数、およびバイナリデータです。

すべてのレジストリ操作は、スクリプトが実行されているコンテキストを基準にします。これには 1 つ例外があり、それはクラスが true で初期化されている場合です。この場合、すべての操作は **Network Monitor** のホストマシンを基準にします。詳細については、このクラスのサンプルスクリプトの例 2 を参照してください。

この章で

サンプルスクリプト:TLuaRegistry

```
--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 KNM") == true then sValue = "";
bOK,sValue = Reg:ReadValue("INSTALL_SHORTCUTFOLDER",sValue);
SetExitStatus("KNM install path is -> "..sValue,bOK);
else
SetExitStatus("Could not open registry location",false);
end
```

BeginEnumValue

BeginEnumValue()

注釈

この関数は、EnumValue()の最初の呼び出しより前に呼び出す必要があります。この関数により、EnumValue()は確実に値リストの先頭から開始されます。EnumValue()より前にこの関数を呼び出すことに失敗すると、予測できない結果になります。

閉じる

Close()

注釈

この関数は、現在開いているレジストリ接続を閉じます。

作成する

bool Create(string sKey)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。エラー説明は GetErrorDescription() を呼び出すことにより取得できます。

パラメータ

- sKey - 作成するキー。

注釈

Create()関数は、指定されたレジストリキーを作成します。キーが既にレジストリに存在している場合は、関数はそのキーを開きます。この関数を使用して、複数のキーを一度に作成できます。たとえば、スクリプトで次の形式の文字列を指定して、3 レベルの深さのサブキーを作成できます。

subkey1\subkey2\subkey3

DeleteValue

bool DeleteValue(string sValueName)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。エラー説明は GetErrorDescription() を呼び出すことにより取得できます。

パラメータ

- sValueName - 削除する値の名前。

注釈

この関数は、現在のキーの値を削除します。この値が存在しない場合、関数は失敗します。

EnumValue

bool EnumValue(string &sValueName)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。エラー説明は GetErrorDescription() を呼び出すことにより取得できます。

パラメータ

- sValueName - 現在のキーで次に列挙されている値の名前。

注釈

現在のキー内にあるすべての変数を列挙するには、この関数が false を返すまで関数を呼び出します。この関数を初めて呼び出す前に、BeginEnumValue()を呼び出す必要があります。

