Kaseya 2

Network Monitor API

Benutzerhandbuch

Versión 7.0

Deutsch

September 15, 2014

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."

Inhalt

Network Monitor Lua-API

Diese Dokumentation behandelt die **Network Monitor** Lua-API. **Network Monitor**verwendet Lua 5.0.

Lua

Lua ist eine leistungsfähige, schlanke Programmiersprache, die zur Erweiterung von Anwendungen entwickelt wurde. Lua wird außerdem häufig als eigenständige Mehrzwecksprache verwendet. Lua ist eine frei erhältliche Software. Lua verbindet einfache prozedurale Syntax mit leistungsfähigen Datenbeschreibungskonstrukten, die auf assoziative Arrays und erweiterbare Semantik basieren. Lua ist dynamisch typisiert, wird von Bytecode übersetzt und verfügt über automatische Speicherverwaltung mit automatischer Speicherbereinigung, wodurch sie ideal für Konfiguration, Skripting und Rapid Prototyping ist.

Hinweis: Diese Dokumentation behandelt nicht die Programmiersprache Lua. Weitere Informationen zur Programmiersprache Lua finden Sie unter **http://www.lua.org.** *(http://www.lua.org.)*

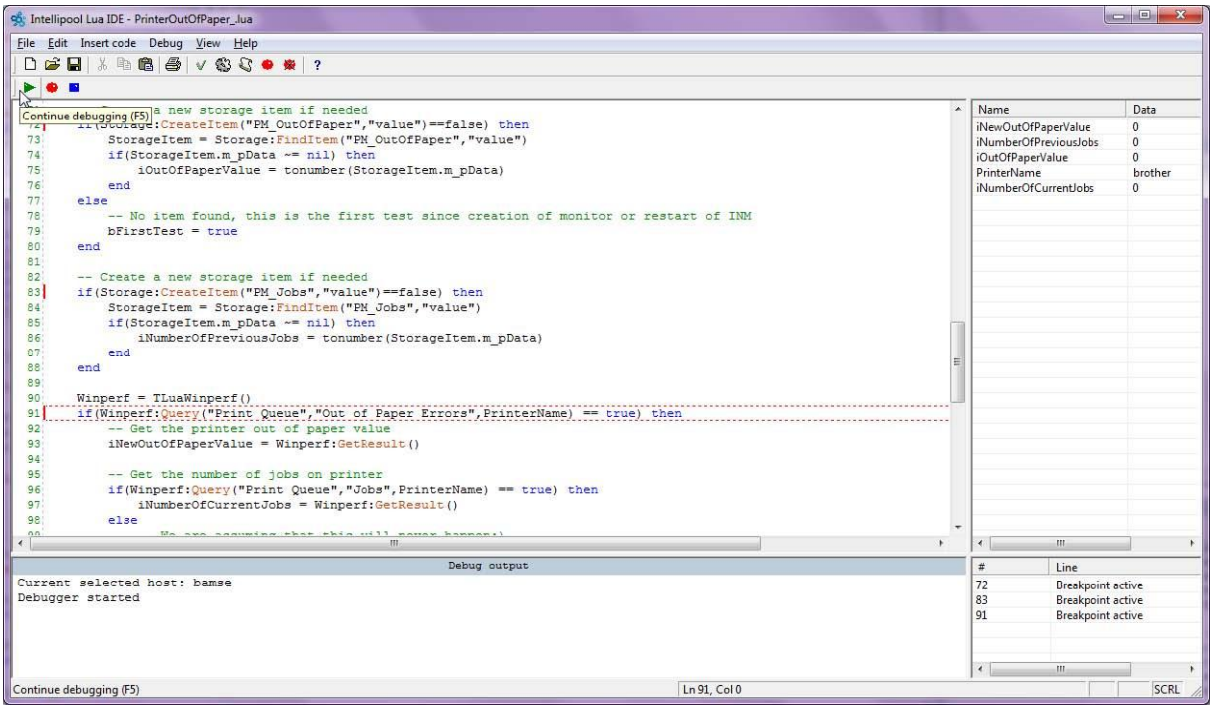
Network Monitor und Lua

Kunden können mit der Skriptsprache Lua benutzerdefinierte Monitore erstellen, um Systeme und Arbeitsgeräte zu testen, die von der aktuell verwendeten Überwachungslösung nicht unterstützt werden.

Neue Monitore, Aktionen und Ereignisse können in der von Kaseya zur Verfügung gestellten Entwicklungsumgebung erstellt und getestet werden, bevor sie in Kaseya Network Monitor exportiert und verwendet werden.

Den Entwicklern steht eine umfassende Bibliothek vordefinierter Klassen, wie z. B. SFTP-Client, HTTP-Client und Dateiverwaltung zur Verfügung.

Die Entwicklungsumgebung beinhaltet Debugger, Hervorhebung von Schlüsselwörtern, eine integrierte Hilfe und andere Funktionen, die in modernen Entwicklungstools verfügbar sind. Die Entwicklungsumgebung kann von unserer Homepage heruntergeladen werden: **http://www.kaseya.com** *(http://www.kaseya.com)*



Lua IDE v3

Kapitel 1

Programmiermodell

Wenn ein benutzerdefiniertes Lua-Skript zur Verwendung in Kaseya Network Monitor erstellt wird, gibt es mehrere Anforderungen, die das Skript erfüllen muss, um erfolgreich von Kaseya Network Monitor ausgeführt werden zu können.

In diesem Kapitel

Erweitertes Skript

Das erweiterte Skriptmodell stellt dem Skriptautor neue leistungsstarke Tools für die Steuerung von Parametern zur Verfügung, die als Argumente im Skript eingefügt werden. Dadurch können Lua-Skripte mit demselben Aussehen und Verhalten wie systemeigene Monitortypen erstellt werden.

Reservierte Funktionsnamen

Es gibt zwei reservierte Funktionsnamen, die von **Network Monitor** verwendet werden, um Informationen abzufragen. Diese Funktionsnamen dürfen für keinen anderen Zweck verwendet werden.

OnConfigure

Diese Funktion wird von **Network Monitor** aufgerufen, damit das Skript eine LuaScriptConfigurator-Klasseninstanz auffüllt. Diese Daten werden dann dafür verwendet, um eine Benutzeroberfläche für das Skript zu erstellen. Der Instanzname muss "Config" (Schreibweise beachten) lauten, sodass **Network Monitor** diese im Lua-Stapel finden kann, wenn die Funktion zurückgegeben wird.

OnEnumerate

Es kann jedes Feld in der Benutzeroberfläche aufgezählt werden, **Network Monitor** ruft die OnEnumerate-Funktion auf, damit das Skript mit einer Datenstruktur, LuaScriptEnumResult, aufgefüllt wird und zwar mit Werten, die der Benutzer auswählen kann. Die Funktion "OnEnumerate" hat einen Parameter, sFieldToEnum, der vom Skript verwendet wird, um zu bestimmen, für welches Feld/Argument Aufzählungsergebnisse geliefert werden sollen. Die zurückgegebene Instanz muss "Enum" genannt werden (Schreibweise beachten).

Der Einstiegspunkt

Das erweiterte Skriptmodell fordert von der Funktion "OnConfigure", dass diese den Namen der Einstiegspunktfunktion festlegt. Diese Funktion wird von **Network Monitor** aufgerufen, damit diese die Ausführung des Skripts startet. Der Einstiegspunktname ist standardmäßig "main", kann jedoch vom Programmierer beliebig geändert werden; die reservierten Funktionsnamen sind davon ausgenommen.

Beispiel

```
--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
```

Einfaches Skript

Das einfache Skriptmodell wurde in **Network Monitor** von der ersten Version an verwendet, und gilt nun als veraltet. Es wird aus Kompatibilitätsgründen mit älteren Skripten beibehalten.

Bestandskontext

Funktionen sind relativ zum Bestandskontext.

Alle Aufrufe bei denen auf Ressourcen zugegriffen wird, sind relativ zum übergeordneten Bestand. Zum Beispiel, wenn das Skript einen Pfad öffnet, muss der bereitgestellte Pfad zur Öffnungsfunktion relativ zum Bestand sein.

Beispiel

Richten Sie den Host als Adresse für den "Domainserver" des Windows-Computer ein.

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

Durch Aufrufen der Funktion öffnet das Skript die Datei test.txt, die sich auf der C-Festplatte des "Domainserver"-Computers befindet.

Aus diesem Grund nehmen alle der Kommunikation zugehörige Klassen wie z. B. TLuaFTPClient, TLuaHTTPClient und TLUASocket nur ein Port-Nummernargument auf, die IP-Adresse ist über Framework mit dem aktuellen Bestand hartcodiert.

Ergebnis

Wenn ein Skript beendet wird, muss es **Network Monitor** melden, ob der Test erfolgreich war oder nicht. Eine globale Funktion steht für diesen Zweck bereit: SetExitStatus. SetExitStatus ist obligatorisch und muss aufgerufen werden, bevor das Skript beendet wird.

Kapitel 2

Globale Funktionen

Globale Funktionen sind Funktionen, die nicht mit einem Bestand verknüpft sind. Es gibt mehrere globale Funktionen in der **Network Monitor** Lua-API. Ein paar müssen aufrufen, wenn ein Skript beendet wird.

In diesem Kapitel

ConvertFromUTF16

string ConvertFromUTF16(local UTF16data,int iSize)

Rückgabewerte

Eine 8-Bit-Zeichenfolge, die von einem UTF16-Zeichenfolge umgewandelt wurde.

Parameter

- UTF16data – UTF16-Zeichenfolge (Doppelbyte) gelesen von TLuaFile::ReadData.
- iSize – Größe der Zeichenfolge.

Hinweise

Diese Funktion akzeptiert nur Daten, die von der Funktion TLuaFile::ReadData erstellt wurden.

FormatErrorString

string FormatErrorString(int iError)

Rückgabewerte

Eine Zeichenfolge, die den Fehlercode "iError" beschreibt.

Parameter

- iError – Ein Windows-Fehlercode, der bereits durch den Aufruf der Funktion GetLastError erhalten wird.

Hinweise

Diese Funktion kann sinnvoll sein, um dem Benutzer aussagekräftigem Text statt eines Fehlercodes zu senden.