例 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
```

GetErrorDescription

string GetErrorDescription()

戻り値

クラス内の任意の関数を呼び出すときに、最後に発生したエラーを説明する文字列を返します。

開く

bool Open(int iKey,string sKey)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。エラー説明は GetErrorDescription() を呼び出すことにより取得できます。

パラメータ

- iKey - レジストリハイブのいずれかを表すキー。
- bCreate - 選択したレジストリハイブ内のサブキー。

注釈

この関数は、選択したレジストリハイブ内にあるレジストリキーを開きます。プロセス（IDE または **Network Monitor**）の資格情報により、特定のキーおよびハイブへのアクセスが制限されることに注意してください。iKey 用に次の定数が定義されています:

- CLASSES_ROOT
- CURRENT_CONFIG
- CURRENT_USER
- LOCAL_MACHINE
- PERFORMANCE_DATA
- USERS

ReadValue

bool ReadValue(string sValueName,string &sData)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。エラー説明は GetErrorDescription() を呼び出すことにより取得できます。

パラメータ

- sValueName - 取得する値の名前。
- sData - 関数が返すデータ。

注釈

この関数は、指定した名前をもつ値のデータを返します。値の型が文字列でない場合、この関数は失敗します。

ReadValue

bool ReadValue(string sValueName,int &iData)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。エラー説明は GetErrorDescription() を呼び出すことにより取得できます。

パラメータ

- sValueName - 取得する値の名前。
- iData - 関数が返すデータ。

注釈

この関数は、指定した名前をもつ値のデータを返します。値の型が整数でない場合、この関数は失敗します。

ReadValue

string ReadValue(string sValueName,int &iSize)

戻り値

レジストリの値に格納されているデータを返します。関数が失敗した場合、空の文字列を返します。エラーの説明は、GetErrorDescription()を呼び出すことで取得できます。

パラメータ

- sValueName - 取得する値の名前。
- iSize - 関数が返すデータのバイト数。

注釈

この関数は、指定した名前をもつ値のデータを返します。値の型が整数でない場合、この関数は失敗します。返されるデータの大きさは、iSize パラメータに格納されます。この関数が失敗した場合、iSize パラメータはゼロに設定されます。

SetValue

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

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。エラー説明は GetErrorDescription() を呼び出すことにより取得できます。

パラメータ

- sValueName - 書き込む値の名前。
- sData - 値に書き込むデータ。
- iSize - 書き込むデータのバイト数。

注釈

この関数は、指定したデータを値に書き込みます。

SetValue

bool SetValue(string sValueName,string sString)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。エラー説明は GetErrorDescription() を呼び出すことにより取得できます。

パラメータ

- sValueName - 書き込む値の名前。
- sString - 値に書き込む文字列。
- iSize - 書き込むデータのバイト数。

注釈

この関数は、指定した文字列を値に書き込みます。値が存在しない場合、この関数は失敗します。

SetValue

bool SetValue(string sValueName,int iValue)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。エラー説明は GetErrorDescription() を呼び出すことにより取得できます。

パラメータ

- sValueName - 書き込む値の名前。
- iValue - 値に書き込む整数。

注釈

この関数は指定した整数を値に書き込みます。値が存在しない場合、この関数は失敗します。

SetValueExpandedString

bool SetValueExpandedString(string sValueName,string sString)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。エラー説明は GetErrorDescription() を呼び出すことにより取得できます。

パラメータ

- sValueName - 書き込む値の名前。

- sString - 値に書き込む文字列。

注釈

この関数は、通常の SetValue 関数と同様に動作しますが、1 つ例外があります。書き込まむ文字列に、展開していない環境変数への参照を含めることができます（例: "%PATH%"）。

チャプター 16

TLuaSFTPClient

このクラスは基本的な SFTP クライアントクラスを実装します。

この章で

サンプルスクリプト:TLuaSFTPClient