GetArgument

string GetArgument(int iNumber)

Rückgabewerte

Ein Argument, dass durch den Aufruf der Anwendung übergeben wird.

Parameter

- iNumber – Ein nullbasierter Index des abzurufenden Arguments. Die Höchstzahl der Argumente kann durch den Aufruf von GetArgumentCount festgelegt werden.

Hinweise

Eine aufrufende Anwendung kann eine Anzahl an Argumente an ein Lua-Skript übergeben, um dessen Verhalten anzupassen. Mit dieser Funktion und dem verwandten GetArgumentCount kann der Programmierer die Argumente extrahieren.

GetArgumentCount

int GetArgumentCount()

Rückgabewerte

Die Anzahl der Argumente, die von einer aufrufenden Anwendung an ein Programm übergeben werden.

Hinweise

Eine aufrufende Anwendung kann eine Anzahl an Argumente an ein Lua-Skript übergeben, um dessen Verhalten anzupassen. Mit dieser Funktion kann der Programmierer bestimmen, wie viele Argumente extrahiert werden sollen.

GetLastError

int GetLastError()

Rückgabewerte

Der letzte Fehlercode, der durch den Aufruf einer Bibliothek-Funktion erzeugt wurde. Der Fehlercode ist ein standardmäßiger Windows-Fehlercode.

Hinweise

Mit SetLastError kann der aktuelle Windows-Fehlercode gelöscht werden, bevor function.sds aufgerufen wird

GetDeviceAddress

string GetDeviceAddress()

Rückgabewerte

Die in das Adressenfeld des Geräts eingegebene Adresse.

Hinweise

Die Zeichenfolge kann als eindeutiger Bezeichner verwendet werden, wenn Daten in TLuaStorage gespeichert werden.

IsIDE

bool IsIDE()

Rückgabewerte

Der boolesche Wert muss "Wahr" sein, wenn das Skript von IDE ausgeführt wird, und "Falsch", wenn das Skript von **Network Monitor** ausgeführt wird.

Hinweise

Diese Funktionen können verwendet werden, wenn das Skript von **Network Monitor** oder IDE ausgeführt wird.

MessageBox

MessageBox(string sText)

Parameter

- sText – Der im Nachrichtenfeld erscheinende Text.

Hinweise

Diese Funktionen rufen ein standardmäßiges Nachrichtenfeld des Betriebssystems auf, um eine Zeichenfolge anzuzeigen. Diese Funktion ist nur in IDE verfügbar. Beachten Sie, dass die Ausführung des Skripts gestoppt wird, solange das Nachrichtenfeld geöffnet ist.

print

print(string sText)

Parameter

- sText – Text, der im Ausgabefenster erscheint

Hinweise

Diese Funktion kann verwendet werden, um Text im Ausgabefenster für Debug-Zwecke auszugeben. Wenn das Skript von **Network Monitor** ausgeführt wird, hat der mit dieser Funktion ausgegebene Text keinen Zweck.

SetExitStatus

SetExitStatus(string sString,bool bSuccess)

Parameter

- sString – Eine Zeichenfolge, die das Ergebnis des Skripts beschreibt.
- bSuccess – Das Skript gilt bei ungleich null (boolescher Wert ist "Wahr") als erfolgreich vom Framework ausgeführt. Wenn dieser Wert auf null (boolescher Wert ist "Falsch") steht, muss die Funktion SetErrorString ebenfalls mit einer Zeichenfolge aufgerufen werden, die den Fehlerstatus beschreibt.

Hinweise

Diese Funktion muss beim Beenden eines Skripts aufgerufen werden. Die Funktion teilt **Network Monitor** mit, ob das Skript erfolgreich war oder nicht. Der Text, der mit der Funktion bereitgestellt wird, wird von **Network Monitor** verwendet, um in der Benutzeroberfläche den letzten Statustext einzusetzen, wenn das Skript im Kontext eines Agents ausgeführt wird.

SetLastError

SetLastError(int iErrorCode)

Parameter

- iErrorCode – Eine ganzzahlige Entsprechung für einen Windows-spezifischen Fehlercode.

Hinweise

Die Funktion richtet den letzten Fehlercode ein, der später über GetLastError abgerufen werden kann.

StoreStatisticalData

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

Rückgabewerte

Wahr: Daten konnten erfolgreich in der statistischen Datenbank gespeichert werden. Falsch: Ein Parameterfehler ist vorgefallen.

Parameter

- iRecordSetIndex – Ein nullbasierter Index des statistischen Kanals, in dem die Daten gespeichert werden. Gültige Konstanten finden Sie unter Hinweise.
- fData – Gleitkommadaten, die vom Skript aufgezeichnet werden.
- fThreshold – Optionaler Schwellenwert für die Probedaten. Dieser Wert muss in allen Aufrufen konstant sein.
- Unit – Optionale Zeichenfolge, die die Dateneinheit beschreibt. Dieser Wert muss in allen Aufrufen konstant sein. Diese Zeichenfolge darf höchstens 16 Zeichen lang sein, sonst schlägt der Aufruf fehl.

Hinweise

Diese Funktion gibt dem Skript die Fähigkeit, statistische Daten zu speichern. Aktuell gibt es 8 Kanäle, die für diesen Zweck verwendet werden können. Der Indexparameter der Datensatzmenge kann eine der folgenden Konstanten sein.

```
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)

Rückgabewerte

Wahr: Daten konnten erfolgreich in der statistischen Datenbank gespeichert werden. Falsch: Ein Parameterfehler ist vorgefallen.

Parameter

- iRecordSetIndex – Ein nullbasierter Index des statistischen Kanals, in dem die Daten gespeichert werden. Gültige Konstanten finden Sie unter Hinweise.
- fData – Gleitkommadaten, die vom Skript aufgezeichnet werden.
- fThreshold – Optionaler Schwellenwert für die Probedaten. Dieser Wert muss in allen Aufrufen konstant sein.
- iVirtualType – Datentyp der gespeicherten Daten.

- iVirtualUnit – Ausgewählte Einheit des gespeicherten Typs. Gültige Kombinationen von Typen und Einheiten finden Sie unter Hinweise.
- Unit – Optionale Zeichenfolge, die die Dateneinheit beschreibt. Dieser Wert muss in allen Aufrufen konstant sein. Diese Zeichenfolge darf höchstens 16 Zeichen lang sein, sonst schlägt der Aufruf fehl.

Hinweise

Diese Funktion ist nur für erweiterte Skripte verfügbar. Der Unterschied zwischen dieser Funktion und der alten Funktion mit demselben Namen ist die Fähigkeit Typinformationen mit den Daten zu speichern.

iVirtualType und iVirtualUnit kann in den folgenden Kombinationen verwendet werden:

```
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
```

Der Indexparameter der Datensatzmenge kann eine der folgenden Konstanten sein.

```
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)

Parameter

- iMs – Die Anzahl der Millisekunden, die die Skriptausführung warten soll.

Hinweise

Diese Funktion ruft die Betriebssystem-Funktion "Sleep" dazu auf, die Ausführung des Thread abzubrechen, der das Skript ausführt.

Kapitel 3

LuaScriptEnumResult

Die Klasse bietet eine Benutzeroberfläche, um Aufzählungsergebnisse in der OnEnumeration-Rückruffunktion einzugeben.

In diesem Kapitel

Beispiel-Skript: 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
```

Add

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

Parameter

- sDisplayValue – Anzeigewert als auswählbare Option.
- sUsageValue – (Optional) Ein Wert der statt dem Anzeigewert verwendet wird.

Hinweise

Der optionale Parameter sUsageValue kann verwendet werden, wenn Sie sehr komplexe und lange Werte haben und einen einfacheren Weg suchen, die Optionen anzuzeigen. Wenn Sie dies verwenden, wird sDisplayValue dem Benutzer als Wert angezeigt, **Network Monitor** verwendet jedoch den Wert sUsageValue.

Kapitel 4

LuaScriptConfigurator

Diese Klasse bietet eine Benutzeroberfläche, um Konfigurationsinformationen zu erstellen, die **Network Monitor** verwendet, um für das Skript eine Benutzeroberfläche zu präsentieren.

In diesem Kapitel

Beispiel-Skript: 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);

Rückgabewerte

Ein Handle, das verwendet werden kann, um auf dieses Argument in nachfolgenden Aufrufen zu verweisen.

Parameter

- sName – Name des Argumentfelds
- sDesc – Beschreibung des Felds für die Validierung von Markierungssteuerungen "iFlags". Markierungen finden Sie unter Hinweise.

Hinweise

Es handelt sich um gültige Markierungen. Manche können kombiniert werden.

CHECK_NOTHING	Standardwert, jedes Zeichen, einschließlich kein Text wird akzeptiert.
CHECK_NOT_EMPTY	Überprüfen, ob das Argument leer ist. Kann nicht mit CHECK_NOTHING kombiniert werden.
CHECK_RANGE_LOW	Muss zusammen mit CHECK_NUMERIC verwendet werden. Überprüft, dass der numerische Wert im Bereich liegt (unterer Bereich)
CHECK_RANGE_HIGH	Muss zusammen mit CHECK_NUMERIC verwendet werden. Überprüft, dass der numerische Wert im Bereich liegt (oberer Bereich)
CHECK_NUMERIC	Überprüft, dass der Wert numerisch ist (reelle oder ganze Zahl)
ENUM_AVAIL	Zeigt an, dass es eine Aufzählungsrückruffunktion mit vordefinierten Werten gibt, die für dieses Feld verfügbar ist.

SetCharacterLimits

SetCharacterLimits(int iArgIndex,int iMaxCharacters,int iMaxVisibleCharacters)

Parameter

- iArgIndex – Handle wird von AddArgument zurückgegeben.
- iMaxCharacters – Max. Eingabezeichen für Argument.
- iMaxVisibleCharacters – Max. sichtbare Zeichen, muss kleiner oder gleiche iMaxCharacters sein.

Hinweise

Diese Funktion legt die Höchstlänge eines Arguments fest und wie viele dieser Zeichen in der Benutzeroberfläche sichtbar sind (Länge des Eingabefelds).

SetNumericLimits

SetNumericLimits(int iArgIndex,float fLow,float fHigh)

Parameter

- iArgIndex – Handle wird von **AddArgument zurückgegeben.** *(siehe 18)*
- Niedrig – Unterer Bereich
- Hoch – Oberer Bereich

Hinweise

Diese Funktion legt den gültigen Bereich für reelle und Ganzzahl-Werte fest, die in das Feld eingetragen werden dürfen. Für das Argument müssen die Markierungen CHECK_RANGE_LOW und CHECK_RANGE_HIGH gesetzt sein.

SetEntryPoint

SetEntryPoint(string sName)

Parameter

- sName – Name der Einstiegspunktfunktion.

Hinweise

Die Funktion registriert den Namen der Einstiegspunktfunktion. Diese Funktion ruft **Network Monitor** als Einstiegspunkt der Ausführung auf. Der Standardwert lautet "main".

SetAuthor

SetAuthor(string sName)

Parameter

- sName – Name des Skriptautors.

Hinweise

Diese Funktion wird vom Autor des Skripts festgelegt. Es wird für Beschreibungszwecke verwendet, wenn ein Benutzer ein Drittanbieterskript lädt, um zu informieren, wer das Skript geschrieben hat.

SetDescription

SetDescription(string sDescription)

Parameter

- sDescription – Eine Beschreibung der Funktion des Skripts.

Hinweise

Die Beschreibung eines Skripts muss dem Benutzer in wenigen Zeilen die Funktion des Skripts mitteilen, und ob Einschränkungen bekannt sind. Es gibt keine Längenbeschränkung für den Text, aber er sollte kurz gehalten werden.

SetMinBuildVersion

SetMinBuildVersion(int iMinBuildNumber)

Parameter

- iMinBuildNumber – Die minimale Buildnummer von **Network Monitor**, die das Skript anfordert.

Hinweise

Die minimale Buildnummer ist ein sehr wichtiges Feld, das eingestellt werden muss. Es teilt **Network Monitor** mit, ob das Skript mit der aktuelle Version von **Network Monitor** verwendet werden kann. Standardmäßig muss diese Nummer auf die Buildnummer eingestellt werden, die der Autor zur Skriptprüfung verwendet hat.

SetScriptVersion

SetScriptVersion(int iMajor,int iMinor)

Parameter

- iMajor – Die Hauptversionsnummer des Skripts.
- iMinor – Die Nebenversionsnummer des Skripts.

Hinweise

Der Skriptautor muss eine Versionsnummer des Skripts festlegen. Eine Hauptversion 0 zeigt an, dass das Skript in einer "Betaphase" ist und nur für die Weiterentwicklung von anderen Benutzern verwendet werden darf. Mit jeder Skriptänderung muss die Versionsnummer erhöht werden. Eine Änderung der Hauptversionsnummer muss eine deutliche Verbesserung oder neu Schreibung des Skripts deutlich machen, die Nebenversionsnummer zeigt eine unbedeutendere Veränderung an.

Kapitel 5

TLuaDateTime

TLuaDateTime stellt Datum- und Zeitfunktionen zur Verfügung. Bei der Zeit handelt es sich um die Ortszeit in Sekunden vom 1. Januar 1970 an.

In diesem Kapitel

Add

Add(TLuaDateTime DateTime)

Parameter

- DateTime – TLuaDateTime-Instanz wird von anderen Klassenfunktionen erhalten oder konstruiert.

Hinweise

Die Funktion fügt die Zeit, die im Parameter DateTime enthalten ist, dem Bestand hinzu.

Create

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

Parameter

- iYear – Jahr, z. B. 1972
- iMonth – Monat, z. B. 10
- iDay – Tag, z. B. 2
- iHour – Festgelegte Stunde, der Wert kann null sein
- iMinute – Festgelegte Minute, der Wert kann null sein
- iSecond – Festgelegte Sekunde, der Wert kann null sein

Hinweise

Die Funktion erstellt TLuaDateTime mit einer absoluten Zeit.

CreateSpan

CreateSpan(int iHour,int iMinute,int iSecond)

Parameter

- iHour – Festgelegte Stunden, der Wert kann null sein
- iMinute – Festgelegte Minuten, der Wert kann null sein
- iSecond – Festgelegte Sekunden, der Wert kann null sein

Hinweise

Die Funktion erstellt TLuaDateTime nicht mit einer absoluten Zeit, sondern mit einer Zeitspanne, die zum Hinzufügen oder Abziehen von einem anderen TLuaDateTime-Bestand verwendet werden kann.

Equal

bool Equal(TLuaDateTime DateTime)

Rückgabewerte

Wahr, wenn DateTime gleich ist, andernfalls Falsch.

Parameter

- DateTime – TLuaDateTime-Instanz wird von anderen Klassenfunktionen erhalten oder konstruiert.

Get

int Get()

Rückgabewerte

Anzahl der Sekunden, die in dieser Instanz enthalten sind.

Hinweise

Die Funktion kann verwendet werden, um die Anzahl der Sekunden, die in dieser Instanz enthalten sind, in absoluter Zeit abzurufen.

GetDate

string GetDate(string sFormat=NULL)

Rückgabewerte

Gibt eine Zeichenfolge mit der aktuellen Zeit zurück, der im Standardformat oder wie im Parameter sFormat angegeben formatiert ist.

Parameter

- sFormat – Optionale Zeichenfolge mit einem alternativen Format der zurückgegebenen Zeit. Das Standardformat lautet JJ–MM–TT. Die verfügbaren Markierungen finden Sie im Abschnitt ''Hinweise''.

Hinweise

Gibt eine Zeichenfolge mit der in der Instanz enthaltenen Zeit zurück. Das Standardformat lautet JJ–MM–TT. Mit Ihrem eigenen Formatcode können Sie die Art ändern, in der die Zeit zurückgegeben wird.

Markierungen formatieren

- %a – Abgekürzter Wochentagsname
- %A – Vollständiger Wochentagsname
- %b – Abgekürzter Monatsname
- %B – Vollständiger Monatsname
- %c – Angemessene Anzeige von Datum und Zeit für das Gebietsschema
- %d – Monatstag als Dezimalzahl (01 – 31)
- %H – Stunde im 24-Stundenformat (00 – 23)
- %I – Stunde im 12-Stundenformat (01 – 12)
- %j – Tag im Jahr als Dezimalzahl (001 – 366)
- %m – Monat als Dezimalzahl (01 – 12)
- %M – Minute als Dezimalzahl (00 – 59)
- %p – Für das Gebietsschema aktuelle Anzeige der Indikatoren ''A.M./P.M.'' für Uhren im 12-Stundenformat
- %S – Sekunden als Dezimalzahl (00 – 59)
- %U – Kalenderwoche als Dezimalzahl, wobei Sonntag der erste Tag der Woche ist (00 – 53)

- %w – Wochentag als Dezimalzahl (0 – 6; Sonntag ist 0)
- %W – Kalenderwoche als Dezimalzahl, wobei Montag der erste Tag der Woche ist (00 – 53)
- %x – Datumsdarstellung für aktuelles Gebietsschema
- %X – Zeitdarstellung für aktuelles Gebietsschema
- %y – Jahr ohne Jahrhundertangabe, als Dezimalzahl (00 – 99)
- %Y – Jahr mit Jahrhundertangabe, als Dezimalzahl
- %z, %Z – Name oder Abkürzung der Zeitzone; bei unbekannter Zeitzone keine Zeichen

GetTime

string GetTime(string sFormat=NULL)

Rückgabewerte

Gibt eine Zeichenfolge mit der aktuellen Zeit zurück, die im Standardformat oder wie im Parameter sFormat angegeben formatiert ist.

Parameter

- sFormat – Optionale Zeichenfolge mit einem alternativen Format der zurückgegebenen Zeit. Das Standardformat lautet HH:MM:SS. Die verfügbaren Markierungen finden Sie im Abschnitt ''Hinweise''.

Hinweise

Gibt eine Zeichenfolge mit der in der Instanz enthaltenen Zeit zurück. Das Standardformat lautet HH:MM:SS. Mit Ihrem eigenen Formatcode können Sie die Art ändern, in der die Zeit zurückgegeben wird.

Markierungen formatieren

- %a – Abgekürzter Wochentagsname
- %A – Vollständiger Wochentagsname
- %b – Abgekürzter Monatsname
- %B – Vollständiger Monatsname
- %c – Angemessene Anzeige von Datum und Zeit für das Gebietsschema
- %d – Monatstag als Dezimalzahl (01 – 31)
- %H – Stunde im 24-Stundenformat (00 – 23)
- %I – Stunde im 12-Stundenformat (01 – 12)
- %j – Tag im Jahr als Dezimalzahl (001 – 366)
- %m – Monat als Dezimalzahl (01 – 12)
- %M – Minute als Dezimalzahl (00 – 59)
- %p – Für das Gebietsschema aktuelle Anzeige der Indikatoren ''A.M./P.M.'' für Uhren im 12-Stundenformat
- %S – Sekunden als Dezimalzahl (00 – 59)
- %U – Kalenderwoche als Dezimalzahl, wobei Sonntag der erste Tag der Woche ist (00 – 53)
- %w – Wochentag als Dezimalzahl (0 – 6; Sonntag ist 0)
- %W – Kalenderwoche als Dezimalzahl, wobei Montag der erste Tag der Woche ist (00 – 53)
- %x – Datumsdarstellung für aktuelles Gebietsschema
- %X – Zeitdarstellung für aktuelles Gebietsschema
- %y – Jahr ohne Jahrhundertangabe, als Dezimalzahl (00 – 99)
- %Y – Jahr mit Jahrhundertangabe, als Dezimalzahl
- %z, %Z – Name oder Abkürzung der Zeitzone; bei unbekannter Zeitzone keine Zeichen

Greater

bool Greater(TLuaDateTime DateTime)

Rückgabewerte

Wahr, wenn DateTime kleiner ist, andernfalls Falsch.

Parameter

- DateTime – TLuaDateTime-Instanz wird von anderen Klassenfunktionen erhalten oder konstruiert.

GreaterOrEqual

bool GreaterOrEqual(TLuaDateTime DateTime)

Rückgabewerte

Wahr, wenn DateTime kleiner oder gleich ist, andernfalls Falsch.

Parameter

- DateTime – TLuaDateTime-Instanz wird von anderen Klassenfunktionen erhalten oder konstruiert.

Less

bool Less(TLuaDateTime DateTime)

Rückgabewerte

Wahr, wenn DateTime größer ist, andernfalls Falsch.

Parameter

- DateTime – TLuaDateTime-Instanz wird von anderen Klassenfunktionen erhalten oder konstruiert.

LessOrEqual

bool LessOrEqual(TLuaDateTime DateTime)

Rückgabewerte

Wahr, wenn DateTime größer oder gleich ist, andernfalls Falsch.

Parameter

- DateTime – TLuaDateTime-Instanz wird von anderen Klassenfunktionen erhalten oder konstruiert.

NotEqual

bool NotEqual(TLuaDateTime DateTime)

Rückgabewerte

Wahr, wenn DateTime nicht gleich ist, andernfalls Falsch.

Parameter

- DateTime – TLuaDateTime-Instanz wird von anderen Klassenfunktionen erhalten oder konstruiert.

Set

Set(int iNewTime)

Parameter

- iNew – Zeitoffset in Sekunden oder absolute Zeit in Sekunden ab 1. Januar 1970

Hinweise

Diese Funktion kann verwendet werden, um eine TLuaDateTime-Instanz zu erstellen, die später hinzugefügt, subtrahiert oder mit einer anderen TLuaDateTime-Instanz verglichen werden wird.

Sub

Sub(TLuaDateTime DateTime)

Parameter

- DateTime – TLuaDateTime-Instanz wird von anderen Klassenfunktionen erhalten oder konstruiert.

Hinweise

Die Funktion subtrahiert die Zeit, die im Parameter DateTime enthalten ist, zuzüglich zur Zeit, die im Bestand gespeichert ist.

Kapitel 6

TLuaDB

Diese Klasse bietet Funktionen, um die SQL-Datenbank abzufragen.

In diesem Kapitel

Beispiel-Skript: 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()

Rückgabewerte

Gibt die Spaltenanzahl im Ergebnis einer erfolgreichen Abfrage zurück.

Connect

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

Rückgabewerte

Wahr: Verbindung wurde erfolgreich ausgeführt. Falsch: ein Fehler ist aufgetreten.

Parameter

- sConnectString – Eine Client-spezifische Verbindungszeichenfolge. Weitere Informationen finden Sie im Abschnitt "Hinweise".
- iClientType – Clienttyp für die Kommunikation, Optionen siehe unten:

Hinweise

Die Verbindungszeichenfolge ist Client-spezifisch. Unten sind die aktuell unterstützten Clients aufgeführt.

CLIENT _ODBC

Beispiel Verbindungszeichenfolge:

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

Diese Verbindungszeichenfolge verwendet eine Datenquelle namens Test und stellt den Benutzernamen (UID) und das Passwort (PWD) für DSN bereit.

Wenn kein Benutzername und Passwort nötig sind, kann die Verbindung folgendermaßen formatiert werden.

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

Network Monitor wird als Service ausgeführt, die Datenquelle muss eine Systemdatenquelle sein. Im Gegensatz zu IDE, wo ebenfalls Benutzer-DSN verwendet werden kann.

CLIENT _SQLSERVER

Beispiel Verbindungszeichenfolge:

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

Zur Verbindung mit einer benannten Instanz von SQL-Server 2000:

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

CLIENT_MYSQL

Beispiel Verbindungszeichenfolge (unter Verwendung des standardmäßigen 3306 Port):