```
--Demonstrates the Lua SFTP client class
--Create the client asset
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)
```

閉じる

bool Close(TLuaSFTPClientHandle FileHandle)

戻り値

操作が正常に実行された場合に true を返し、失敗した場合に false を返します。

パラメータ

- FileHandle - 以前に開いたファイルのハンドル。

CloseDir

bool CloseDir(TLuaSFTPClientDirectoryHandle Handle);

戻り値

操作が正常に実行された場合に true を返し、失敗した場合に false を返します。操作が正常に実行されると、TLuaSFTPClientDirectoryHandle ハンドルが閉じられます。

パラメータ

- Handle - OpenDir 関数が開いたハンドル。

接続

bool Connect(const unsigned int iPort=22,const unsigned short dwTimeout=25000);

戻り値

接続操作が正常に実行された場合に true を返し、失敗した場合に false を返します。

パラメータ

- sUsername - ユーザー名。
- sPassword - パスワード。
- iPort - サーバーがリスンしているポート番号。デフォルト値は 22 です。
- iTimeout - サーバーが応答を待機するタイムアウト時間（単位: ms）。デフォルト値は 25000（25 秒）です。

CreateFile

bool CreateFile(string sPath,TLuaSFTPClientHandle hHandle)

戻り値

ファイルが作成された場合に true を返し、操作が失敗した場合に false を返します。操作が正常に実行された場合、TLuaSFTPClientHandle は開いているファイルンへの参照を含みます。

パラメータ

- sPath - 作成するファイルのフルパス。パスに指定したディレクトリは存在している必要があります。存在しない場合、操作は失敗します。
- hHandle - ファイルを作成するハンドル。

注釈

新規に作成されたファイルには読み取りと書き込みのアクセス権があります。

ListDir

bool ListDir(TLuaSFTPClientDirectoryHandle Handle);

戻り値

操作が正常に実行された場合に true を返し、失敗した場合に false を返します。操作が正常に実行された場合、TLuaSFTPClientDirectoryHandle クラスでデータを取得できます。

パラメータ

- Handle - OpenDir 関数が開いたハンドル。

MkDir

bool MkDir(string sPath)

戻り値

操作が正常に実行された場合に true を返し、失敗した場合に false を返します。

パラメータ

- sPath - 作成するディレクトリのパスで、新しいディレクトリの名前を含みます。

注釈

この関数は再帰的に新しいディレクトリを作成することができません。パスにある最後のディレクトリの親ディレクトリがすべて、存在している必要があります。

OpenDir

bool OpenDir(string sPath,TLuaSFTPClientDirectoryHandle &Handle)

戻り値

操作が正常に実行された場合に true を返し、失敗した場合に false を返します。

パラメータ

- sPath - オープンするディレクトリのパス。
- Handle - 後続の操作で使用するために返すハンドル。

注釈

この関数は、ListDir 関数で内容をリスト表示できるようにディレクトリを"開き"ます。

Open_ForRead

bool Open_ForRead(string _sPath,TLuaSFTPClientHandle hHandle)

戻り値

ファイルが正常に開いた場合に true を返し、操作が失敗した場合に false を返します。

パラメータ

- sPath - ファイルのフルパス。
- hHandle - 後続の操作で使用するファイルのハンドル。

Open_ForWrite

bool Open_ForWrite(string _sPath,TLuaSFTPClientHandle hHandle)

戻り値

ファイルが正常に開いた場合に true を返し、操作が失敗した場合に false を返します。

パラメータ

- sPath - ファイルのフルパス。
- hHandle - 後続の操作で使用するファイルのハンドル。

Open_ForAppend

bool Open_ForAppend(string _sPath,TLuaSFTPClientHandle hHandle)

戻り値

ファイルが正常に開いた場合に true を返し、操作が失敗した場合に false を返します。

パラメータ

- sPath - ファイルのフルパス。hHandle - 後続の操作で使用するファイルのハンドル。

注釈

Open_ForAppend はファイルを書き込みモードで開きます。この関数が Open_ForWrite と異なる点は、2 回の書き込みの間にファイルポインタの位置が変化した場合でも、すべてのデータをファイルの終わりに書き込む点です。

読み取り

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

戻り値

操作が正常に実行された場合に true を返し、失敗した場合に false を返します。

パラメータ

- FileHandle - 以前に開いたファイルのハンドル。
- iOffset - ファイル内の読み出し位置を示すオフセットのバイト数。
- iLen - 読み出すデータ長。
- sData - 読み出したデータを入れる変数。

注釈

この関数ではテキストファイルのみから読み出すことができます。

例

```
--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)
```

削除する

bool Remove(string sPath);

戻り値

操作が正常に実行された場合に true を返し、失敗した場合に false を返します。

パラメータ

- sPath - 削除するファイルのパス。

名前の変更

bool Rename(string sPath,string sNewPath);

戻り値

操作が正常に実行された場合に true を返し、失敗した場合に false を返します。

パラメータ

- sPath - 名前を変更する既存ファイルのパス。
- sNew - 新しいファイル名のパス。

RmDir

bool RmDir(string sPath)

戻り値

操作が正常に実行された場合に true を返し、失敗した場合に false を返します。

パラメータ

- sPath - 削除するディレクトリのパス。

注釈

この関数は、空のディレクトリのみを削除できます。

書き込み

bool Write(TLuaSFTPClientHandle FileHandle,const int iOffset,string sData)

戻り値

操作が正常に実行された場合に true を返し、失敗した場合に false を返します。

パラメータ

- FileHandle - 以前に開いたファイルのハンドル。
- iOffset - ファイル内の書き込み位置を示すオフセットのバイト数。
- sData - 書き込むテキストの文字列。

例