```
sConnectString =  myserver@mydatabase
```

Verbindung mit einem Server, mit einem benutzerdefinierten empfangsbereiten Port

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

Beachten Sie, dass die MySQL-Clientbibliothek installiert werden muss ("MySQL Workbench" oder "Connector/C (libmysql)") und die standardmäßige PATH-Systemvariable bei Windows aufgenommen werden muss, damit diese von **Network Monitor** und IDE gefunden werden kann.

Connect(2)

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

Rückgabewerte

Wahr: Verbindung wurde erfolgreich ausgeführt. Falsch: ein Fehler ist aufgetreten.

Parameter

- sConnectString – Eine Client-spezifische Verbindungszeichenfolge. Weitere Informationen finden Sie im Abschnitt "Hinweise".

- sUsername – Für die Verbindung zu verwendende Anmeldeinformationen.
- sPassword – Für die Verbindung zu verwendende Anmeldeinformationen. Wenn der Benutzername festgelegt ist, muss dieses Feld ausgefüllt sein.
- iClientType – Clienttyp für die Kommunikation. Aktuell ist die einzige Option CLIENT_ODBC.

Hinweise

Die Verbindungszeichenfolge ist Client-spezifisch. Weitere Informationen finden Sie unter **Verbinden** *(siehe 30)* .

Execute

bool Execute(string sSQL)

Rückgabewerte

Wahr: die Abfrage wurde erfolgreich ausgeführt. Falsch: ein Fehler ist aufgetreten.

Parameter

- sSQL – Eine SQL-Anweisung.

Hinweise

Wenn ein Fehler auftritt, wenn die SQL-Anweisung ausgeführt wird, gibt die Funktion GetErrorDescription() eine Zeichenfolge mit der Fehlerbeschreibung zurück.

GetCol

string GetCol(int iIndex)

Rückgabewerte

Gibt eine Zeichenfolge mit den abgerufenen Daten der Spalte zurück.

Parameter

- iIndex – Ein einsbasierter Spaltenindex.

Hinweise

Beachten Sie, dass der Spaltenindex einsbasiert ist. Wenn ColCount() 10 zurück gibt, ist der gültige Indexbereich 1–10. Um den Datentyp abzurufen, GetColType() abrufen

GetCol_AsDateTime

TLuaDateTime GetCol_AsDateTime(int iIndex)

Rückgabewerte

Gibt eine TLuaDateTime-Struktur mit dem abgerufenen Datum und der Zeit aus der Spalte zurück.

Parameter

- iIndex – Ein einsbasierter Spaltenindex.

Hinweise

Beachten Sie, dass der Spaltenindex einsbasiert ist. Wenn ColCount() 10 zurück gibt, ist der gültige

Indexbereich 1–10. Diese Funktion muss nicht abgerufen werden, wenn die Spalte nicht vom Typ Datum/Zeit ist.

GetColType

int GetColType(int iIndex)

Rückgabewerte

Gibt eine Ganzzahl zurück, die den in der Spalte enthaltenen Datentyp darstellt.

Parameter

- iIndex – Ein einsbasierter Spaltenindex.

Hinweise

Die Funktion GetCol() gibt die Daten immer als Zeichenfolge zurück. Die Funktion GetColType kann verwendet werden, um den Spaltentyp zu bestimmen, der durch die Umwandlung der Zeichenfolge nach der Extraktion gemacht werden kann. Die folgenden Typen existieren:

- TYPE_BOOL – Boolescher Wert
- TYPE_NUMERIC – Numerisch
- TYPE_SHORT – Kurz
- TYPE_LONG – Lang
- TYPE_DOUBLE – Doppelt (real)
- TYPE_DATETIME – Datum/Zeit
- TYPE_STRING – Zeichenfolge
- TYPE_UNKNOWN – Unbekannter Feldtyp/nicht unterstützt
- TYPE_BYTES – Byte
- TYPE_LONG_BINARY – Langer Binärwert
- TYPE_LONG_CHAR – Lange Zeichen
- TYPE_BLOB – Binärgerät
- TYPE_DBMS_SPECIFIC – Client-spezifische Daten (keine Konvertierung)

GetErrorDescription

string GetErrorDescription(void)

Rückgabewerte

Gibt die letzte Fehlerbeschreibung zurück, die von TLuaDB-API erzeugt wurde.

NextRow

bool NextRow()

Rückgabewerte

Wahr: ein neues Ergebnis wurde abgerufen. Falsch: es gibt keine weiteren Ergebnisse für die Abfrage.

Hinweise

Die Funktion ruft ein neues Ergebnis ab, dass bei einem bereits durchgeführten Abruf der Execute-Funktion erzeugt wurde. Diese Funktion muss vor dem ersten Abruf der Funktionen

ColCount, GetCol oder GetColType abgerufen werden.

ResultAvailable

bool ResultAvailable()

Rückgabewerte

Wird als wahr zurückgegeben, wenn die SQL-Anweisung ein Ergebnis zurückgegeben hat.

Hinweise

Es ist festgelegt, dass keine Anweisungen eingefügt, aktualisiert oder gelöscht werden, die Ergebnisse nach der Ausführung zurückgegeben haben. Der Rückgabewert der Execute-Funktion muss verwendet werden, um vor Abruf dieser Funktion festzustellen, ob eine Anweisung erfolgreich ausgeführt wurde oder nicht.

Kapitel 7

TLuaDNS

Die Klasse stellt Funktionen zur Abfrage von DNS-Servern bereit.

In diesem Kapitel

Beispiel-Skript: 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
```

Begin

bool Begin(bool bUselocalhost=false)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null.

Parameter

- bUselocahost – Stellen Sie den Wert auf wahr ein, wenn Sie einen DNS-Server auf dem Hostrechner von **Network Monitor** abfragen wollen.

Hinweise

Die Funktion startet eine DNS-Abfrage. Wenn Sie die Abfrage ausführen, bevor die der Funktion Begin ausgeführt wird, kann es zu unbekannten Ergebnissen kommen.

End

void End()

Hinweise

Die Funktion beendet eine Transaktion und setzt den Bestand zurück, sodass eine neue Abfrage ausgeführt werden kann. Wenn diese Funktion nicht abgerufen wird, ist das Ergebnis der nächsten Abfrage unvorhersehbar.

GetErrorDescription

string GetErrorDescription(void)

Rückgabewerte

Gibt die letzte Fehlerbeschreibung zurück, die von TLuaDNS-API erzeugt wurde.

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);
```

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null.

Parameter

Record DA DNS Datensatztyp, der die abgefragten Informationen vom DNS-Server erhält.// Aufzeichnen ...

Hinweise

Diese Funktion wird als wahr zurück gegeben, wenn sie erfolgreich einen neuen Datensatz abgerufen hat. Die Funktion wird als falsch zurückgegeben, wenn keine weiteren Datensätze abgerufen werden können.

Query

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

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null.

Parameter

- sDomainName – Domäne, die abgefragt wird
- iRecordType – Eines der Datensatztypen, die im Abschnitt "Hinweise" aufgelistet ist.
- bBypassCache – Standardmäßig auf falsch eingestellt. Wenn der Wert auf wahr eingestellt wird, umgeht die Abfrage den lokalen Resolver und fragt den DNS-Server direkt an.

Hinweise

Die Funktion sendet eine Abfrage an den DNS-Server. Das Ergebnis kann mit einer oder mehreren Abfragen der Funktion Next() extrahiert werden.

Die folgenden Datensatztypen können abgefragt werden:

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

Referenz zu DNS-Datensatztyp siehe: **http://www.iana.org/assignments/dns-parameters** *(http://www.iana.org/assignments/dns-parameters)*

TLuaDNS_ARecord

Klassenmitglieder

- int m_iTTL
- string m_sIPAddress;

TLuaDNS_CNAMERecord

Klassenmitglieder

- int m_iTTL
- string m_sHostname

TLuaDNS_MXRecord

Klassenmitglieder

- int m_iPreference
- string m_sNameExchange

TLuaDNS_NSRecord

Klassenmitglieder

- int m_iTTL
- string m_sHostname

TLuaDNS_PTRRecord

Klassenmitglieder

- int m_iTTL
- string m_sHostname

TLuaDNS_SOARecord

Klassenmitglieder

- 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

Klassenmitglieder

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

Kapitel 8

TLuaFile

Die Klasse bietet grundlegende Dateiverarbeitungsroutinen und unterstützt sowohl Binär- als auch Textdateien. Alle Dateivorgänge stehen mit dem Kontext in Beziehung, in dem das Skript ausgeführt wird. Wenn das Skript z. B. von einem Bestand mit der Adresse `\\myserver` ausgeführt wird, wird der Dateipfad `c:\test.txt` in `\\myserver\C$\test.txt` übersetzt.

Es gibt eine Ausnahme: Wenn die Klasse mit dem Wert wahr initialisiert wird, dann stehen alle Vorgänge in Beziehung mit dem Hostrechner von **Network Monitor**. Weitere Informationen finden Sie unter Beispiel-Skript drei in dieser Klasse.

In diesem Kapitel

Beispiel-Skripte: TLuaFile

Beispiel 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
```

Beispiel 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
```

Beispiel 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
```

Close

int Close()

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Hinweise

Schließt die Datei, die zum Lesen und Schreiben nicht mehr verfügbar ist. Wenn Sie die Datei nicht schließen, bevor der Bestand zerstört wird, wird dies vom Destruktor übernommen.

CopyFile

int CopyFile(string sSource,string sDest)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sSource – Pfad und Name der Datei, die kopiert werden soll.
- sDest – Neuer Pfad und Name der neuen Datei.

Hinweise

Die Funktion kopiert eine Datei. Verzeichnisse können mit dieser Funktion nicht kopiert werden.

CreateDirectory

int CreateDirectory(string sPath)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sPath – Pfad des Verzeichnisses, das erstellt werden soll.

Hinweise

Das übergeordnete Verzeichnis, das erstellt werden soll, muss existieren, andernfalls schlägt diese Funktion fehl.

DeleteDirectory

int DeleteDirectory(string sDirectory)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sDirectory – Pfad des Verzeichnisses, das gelöscht werden soll.

Hinweise

Das Verzeichnis muss leer sein und darf kein Stammverzeichnis oder das aktuelle Arbeitsverzeichnis sein.

DeleteFile

int DeleteFile(string sFileName)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sFileName – Pfad und Name der Datei, die gelöscht werden soll.

Hinweise

Die Funktion löscht eine Datei. Verzeichnisse können mit dieser Funktion nicht gelöscht werden.

DoesFileExist

int DoesFileExist(string sFile);

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sFile – Pfad und Name der Datei.

Hinweise

Gibt einen Wert zurück, der ungleich null ist, wenn die Datei existiert.

GetDirectoryList

string GetDirectoryList(string sDirectory)

Rückgabewerte

Gibt eine Zeichenfolge mit den Unterverzeichnissen zurück, die mit einem Zeilenumbruch getrennt sind. Andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sDirectory – Pfad des Verzeichnisses, nach dem gesucht werden soll.

Hinweise

Die Rückgabezeichenfolge enthält alle Unterverzeichnisse in dem angegebenen Verzeichnis. Jedes Verzeichnis wird mit dem vollständigen Pfad zurückgegeben. In der Zeichenfolge werden Verzeichnisse mit einem Zeilenumbruchsymbol getrennt.

GetFileAccessedTime

TLuaDateTime GetFileAccessedTime(string sFilename)

Rückgabewerte

Gibt einen TLuaDateTime-Bestand mit dem Zeitpunkt des letzten Dateizugriffs zurück. Andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sFilename – Pfad und Name der Datei.

Hinweise

Verwenden Sie den TLuaDateTime-Bestand, um eine Zeichenfolge zu erstellen, die die im Bestand gespeicherte Zeit darstellt oder vergleichen Sie ihn mit einem anderen TLuaDateTime-Bestand.

GetFileCreatedTime

TLuaDateTime GetFileCreatedTime(string sFilename)

Rückgabewerte

Gibt einen TLuaDateTime-Bestand mit dem Zeitpunkt zurück, als die Datei erstellt wurde. Andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- pFilename – Pfad und Name der Datei.

Hinweise

Verwenden Sie den TLuaDateTime-Bestand, um eine Zeichenfolge zu erstellen, die die im Bestand gespeicherte Zeit darstellt oder vergleichen Sie ihn mit einem anderen TLuaDateTime-Bestand.

GetFileList

string GetFileList(string sDirectory,string sWildCard)

Rückgabewerte

Gibt eine Zeichenfolge mit den aufgelisteten Dateien zurück, die mit einem Zeilenumbruch getrennt sind. Andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sDirectory – Pfad des Verzeichnisses.
- sWildCard – Platzhalter, um nach gesuchten Dateien zu filtern. Verwenden Sie um alle Dateien im Verzeichnis abzurufen den Platzhalter *.*.

Hinweise

Die Rückgabezeichenfolge enthält alle Dateien in dem angegebenen Verzeichnis, die mit dem Platzhalter übereinstimmen. Jede Datei wird mit dem vollständigen Pfad zurückgegeben. In der Zeichenfolge werden Dateien mit einem Zeilenumbruchsymbol getrennt.

GetFileModifiedTime

TLuaDateTime GetFileModifiedTime(string sFilename)

Rückgabewerte

Gibt einen TLuaDateTime-Bestand mit dem Zeitpunkt zurück, als die Datei zum letzten Mal verändert wurde. Andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sFilename – Pfad und Name der Datei.

Hinweise

Verwenden Sie einen TLuaDateTime-Bestand, um eine Zeichenfolge zu erstellen, die die im Bestand gespeicherte Zeit darstellt oder vergleichen Sie ihn mit einem anderen TLuaDateTime-Bestand

GetFileSize

int GetFileSize(string sFile)

Rückgabewerte

Dateigröße in Anzahl der Byte, wenn die Funktion erfolgreich ist. Wenn der Zugriff auf die Datei nicht möglich ist, ist der Rückgabewert -1.

Parameter

- sFile – Pfad und Name der Datei.

Hinweise

Die Funktion ist auf Dateien begrenzt die kleiner als 2^31 Byte sind.

GetFileSizeMB

int GetFileSizeMB(string sFile)

Rückgabewerte

Dateigröße in Anzahl der Megabytes, wenn die Funktion erfolgreich ist. Wenn der Zugriff auf die Datei nicht möglich ist, ist der Rückgabewert -1.

Parameter

- sFile – Pfad und Name der Datei.

GetFileStatus

int GetFileStatus(string sFile)

Rückgabewerte

Ein Wert, der den aktuellen Status der Datei beschreibt, wenn die Funktion erfolgreich ist. Wenn der Zugriff auf die Datei nicht möglich ist, ist der Rückgabewert -1.

Parameter

- sFile – Pfad und Name der Datei.

Hinweise

Die Funktion gibt eine Kombination der folgenden Markierungen zurück, um den Dateistatus zu beschreiben.

FILE_NORMAL = 0x00	Normale Datei.
FILE_READONLY = 0x01	Die Datei ist schreibgeschützt.
FILE_HIDDEN = 0x02	Die Datei ist ausgeblendet.
FILE_SYSTEM = 0x04	Bei der Datei handelt es sich um eine Systemdatei.
FILE_VOLUME = 0x08	Die Datei ist ein Datenträger.
FILE_DIR = 0x10	Die Datei ist ein Verzeichnis.
FILE_ARCHIV = 0x20	Datei mit Archiv-Bit-Satz

MoveFile

int MoveFile(string sSource,string sDest)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sSource – Pfad und Name der Datei, die verschoben werden soll. sDest Neuer Pfad und Name der Datei.

Hinweise

Die Funktion verschiebt eine Datei. Das Verzeichnis, in das die Datei verschoben werden soll, muss existieren, andernfalls schlägt die Funktion fehl. Verzeichnisse können mit dieser Funktion nicht verschoben werden.

Open

int Open(string sFileName, bool bCreate=false)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sFileName – Name und Pfad der Datei, die geöffnet oder erstellt werden soll. bCreate: Wenn der Wert auf ungleich null eingestellt wird, erzeugt die Funktion eine Datei. Wenn die Datei bereits existiert, wird deren Inhalt zerstört.

Hinweise

Wenn bCreate auf null eingestellt ist und die Datei nicht existiert, schlägt diese Funktion fehl.

Read

string,int Read(int iSize)

Rückgabewerte

Eine Zeichenfolge, wenn die Funktion erfolgreich ist. iSize wird auf die Größe der zurückgegebenen Daten eingestellt. Andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- iSize – Datengröße, die in Byte gelesen werden soll.