```
--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)
```

チャプター 17

TLuaSFTPClientAttributes

このクラスには、ListDir 関数で取得されるディレクトリまたはファイルを表す属性があります。

この章で

AccessedTime

bool AccessedTime(TLuaDateTime &Time)

戻り値

値が存在する場合に true を返し、存在しない場合に false を返します。

パラメータ

- Time - ファイルが最後にアクセスされた時刻を含みます。

CreatedTime

bool CreatedTime(TLuaDateTime &Time)

戻り値

値が存在する場合に true を返し、存在しない場合に false を返します。

パラメータ

- Time - ファイルが作成された時刻を含みます。

グループを作成する

bool Group(string &sGroup)

戻り値

値が存在する場合に true を返し、存在しない場合に false を返します。

パラメータ

- sGroup - ファイルまたはディレクトリのグループの名前を含みます。

ModifiedTime

bool ModifiedTime(TLuaDateTime &Time)

戻り値

値が存在する場合に true を返し、存在しない場合に false を返します。

パラメータ

- Time - ファイルが最後に変更された時刻を含みます。

オーナー

bool Owner(string &sOwner)

戻り値

値が存在する場合に true を返し、存在しない場合に false を返します。

パラメータ

- sOwner - ファイルまたはディレクトリのオーナーの名前を含みます。

PermissionBits

bool PermissionBits(int &iPermissionsBits)

戻り値

値が存在する場合に true を返し、存在しない場合に false を返します。

パラメータ

- iPermissionsBits - ファイルまたはディレクトリの権限を表す 10 進数の値を含みます。

サイズ

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

戻り値

値が存在する場合に true を返し、存在しない場合に false を返します。

パラメータ

- iBytesHighDWord - 64 ビット整数の上位 DWORD 部分を含みます。
- iBytesLowDWord - 64 ビット整数の下位 DWORD 部分を含みます。

注釈

ファイルの大きさは、バイト単位の 64 ビット整数で報告されます。Lua には 64 ビット整数型がないので、情報は 2 つの 32 ビット整数に分割されました。ファイルの大きさが 2 GB 未満の場合、iBytesHighDWord は常に 0 です。

SizeMB

bool SizeMB(unsigned int &iSizeMB);

戻り値

値が存在する場合に true を返し、存在しない場合に false を返します。

パラメータ

- iSizeMB - ファイルの大きさ（単位: MB）の値を含みます。

注釈

Size()関数に代わる使いやすい選択肢として提供され、切り捨てたファイルサイズを返します。

チャプター 18

TLuaSFTPClientDirectoryHandle

このクラスは、OpenDir、ListDir、および CloseDir 関数と共に使用されます。

この章で

次

bool Next(TLuaSFTPClientFile &hFile)

戻り値

指定した TLuaSFTPClientFile にデータがある場合に true を返します。

注釈

ListDir 関数が返した情報をすべて取得するには、false が返されるまでこの関数を繰り返し実行します。

チャプター 19

TLuaSFTPClientFile

TLuaSFTPClientFile は読み取り専用のクラスです。

クラスメンバー

- string m_sFilename
- string m_sLongFilename
- TLuaSFTPClientAttributes m_Attribs

チャプター 20

TLuaSNMP

このクラスは、設定と取得の操作を実行できる基本的な SNMP クライアントを実装します。

この章で

サンプルスクリプト:TLuaSNMP

```
--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
```

BeginWalk

BeginWalk(string sOID)

パラメータ

- sOID - OID ツリースキャンの開始位置を表す OID。OID の例: iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable

注釈

Walk 関数を最初に呼び出す前に、プログラムは BeginWalk 関数を呼び出して Walk の開始位置を設定する必要があります。Walk は BeginWalk 関数によって設定される開始 OID の子アセットおよび兄弟アセットの識別子をすべて取得します。

閉じる

Close()

注釈

SNMP 接続を閉じます。

Get

string Get(string sOID,int iSyntax)

戻り値

リモート SNMP エージェントから取得した値をもつ文字列。

パラメータ

- sOID - 取得操作で使用する OID。インターフェースを照会するときには、インターフェースインデックスを指定するために@ユーザーを使うことができます。

@ユーザーの使用例

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

通常の OID の例

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

- iSyntax - 返されるデータのフォーマットを指定します。次の定数のいずれかを使用できます。
- SNMP_NOSYNTAX
- SNMP_IPADDRESS
- SNMP_INTEGER
- SNMP_UNSIGNED32
- SNMP_COUNTER32
- SNMP_GAUGE32
- SNMP_TIMETICKS
- SNMP_OPAQUE
- SNMP_OCTETSTRING
- SNMP_DATA_AS_HEXSTRING

バイナリ値の読み出し

一部の OID は、文字列や整数の代わりにバイナリデータを返します。Get 関数は null を終端とする文字列を返すので、これが問題になることがあります。この問題に対する解決策は、iSyntax 変数に SNMP_DATA_AS_HEXSTRING を設定することです。これにより、この関数は 16 進数にエンコードしたバイナリデータを返します。

16 進数でエンコードした 3 バイトの例

```
49 4E4D
```

開く

bool Open(string sCommunity, int iPort=161)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sCommunity - コミュニティの名前で、通常は"public"です。
- iPort（オプション） - 標準ポート（ポート 161)以外のポートを使用する必要がある場合は、ポート番号を指定します。

設定

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

戻り値

関数が正常に終了した場合は非ゼロの値を返し、それ以外の場合は 0 を返します。

パラメータ

- sOID - 設定操作で使用する OID。
- sData - 設定操作で使用するテキストデータ。
- iSyntax - sData パラメータのフォーマットを指定します。次の定数のいずれかを使用できます。
  - ✓ SNMP_NOSYNTAX
  - ✓ SNMP_IPADDRESS
  - ✓ SNMP_INTEGER
  - ✓ SNMP_UNSIGNED32
  - ✓ SNMP_COUNTER32
  - ✓ SNMP_GAUGE32
  - ✓ SNMP_TIMETICKS
  - ✓ SNMP_OPAQUE
  - ✓ SNMP_OCTETSTRING

Walk

TLuaSNMPResult TLuaSNMP::Walk()

戻り値

TLuaSNMPResult 構造内のメンバー変数 m_sOID に次の OID の識別子を返します。完全な OID 文字列は返しません。最後まで到達すると、TLuaSNMPResult 構造内の m_sOID メンバーは空になります。

注釈

Walk 関数を最初に呼び出す前に、プログラムは BeginWalk 関数を呼び出して Walk の開始位置を設定する必要があります。Walk は BeginWalk 関数によって設定される開始 OID の子およびオブジェクトの識別子をすべて取得します。

例

```
--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 == "";
```

TLuaSNMPResult

TLuaSNMPResult は、Walk 関数が返す読み取り専用のクラスです。修正すると例外が発生します。

クラスメンバー

- string m_sOID
- string m_sData
- int m_iSyntax

チャプター 21

TLuaSSH2Client

このクラスは、リモートサーバー上でコマンドを実行可能な SSH 2.0 クライアントを実装します。

この章で

サンプルスクリプト:TLuaSSH2Client

```
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
```

ExecuteCommand

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

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sCommand - リモートホスト上で実行するコマンドの文字列。
- dwWait - （オプション）実行が完了するまでの待機時間で、デフォルト値は 25 秒です。

GetErrorDescription

string GetErrorDescription(void)

戻り値

クライアントが生成した最後のエラー説明を文字列として返します。

GetStdErr

string GetStdErr(void)

戻り値

リモートホストからの標準エラー出力を返します。

GetStdOut

string GetStdOut(void)

戻り値

リモートホストからの標準出力を返します。

開く

bool Open(int iPort)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードは関数の GetErrorDescription を呼び出すことにより取得できます。コマンドの実行結果が失敗である場合、GetStdErr を呼び出すことで詳細情報を取得できます。

パラメータ

- iPort - SSH ポートで、デフォルト値は 23 です。
- sUsername - ユーザー名。
- sPassword - パスワード。

チャプター 22

TLuaSocket

このクラスは基本的なソケット操作を提供します。ソケットは、UDP モードまたは TCP モードで開くことができます。

この章で

サンプルスクリプト:TLuaSocket