Hinweise

Die Funktion liest die angegebene Datengröße aus der Datei und verschiebt den Dateizeiger um dieselbe Menge nach vorne. Wenn das Ende der Datei erreicht wird, wird der Lesevorgang gestoppt und die gelesene Daten werden zurückgegeben, bis das Ende der Datei erreicht wird.

ReadData

local data,int ReadData(int iSize)

Rückgabewerte

Die Funktion gibt einen speziellen Datentyp zurück, der nur zusammen mit ConvertFromUTF16 verwendet werden kann. Wenn die Funktion erfolgreich ist, wird iSize auf die Größe der zurückgegebenen Daten eingestellt. Andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- iSize – Datengröße, die in Byte gelesen werden soll.

Hinweise

Die Funktion liest die angegebene Datengröße aus der Datei und verschiebt den Dateizeiger um dieselbe Menge nach vorne. Wenn das Ende der Datei erreicht wird, wird der Lesevorgang gestoppt und die gelesene Daten werden zurückgegeben, bis das Ende der Datei erreicht wird.

RenameFile

int RenameFile(string sOrgFile,string sNewFile)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sOrgFile – Pfad und Name der Datei, die umbenannt werden soll.
- sNewFile pFilename – Neuer Pfad und Name der Datei.

Hinweise

Die Funktion benennt eine Datei um. Verzeichnisse können mit dieser Funktion nicht umbenannt werden.

SeekFromCurrent

int SeekFromCurrent(int iNumberofBytes)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- iNumberofBytes – Anzahl der Byte, um die der Dateizeiger umpositioniert werden soll; im Verhältnis zur aktuellen Position.

Hinweise

Die Funktion verschiebt den Dateizeiger im Verhältnis zu seiner aktuellen Position. Es können sowohl positive als auch negative Werte angegeben werden. Der Zeiger kann unter dem Dateiende positioniert werden. Er löscht dann das Ende des Dateimarkers.

SeekFromEnd

int SeekFromEnd(int iNumberofBytes)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- iNumberofBytes – Anzahl der Byte, um die der Dateizeiger umpositioniert werden soll; ausgehend vom Dateiende.

Hinweise

Die Funktion verschiebt die aktuelle Position des Dateizeigers ausgehend vom Dateiende zu iNumberofBytes. Beachten Sie, dass iNumberofBytes negativ sein muss, um den Dateizeiger in der Datei "nach oben" zu bewegen.

SeekFromStart

int SeekFromStart(int iNumberofBytes)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- iNumberofBytes – Anzahl der Byte, um die der Dateizeiger umpositioniert werden soll; ausgehend vom Dateianfang.

Hinweise

Die Funktion verschiebt die aktuelle Position des Dateizeigers ausgehend vom Dateianfang zu iNumberofBytes. Der Zeiger kann unter dem Dateiende positioniert werden. Er löscht dann das Ende des Dateimarkers.

Write

int Write(string sData,int iSize)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sData – Datenarray, das in die Datei geschrieben werden soll.
- iSize – Datengröße, die geschrieben werden sollen.

Hinweise

Die Funktion schreibt ausgehend von der aktuelle Position des Dateizeigers aus. Die Datei wird, falls erforderlich, erweitert, wenn das aktuelle Dateiende überschritten wird. Die Funktion schlägt fehl, wenn die geöffnete Datei schreibgeschützt ist.

Kapitel 9

TLuaFTPClient

Die Klasse implementiert einen FTP-Client, der grundlegende FTP-Vorgänge ausführen kann.

In diesem Kapitel

Beispiel-Skript: 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)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sDir – Pfad des neuen aktuellen Verzeichnisses.

Hinweise

Die Funktion schlägt fehl, wenn das Verzeichnis nicht existiert.

Close

Close()

Hinweise

Die Funktion schließt die aktuelle Verbindung mit dem FTP-Server. Die Funktion muss aufgerufen werden, um die aktuelle Verbindung zu schließen.

CloseFile

void CloseFile()

Hinweise

Schließt eine mit OpenFile geöffnete Datei.

Connect

bool Connect(unsigned int iPort=21)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sUsername – Zeichenfolge mit dem Benutzernamen
- sPassword – Zeichenfolge mit dem Passwort
- iPort – Standardport 21 (standardmäßiger FTP-Port)

Hinweise

Die Funktion stellt mit dem zur Verfügung gestellten Benutzernamen und Passwort eine Verbindung zu einem FTP-Server her. Der Port kann vom Standardport 21 geändert werden, wenn der FTP-Server an einen anderen Port gebunden ist.

CreateDirectory

int CreateDirectory(string sDir)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sDir – Pfad des Verzeichnisses, das erstellt werden soll.

Hinweise

Das Verzeichnis, das vom neuen Verzeichnis erstellt wird, muss existieren oder diese Funktion schlägt fehl.

DeleteDirectory

int DeleteDirectory(string sDir)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sDir – Pfad des Verzeichnisses, das gelöscht werden soll.

Hinweise

Die Funktion schlägt fehl, wenn das Verzeichnis nicht leer ist.

DeleteFile

int DeleteFile(string sFilename)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sFilename – Pfad und Name der Datei, die gelöscht werden soll.

Hinweise

Die Funktion schlägt fehl, wenn die Datei nicht existiert.

FindDirectory

string FindDirectory()

Rückgabewerte

Gibt eine Zeichenfolge mit den aufgelisteten Verzeichnissen zurück, die mit einem Zeilenumbruch getrennt sind. Andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Hinweise

Die Rückgabezeichenfolge enthält alle Unterverzeichnisse des aktuellen Verzeichnisses. Jedes Verzeichnis wird mit dem vollständigen Pfad zurückgegeben. Verzeichnisse sind in der Zeichenfolge mit einem Zeilenumbruchsymbol getrennt.

FindFile

string FindFile(string sWildcard)

Rückgabewerte

Gibt eine Zeichenfolge mit den aufgelisteten Dateien zurück, die mit einem Zeilenumbruch getrennt sind. Andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sWildCard – Platzhalter, um nach gesuchten Dateien zu filtern. Verwenden Sie um alle Dateien im Verzeichnis abzurufen den Platzhalter *.*.

Hinweise

Die Rückgabezeichenfolge enthält alle Dateien in dem aktuellen Verzeichnis, die mit dem Platzhalter übereinstimmen. Jede Datei wird mit dem vollständigen Pfad zurückgegeben. In der Zeichenfolge werden Dateien mit einem Zeilenumbruchsymbol getrennt.

GetCurrentDirectory

int GetCurrentDirectory(string sDir)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sDir – Pfad des neuen aktuellen Verzeichnisses.

Hinweise

Die Funktion schlägt fehl, wenn das Verzeichnis nicht existiert.

GetFileModifiedTime

TLuaDateTime GetFileModifiedTime(string sFilename)

Rückgabewerte

Ein TLuaDateTime-Bestand, der die Zeit der letzten Dateiänderung enthält, wenn die Funktion erfolgreich ist. Andernfalls ist der TLuaDateTime-Bestand auf 0 eingestellt.

Parameter

- sFilename – Name der Datei im aktuellen Verzeichnis.

Hinweise

Die Datei muss im aktuellen Verzeichnis sein, damit die Funktion erfolgreich ist. Relative Pfade funktionieren nicht.

GetFileSize

int GetFileSize(string sFilename)

Rückgabewerte

Dateigröße in Anzahl der Byte, wenn die Funktion erfolgreich ist. Wenn der Zugriff auf die Datei nicht möglich ist, ist der Rückgabewert -1.

Parameter

- sFilename – Pfad und Name der Datei.

Hinweise

Die Funktion ist auf Dateien begrenzt die kleiner als 2^31 Byte sind.

OpenFile

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

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sFilename – Name der Datei, die geöffnet werden soll.
- bWrite – Wenn der Wert ungleich null ist, wird die Datei im Schreibmodus geöffnet. Wenn der Wert null ist, wird die Datei schreibgeschützt geöffnet.
- bText – Wenn der Wert ungleich null ist, wird die Datei Textübersetzungsmodus geöffnet. Andernfalls wird die Datei im Binärmodus geöffnet.

Hinweise

Die Funktion öffnet eine Datei auf dem FTP-Server. Nachdem die Datei geöffnet wurde, kann der Abruf der Lese- und Schreibfunktion ausgeführt werden. Die Funktion CloseFile wird zum Schließen der Datei verwendet.

Read

string Read(int iSize)

Rückgabewerte

Ein Array mit den Daten, die von der Datei gelesen wurden. Andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- iSize – Größe, die von der Datei gelesen werden soll. Wenn die Funktion zurückgegeben wird, wird angezeigt, wie viel gelesen wurde. Der Wert kann unter der angeforderten Größe liegen.

Hinweise

Diese Funktion schlägt fehl, wenn eine Datei nicht geöffnet wurde.

RenameFile

bool RenameFile(string sOldname,string sNewname)

Rückgabewerte

Der Wert ist bei wahr, wenn die Funktion erfolgreich ist. Andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sOldname – Alter Name der Datei, die umbenannt werden soll. sNew name Neuer Name der Datei.

Hinweise

Die Funktion schlägt fehl, wenn eine Datei mit demselben Namen existiert, wie im zweiten Parameter angegeben.

Write

int Write(string sData,int iSize)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sData – Datenarray, das in die Datei geschrieben werden soll.
- iSize – Arraygröße, die geschrieben werden sollen.

Hinweise

Diese Funktion schlägt fehl, wenn eine Datei nicht geöffnet wurde oder wenn eine Datei im schreibgeschützten Modus geöffnet ist. Die Funktion kann ebenfalls fehlschlagen, wenn der Speicher auf dem FTP-Server belegt ist.

Kapitel 10

TLuaHTTPClient

Die Klasse implementiert einen grundlegenden HTTP-Client.