```
--Construct a new socket device
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()
```

閉じる

int Close()

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

注釈

既に OpenTCP または OpenUDP を使用して開いたソケットを閉じます。

OpenTCP

int OpenTCP(int iPort)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- iPort - 開くポート。

注釈

指定したポート番号を使用して TCP ソケットを開きます。

OpenUDP

int OpenUDP(int iPort)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- iPort - ソケットで使用する特定のポート。

注釈

指定したポート番号を使用して UDP ソケットを開きます。

読み取り

string Read(int iSize,int iTimeout=1)

戻り値

関数が正常に実行された場合はデータの配列、データを読み出せない場合は nil を返します。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- iSize - 関数呼び出しが戻ると、この変数には読み出したデータの大きさが設定されます。データが何も読み出されなかった場合、この値はゼロになります。
- iTimeout - データがソケットに到着するまで待機する時間（単位: 秒）。デフォルト値は 1 秒です。

注釈

この関数はタイムアウト値で指定された時間、実行をブロックするだけです。この時間に何もデータを受け取らなかった場合、nil 値を返します。

書き込み

int Write(string sData,int iSize)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sData - 送信するデータの配列。
- iSize - 配列内のデータの大きさ。

チャプター 23

TLuaSocketSecure

このクラスは基本的なセキュアなソケット操作を提供します。これは一般的には TLS（Transport Layer Security）またはその前身の SSL（Secure Sockets Layer）と呼ばれます。

この章で

サンプルスクリプト:TLuaSocketSecure

```
--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 thescript 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 retrieved 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
```

開く

int Open(int iPort)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- iPort - 開くポート。

閉じる

int Close()

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

読み取り

string Read(int iSize)

戻り値

関数が正常に実行された場合はデータの配列、データを読み出せない場合は nil を返します。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- iSize - 関数呼び出しが戻ると、この変数には読み出したデータの大きさが設定されます。データが何も読み出されなかった場合、この値はゼロになります。

書き込み

int Write(string sData,int iSize)

戻り値

関数が正常に実行された場合は非ゼロ、その他の場合は 0 です。特定のエラーコードはグローバル関数の GetLastError を呼び出すことにより取得できます。

パラメータ

- sData - 送信するデータの配列。
- iSize - 配列内のデータの大きさ。

GetCertificateExpiryDate

TLuaDateTime GetCertificateExpiryDate()

戻り値

リモートホストの証明書の有効期限日を含む TLuaDateTime 構造。Connect()呼び出しが失敗した場合、この構造にはゼロの日付が入ります。

注釈

この関数を使用して、証明書が期限切れに近付いているか、既に期限切れとなっているかを調べることができます。

チャプター 24

TLuaStorage

このクラスは、スクリプトセッション間でテキストデータを保存する関数を提供します。これは現在のスクリプトの繰り返しを以前の結果に基づいて行う場合、または無関係の 2 つのスクリプト間でやりとりを行いたい場合に便利です。

この章で

CreateItem

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

戻り値

関数が正常に終了した場合は非ゼロの値を返し、それ以外の場合は 0 を返します。

パラメータ

- sName - 項目の一意の名前です。この名前が既に作成されている場合、この関数は失敗します。
- sKey - 項目のキー名で一意である必要があります。名前が既に存在している場合、この関数は失敗します。
- sData - 項目と関連付けられているオプションのデータ。
- iSize - データの大きさで、関数にデータが指定されている場合にのみ必須です。

注釈

この関数は項目と"キー"と呼ばれるサブ項目を作成し、ユーザーはデータとこのキーを関連付けることができます。データは後から FindItem 関数を呼び出すことで取得できます。

UpdateItem

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

戻り値

関数が正常に終了した場合は非ゼロの値を返し、それ以外の場合は 0 を返します。

パラメータ

- sName - 項目の一意の名前です。この名前をもつ項目が既に存在している必要があります。
- sKey - 項目のキー名です。その名前をもつキーが既に存在している必要があります。
- sData - 項目と関連付けられているオプションのデータで、このデータは項目内に格納されている現在のデータを置き換えます（存在する場合）。
- iSize - データの大きさで、関数にデータが指定されている場合にのみ必須です。

注釈

この関数は既に作成されている項目を更新します。項目/キーの組み合わせが存在しない場合、この関数は失敗します。

DeleteItem

void DeleteItem(string sName,string sKey);

パラメータ

- sName - 項目の名前。
- sKey - 削除するキー名。

注釈

この関数は項目/キーの組み合わせを削除し、キーと関連付けられているデータも削除します。

FindItem

TLuaStorageItem FindItem(string sName,string sKey);

戻り値

関数が正常に終了した場合は非ゼロの値を返し、それ以外の場合は 0 を返します。

パラメータ

- sName - 項目の一意の名前です。この名前をもつ項目が既に存在している必要があります。
- sKey - 項目のキー名です。その名前をもつキーが既に存在している必要があります。

注釈

この関数は格納されている項目を取得し、返されたクラスは項目/キー名だけでなく項目と関連付けられているデータも含みます。

TLuaStorageItem

TLuaStorageItem は読み取り専用のクラスです。修正すると例外が発生します。

クラスメンバー

- string m_Key
- string m_Name
- string m_pData
- int m_iSize

チャプター 25

TLuaTimer

このクラスは ms の精度を持つタイマーを提供します。

この章で

サンプルスクリプト:TLuaTimer

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

開始

Start()

注釈

この関数はタイマーを開始し、後続の Stop()関数の呼び出しにより、Start 関数の呼び出しから Stop 関数の呼び出しまでの時間を返します。Stop 関数を呼び出した後、Start 関数を再度呼び出すとタイマーがリセットされ、新しい期間を開始します。

停止する

int Stop()

戻り値

Start()関数の呼び出しからの時間（単位: ms）を返します。

チャプター 26

TLuaWinperf

このクラスは Windows パフォーマンスレジスタの数値を照会する関数を提供します。これは、より高度な TLuaWMIQuery クラスの代わりとなる使いやすい選択肢として提供されています。このクラスは、スクリプトを起動したプロセスまたはスレッドのセキュリティコンテキストで動作します。IDE のセキュリティコンテキストはデスクトップから継承されます。Lus スクリプトモニターによって実行されるときには、モニターのプロパティのページでデフォルトアカウントを選択することにより、セキュリティコンテキストを設定できます。

この章で

サンプルスクリプト:TLuaWinperf

```
--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
```

GetErrorDescription

string GetErrorDescription()

戻り値

クラス内の任意の関数を呼び出すときに、最後に発生したエラーを説明する文字列を返します。

GetResult

double GetResult()

戻り値

カウンターの数値を返します。以前の Query()呼び出しが失敗している場合、この関数は 0 を返します。

Query

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

戻り値

クエリーが正常に実行された場合は true、エラーが発生した場合は false です。

パラメータ

- sDeviceName - 照会するカウンターを含むアセットの名前をもつ文字列。
- sCounterName - 照会するカウンターの名前をもつ文字列。
- sInstanceName - （オプション）カウンターインスタンスの名前をもつ文字列。

注釈

アセット、カウンター、およびインスタンスの名前は、[列挙]ボタンをクリックすることで **Network Monitor** の Winperf モニターから得られるか、Windows の perfmon.exe アプリケーションを使用することで得られます。値を取得するには、この関数の完了後に GetResult()を呼び出します。

チャプター 27

TLuaWMIQuery

このクラスは WMI のプロパティを照会する関数を提供します。このクラスは、スクリプトを起動したプロセスまたはスレッドのセキュリティコンテキストで動作します。IDE のセキュリティコンテキストはデスクトップから継承されます。Lus スクリプトモニターによって実行されるときには、モニターのプロパティのページでデフォルトアカウントを選択することにより、セキュリティコンテキストを設定できます。このアカウントの委任を有効にしておく必要があります。

この章で

TLuaWMIQuery