In diesem Kapitel

Beispiel-Skript: 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
```

Connect

bool Connect(bool bSecure,unsigned short iPort)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- iPort – Portnummer, mit der die Verbindung erstellt wird, Standardport 80
- bSecure – Auf ungleich null eingestellt, wenn die Verbindung mit HTTPS erstellt wird
- sUsername – Optionaler Benutzername für Server, bei denen eine Authentifizierung erforderlich ist
- sPassword – Optionales Passwort für Server, bei denen eine Authentifizierung erforderlich ist

Hinweise

Die Funktion stellt mit den zur Verfügung gestellten Parametern eine Verbindung mit einem HTTP-Server auf. Diese Funktion muss abgerufen werden, bevor eine andere Funktion in dieser Klasse abgerufen wird.

Close

Close()

Hinweise

Schließt eine offene Verbindung.

Get

int Get(string sUrl,string sHeaders=NULL)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sUrl – URL, die zur Basis-URL der Site relativ ist.
- sHeaders – Optionale Zeichenfolge der Kopfzeile, die Kopfzeilen enthält, die mit der Anfrage gesendet werden.

Hinweise

Die Verbindung ist im Kontext des Bestands immer geöffnet. Aus diesem Grund muss die URL, die für diese Funktion zur Verfügung gestellt wird, relativ zur Basis-URL sein.

Post

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

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sUrl – URL, die zur Basis-URL der Site relativ ist.
- sHeaders – Optionale Zeichenfolge der Kopfzeile, die Kopfzeilen enthält, die mit der Anfrage gesendet werden.
- sData – Optionale Daten, die in die Post-Anforderung aufgenommen werden.

Hinweise

Die Verbindung ist im Kontext des Bestands immer geöffnet. Aus diesem Grund muss die URL, die für diese Funktion zur Verfügung gestellt wird, relativ zur Basis-URL sein. Jede angegebene Kopfzeile muss mit einem Paar aus Zeilenumbruch und Zeilenvorschub abgeschlossen sein.

GetContent

string GetContent(int iSize)

Rückgabewert

Wenn die Funktion erfolgreich ist, ist der Wert eine Zeichenfolge mit dem Inhalt einer GET-Anfrage. Andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- iSize – Stellt bei Funktionsrückgabe auf die Größe der zurückgegebenen Zeichenfolge ein.

Hinweise

Der Inhalt bezieht sich auf die Daten, die von einer Anfrage zurückgegeben werden, die auf eine Kopfzeile folgt.

GetHeadersRaw

string GetHeadersRaw()

Rückgabewerte

Wenn die Funktion erfolgreich ist, ist der Wert eine Zeichenfolge mit den Kopfzeilen, die von der Anfrage zurückgegeben werden. Andernfalls ist der Wert eine leere Zeichenfolge.

Hinweise

Die Kopfzeilen müssen in derselben Form zurückgegeben werden, wie sie vom Server gesendet werden.

GetHeaderLocation

string GetHeaderLocation()

Rückgabewerte

Wenn die Funktion erfolgreich ist, ist der Wert eine Zeichenfolge mit der Kopfzeile "Speicherort:". Andernfalls ist der Wert eine leere Zeichenfolge.

GetHeaderContentLength

int GetHeaderContentLength()

Rückgabewerte

Die Länge in Byte des Anfrageinhalts.

GetHeaderContentType

string GetHeaderContentType()

Rückgabewert

Wenn die Funktion erfolgreich ist, ist der Wert eine Zeichenfolge mit der Kopfzeile "Inhaltstyp:". Andernfalls ist der Wert eine leere Zeichenfolge.

GetHeaderContentTransferEncoding

string GetHeaderContentTransferEncoding()

Rückgabewert

Wenn die Funktion erfolgreich ist, ist der Wert eine Zeichenfolge mit der Kopfzeile "Transfercodierung:". Andernfalls ist der Wert eine leere Zeichenfolge.

GetHeaderCookies

string GetHeaderCookies()

Rückgabewert

Wenn die Funktion erfolgreich ist, ist der Wert eine Zeichenfolge mit allen Cookies, die vom Server mit Zeilenumbruch getrennt zurückgegeben werden. Andernfalls ist der Wert eine leere Zeichenfolge.

GetHeaderCookie

string GetHeaderCookie(int iIndex)

Rückgabewert

Eine Zeichenfolge mit der angefragten Cookiezeichenfolge.

Parameter

- iIndex – Ein nullbasierter Index, der festlegt welche Cookies zurückgegeben werden.

Hinweise

Wenn eine negative Zahl oder ein Index außerhalb des Bereichs angegeben wird, wird eine leere Zeichenfolge zurückgegeben.

GetHeaderCookieCount

int GetHeaderCookieCount()

Rückgabewert

Die Anzahl der Cookies, die von der Anfrage zurückgegeben werden.

GetHeaderDate

string GetHeaderDate()

Rückgabewert

Wenn die Funktion erfolgreich ist, ist der Wert eine Zeichenfolge mit der Kopfzeile "Datum:". Andernfalls ist der Wert eine leere Zeichenfolge.

GetHeaderExpires

string GetHeaderExpires()

Rückgabewert

Wenn die Funktion erfolgreich ist, ist der Wert eine Zeichenfolge mit der Kopfzeile "Läuft ab:". Andernfalls ist der Wert eine leere Zeichenfolge.

GetHeaderHost

string GetHeaderHost()

Rückgabewert

Wenn die Funktion erfolgreich ist, ist der Wert eine Zeichenfolge mit der Kopfzeile "Host:". Andernfalls ist der Wert eine leere Zeichenfolge.

Kapitel 11

TLuaICMP

Die Klasse stellt Ping- und Ablaufverfolgungsfunktionen zur Verfügung, die zur Diagnose einer Netzwerkverbindung verwendet werden können.

In diesem Kapitel

Beispiel-Skript: 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)

Rückgabewerte

Die Funktion wird bei erfolgreicher Ablaufverfolgung mit dem Wert "Wahr" zurückgegeben. Wenn diese fehlschlägt, ist der Wert "Falsch".

Parameter

- iPacketsToSend – Eine positive Ganzzahl zwischen 1 und 255
- iPacketSize – Größe der zu sendenden Pakete, eine Ganzzahl zwischen null und
- bNoFragment – Auf "Wahr" eingestellt, damit der Versand von Paketen nicht fragmentiert wird. Die Funktion schlägt fehl und wird mit dem Wert "Falsch" zurückgegeben, wenn die Option eingestellt ist und das Paket fragmentiert ist.
- iTimeoutMs – Höchstzeit in ms, die die Funktion auf das zurückgegebene Paket wartet.

Hinweise

Indem bNoFragment auf "Wahr" eingestellt wird, ist es möglich die größte Framegröße für eine Route zu testen; passen Sie dazu iPacketSize an, bis die Funktion aufgrund von Fragmentierung fehlschlägt.

EndTrace

EndTrace()

Hinweise

Die Funktion führt eine Bereinigung der verwendeten Ressourcen aus. Muss für jeden Abruf BeginTrace() aufgerufen werden.

NextTraceResult

bool NextTraceResult(TLuaICMPTraceResult Result)

Rückgabewerte

Die Funktion wird als "Wahr" zurückgegeben, solange ein Ergebnis verfügbar ist.

Parameter

- Result – Eine Variable **TLuaICMPTraceResult**, die das Ergebnis des aktuellen Hop erhält.

Hinweise

Rufen Sie die Funktion ab bis null zurückgegeben wird, um das Resultset zu durchlaufen.

Ping

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

Rückgabewerte

Die Funktion gibt einen Bestand TLuaICMPPingResult zurück, der das Ergebnis des Vorgangs enthält.

Parameter

- iPacketsToSend – Eine positive Ganzzahl zwischen 1 und 255
- iPacketSize – Größe der zu sendenden Pakete, eine Ganzzahl zwischen null und
- bNoFragment – Auf "Wahr" eingestellt, damit der Versand von Paketen nicht fragmentiert wird. Die Funktion schlägt fehl und wird mit dem Wert "Falsch" zurückgegeben, wenn die Option eingestellt ist und das Paket fragmentiert ist.
- iTimeoutMs – Höchstzeit in ms, die die Funktion auf das zurückgegebene Paket wartet.

Hinweise

Durch Einstellen des Arguments bNoFragment kann die größtmögliche Framegröße für eine Route getestet werden.

Kapitel 12

TLuaICMPPingResult

TLuaICMPPingResult ist eine schreibgeschützte Klasse.

Klassenmitglieder

- int m_iRoundTripTimeMs
- float m_fPacketloss

Kapitel 13

TLuaICMPTraceResult

TLuaICMPTraceResult ist eine schreibgeschützte Klasse.

Klassenmitglieder

- string m_IP
- string m_Hostname
- int m_iRoundTripTimeMs

Kapitel 14

TLuaPowershell

Führt eine Powershell-Befehlszeichenfolge aus. Gibt die standardmäßige Powershell-Befehlsausgabe, Fehlerausgabe, Fehlercodes und Fehlerbeschreibungen zurück.

Voraussetzungen

Die TLuaPowershell-Bibliothek verwendet WinRS (Windows Remoteshell), um sich mit einem für Windows Remoteverwaltung aktivierten Bestand zu verbinden.

Informationen von Microsoft bezüglich WinRM/WinRS
(http://msdn.microsoft.com/en-us/library/aa384426%28v=vs.85%29.aspx):

"Windows Remoteverwaltung (WinRM) ist die Implementierung von Microsoft des WS-Verwaltungsprotokolls: ein standardmäßiges, auf Simple Object Access-Protokoll (SOAP) basiertes, firewallfreundliches Protokoll, durch das Hardware und Betriebssysteme verschiedener Anbieter miteinander arbeiten können.

Sie können Skripterstellung für Objekte von WinRM, das Befehlszeilentool WinRM oder das Windows Remoteshell-Befehlszeilentool WinRS verwenden, um Verwaltungsdaten von lokalen und Remotecomputern zu erhalten, die eventuell über Baseboard Management Controller (BCMs) verfügen. Wenn der Computer eine Windows-basierte Betriebssystemversion ausführt, die WinRM beinhaltet, werden Verwaltungsdaten von Windows Management Instrumentation (WMI) zur Verfügung gestellt."

Geben Sie Folgendes in eine Eingabeaufforderung ein, um WinRM auf einem Bestand zu aktivieren:

```
winrm /quickconfig
```

Sie können Powershell-Befehle oder Skripte ausführen. Beachten Sie jedoch, dass die ausgeführten Befehle oder Skripte, die WinRS verwenden, nicht von der Benutzeroberfläche abhängen dürfen. Sie können z. B. keine Befehle ausführen, die Sie auffordern, an der lokalen Konsole eine beliebige Taste zu drücken oder die andere interaktive Reaktionen erfordern.

In diesem Kapitel

Beispiel-Skript: 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)
```

Beispiel-Skript: TLuaPowershell (Windows Scripting)

```
--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)
```

Open

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

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null. Ein bestimmter Fehlercode kann durch Aufrufen von GetStdErr() abgerufen werden.