```
--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
```

実行する

bool Execute(const char *pWQL)

bool Execute(const char *pWQL,const int iPrivacy)

戻り値

クエリーが正常に実行された場合は true、エラーが発生した場合は false です。

パラメータ

- sWQL - WQL クエリーを含む文字列。

注釈

WQL（WMI クエリー言語: WMI Query Language）クエリーを実行します。Next()および GetProperty()の呼び出しを使用して、結果を取得できます。

GetErrorDescription

string GetErrorDescription()

戻り値

クラス内の任意の関数を呼び出すときに、最後に発生したエラーを説明する文字列を返します。WMI クエリーをデバッグするときに便利です。

GetProperty

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

戻り値

正常に実行された場合は true および文字列の値を返し、関数が失敗した場合は false および空の文字列を返します。エラーに関する詳細情報は、GetError()を呼び出すことで取得できます。

パラメータ

- sPropertyName - 取得するプロパティの名前。
- sReturnValue - 戻り値を受け取るように定義された文字列。プロパティの型が整数や実数の場合でも、戻り値は常に文字列です。

注釈

現在の結果のプロパティ値を取得します。同じプロパティの次の値を取得するには、NextInstance()関数を呼び出します。NextInstance()が false を返した場合、もう値はありません。

NextInstance

bool NextInstance()

戻り値

新しい結果が取得された場合は true、クエリーの結果がこれ以上存在しない場合は false。

注釈

この関数は、以前の Execute 関数の呼び出しで生成された新しい結果を取得します。この関数は、GetProperty 関数の最初の呼び出しより前に呼び出す必要があります。

SetNamespace

SetNamespace(string sNamespace)

パラメータ

- sNamespace - 後続のすべての呼び出しで使用する WMI ネームスペースをもつ文字列。

注釈

TLuaWMIQuery クラスが使用するデフォルトのネームスペースは root\cvmi2 です。

チャプター　28

TLuaXMLNode

このクラスは XML 要素を表し、子要素を 1 つ以上もつことができます。

この章で

FindAttribute

string FindAttribute(string sName)

戻り値

この関数は、属性の値をもつ文字列を返します。属性が見つからない場合、空の文字列が返されます。

パラメータ

- sName - 属性の名前。

FindChildNode

TLuaXMLNode FindChildNode(string sElementName, int iOffset)

戻り値

要素が見つかった場合、この関数は有効な TLuaXMLNode アセットを返します。

パラメータ

- sElementName - このノードの子である要素の名前。
- iOffset - ノード内で同じ名前をもつ子要素を取得するためのゼロから始まるインデックス。

注釈

この関数は、同じ名前をもつ複数の子要素に対して繰り返し使用できます。次の要素を取得するには、オフセットパラメータを増加します。

GetData

string GetData()

戻り値

この関数は要素内のデータを返します。

GetTag

string GetTag()

戻り値

この関数は、要素のタグ名を返します。

GetParentNode

TLuaXMLNode GetParentNode()

戻り値

この関数は、現在の XML ドキュメント要素の親を返します。

IsValid

bool IsValid()

戻り値

この関数は、ノードが有効の場合に true、ノードが無効の場合に false を返します。

注釈

すべての検索関数は TLuaXMLNode アセットを返します。IsValid()関数は、検索が正常に実行されたかどうかを判定するために使用します。

チャプター 29

TLuaXMLReader

このクラスは、XML ドキュメントの構文解析とスキャンを行う基本機能を提供します。

この章で

FindChildNode

TLuaXMLNode FindChildNode(string sElementName, TLuaXMLNode ParentNode)

戻り値

要素が見つかった場合、この関数は有効な TLuaXMLNode アセットを返します。

パラメータ

- sElementName - ParentNode の子である要素の名前。
- ParentNode - 検索対象の親ノード。

注釈

この関数は指定した名前をもつ最初の要素を返すことに注意してください。

FindNode

TLuaXMLNode FindNode(string sElementName, TLuaXMLNode RootNode)

戻り値

要素が見つかった場合、この関数は有効な TLuaXMLNode アセットを返します。

パラメータ

- sElementName - ParentNode の子である要素の名前。
- RootNode - 検索の開始位置となる親ノード。

注釈

この関数は、検索の開始位置として"RootNode"をもつ XML ドキュメントを再帰的に検索します。

FromXML

bool FromXML(string XML)

戻り値

操作が正常に実行された場合は true、エラーが発生した場合は false です。

パラメータ

- sXML - 構文解析を行う XML ドキュメント。

注釈

構文解析ではドキュメントのスキーマを検証しないことに注意してください。

GetRootNode

TLuaXMLNode GetRootNode()

戻り値

この関数は、XML ドキュメントのルート要素を返します。

インデックス

N

O

P

Q

R

S

T

U

W

あ

か

さ

は

漢字