Parameter

- iPort – TCP-Port, der von Windows Remoteverwaltung (WinRM) verwendet wird.
- bSecure – Wenn SSL verwendet wird, auf "Wahr" (Standard) eingestellt und bei nicht-SSL-Verbindungen auf "Falsch".
- dwWait – Das Zeitlimit für eine erfolgreiche Verbindung.

ExecuteCommand

ExecuteCommand(const char *pCommand)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null. Ein bestimmter Fehlercode kann durch Aufrufen der globalen Funktion GetStdErr abgerufen werden.

Parameter

- pCommand – Die Powershell-Befehlszeichenfolge.

GetStdOut

string GetStdOut(void)

Rückgabewerte

Gibt die Standardausgabe vom Remotehost zurück.

GetStdErr

string GetStdErr(void)

Rückgabewerte

Gibt die Standardfehlerausgabe vom Remotehost zurück.

GetErrorDescription

string GetErrorDescription(void)

Rückgabewerte

Gibt die letzte Fehlerbeschreibung zurück, die vom Remotehost als Zeichenfolge erzeugt wurde.

GetErrorCode

dWord GetErrorCode(void)

Rückgabewerte

Gibt den letzten Fehlercode zurück, der vom Remotehost als Zeichenfolge erzeugt wurde.

Kapitel 15

TLuaRegistry

Die Klasse ermöglicht den Zugriff auf die Windows-Registrierung. Bei der Arbeit mit der Registrierung werden zwei wichtige Begriffe in der Dokumentation verwendet.

- Schlüssel – Eine Entität in der Registrierungsstruktur, die untergeordnete Schlüssel und Werte enthalten kann.
- Wert – Eine Entität ohne untergeordnete Einträge, die Daten beinhaltet. Es kann sich um unterschiedliche Datentypen handeln. Diese Implementierung unterstützt die Typen: Zeichenfolge, Ganzzahl und Binärdaten.

Alle Registrierungsvorgänge sind relativ zum Kontext, in dem das Skript ausgeführt wird. Es gibt eine Ausnahme: Wenn die Klasse mit dem Wert "Wahr" initialisiert wird, dann sind alle Vorgänge relativ zum Hostrechner von **Network Monitor**. Weitere Informationen finden Sie unter Beispiel-Skript zwei in dieser Klasse.

In diesem Kapitel

Beispiel-Skript: 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()

Hinweise

Diese Funktion muss vor dem ersten Abruf von EnumValue() abgerufen werden. Die Funktion stellt sicher, dass EnumValue() am Anfang der Werteliste beginnt. Wenn diese Funktion nicht vor EnumValue() aufgerufen wird, führt dies zu unvorhersehbaren Ergebnissen.

Close

Close()

Hinweise

Die Funktion schließt die aktuell geöffnete Verbindung mit Registrierung.

Create

bool Create(string sKey)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist. Andernfalls ist der Wert null und eine Fehlerbeschreibung kann durch Aufrufen von GetErrorDescription() abgerufen werden.

Parameter

- sKey – Ein Schlüssel, der erstellt wird.

Hinweise

Die Funktion Create() erstellt einen bestimmten Registrierungsschlüssel. Die Funktion öffnet den Schlüssel, wenn dieser in der Registrierung bereits existiert. Die Funktion kann verwendet werden, um in einem Schritt mehrere Schlüssel zu erstellen. Ein Skript kann einen Unterschlüssel erstellen, der drei Ebenen tief geht, indem eine Zeichenfolge im folgenden Format angegeben wird:

subkey1\subkey2\subkey3

DeleteValue

bool DeleteValue(string sValueName)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist. Andernfalls ist der Wert null und eine Fehlerbeschreibung kann durch Aufrufen von GetErrorDescription() abgerufen werden.

Parameter

- sValueName – Name des Werts, der gelöscht wird.

Hinweise

Die Funktion löscht einen Wert im aktuellen Schlüssel. Wenn dieser Wert nicht existiert, schlägt die Funktion fehl.

EnumValue

bool EnumValue(string &sValueName)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist. Andernfalls ist der Wert null und eine Fehlerbeschreibung kann durch Aufrufen von GetErrorDescription() abgerufen werden.

Parameter

- sValueName – Name des nächsten aufgezählten Werts im aktuellen Schlüssel.

Hinweise

Rufen Sie diese Funktion so lange auf, bis diese mit dem Wert "Falsch" zurückgegeben wird, um alle Werte im aktuellen Schlüssel aufzuzählen. Bevor diese Funktion zum ersten Mal aufgerufen wird, muss ein Aufruf von BeginEnumValue() ausgeführt werden.

Beispiel 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()

Rückgabewerte

Gibt eine Zeichenfolge zurück, die den letzten Fehler beschreibt, der beim Aufruf einer Funktion in der Klasse gefunden wurde.

Open

bool Open(int iKey,string sKey)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist. Andernfalls ist der Wert null und eine Fehlerbeschreibung kann durch Aufrufen von GetErrorDescription() abgerufen werden.

Parameter

- iKey – Ein Schlüssel, der eine der Registrierungsstrukturen darstellt.
- bCreate – Ein Unterschlüssel in der ausgewählten Registrierungsstruktur.

Hinweise

Die Funktion öffnet einen Registrierungsschlüssel in der ausgewählten Registrierungsstruktur. Beachten Sie, das die Anmeldeinformationen des Vorgangs (sowohl IDE als auch **Network Monitor**) den Zugriff auf bestimmte Schlüssel und Strukturen einschränken können. Die folgenden Konstanten sind für iKey definiert:

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

ReadValue

bool ReadValue(string sValueName,string &sData)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist. Andernfalls ist der Wert null und eine Fehlerbeschreibung kann durch Aufrufen von GetErrorDescription() abgerufen werden.

Parameter

- sValueName – Name des abzurufenden Werts.
- sData – Daten, die von der Funktion zurückgegeben werden.

Hinweise

Die Funktion gibt die Daten des Werts zurück, dessen Name angegeben wurde. Wenn der Werttyp nicht eine Zeichenfolge ist, schlägt die Funktion fehl.

ReadValue

bool ReadValue(string sValueName,int &iData)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist. Andernfalls ist der Wert null und eine Fehlerbeschreibung kann durch Aufrufen von GetErrorDescription() abgerufen werden.

Parameter

- sValueName – Name des abzurufenden Werts.
- iData – Daten, die von der Funktion zurückgegeben werden.

Hinweise

Die Funktion gibt die Daten des Werts zurück, dessen Name angegeben wurde. Wenn der Werttyp nicht eine Ganzzahl ist, schlägt die Funktion fehl.

ReadValue

string ReadValue(string sValueName,int &iSize)

Rückgabewerte

Daten, die im Registrierungswert gespeichert sind. Wenn die Funktion fehl schlägt, wird eine leere Zeichenfolge zurückgegeben. Es kann eine Fehlerbeschreibung durch Aufrufen von GetErrorDescription() abgerufen werden.

Parameter

- sValueName – Name des abzurufenden Werts.
- iSize – Größe der Daten, die von der Funktion in Byte zurückgegeben werden.

Hinweise

Die Funktion gibt die Daten des Werts zurück, dessen Name angegeben wurde. Wenn der Werttyp nicht eine Ganzzahl ist, schlägt die Funktion fehl. Die Größe der zurückgegebenen Daten wird im Parameter iSize gespeichert. Wenn diese Funktion fehlschlägt, wird der Parameter iSize auf null eingestellt.

SetValue

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

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist. Andernfalls ist der Wert null und eine Fehlerbeschreibung kann durch Aufrufen von GetErrorDescription() abgerufen werden.

Parameter

- sValueName – Name des zu schreibenden Werts.
- sData – In den Wert zu schreibende Daten.
- iSize – Größe der zu schreibenden Daten, in Byte.

Hinweise

Die Funktion schreibt die angegebenen Daten in den Wert.

SetValue

bool SetValue(string sValueName,string sString)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist. Andernfalls ist der Wert null und eine Fehlerbeschreibung kann durch Aufrufen von GetErrorDescription() abgerufen werden.

Parameter

- sValueName – Name des zu schreibenden Werts.
- sString – In den Wert zu schreibende Zeichenfolge.
- iSize – Größe der zu schreibenden Daten, in Byte.

Hinweise

Die Funktion schreibt die angegebene Zeichenfolge in den Wert. Wenn dieser Wert nicht existiert, schlägt die Funktion fehl.

SetValue

bool SetValue(string sValueName,int iValue)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist. Andernfalls ist der Wert null und eine Fehlerbeschreibung kann durch Aufrufen von GetErrorDescription() abgerufen werden.

Parameter

- sValueName – Name des zu schreibenden Werts.
- iValue – In den Wert zu schreibende Ganzzahl.

Hinweise

Die Funktion schreibt die angegebene Ganzzahl in den Wert. Wenn dieser Wert nicht existiert, schlägt die Funktion fehl.

SetValueExpandedString

bool SetValueExpandedString(string sValueName,string sString)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist. Andernfalls ist der Wert null und eine Fehlerbeschreibung kann durch Aufrufen von GetErrorDescription() abgerufen werden.

Parameter

- sValueName – Name des zu schreibenden Werts.
- sString – In den Wert zu schreibende Zeichenfolge.

Hinweise

Diese Funktion funktioniert wie die übliche Funktion SetValue, mit einer Ausnahme. Die geschriebene Zeichenfolge kann nicht erweiterte Verweise auf Umgebungsvariablen (z. B. %PATH%) enthalten.

Kapitel 16

TLuaSFTPClient

Die Klasse implementiert eine grundlegende SFTP-Clientklasse.

In diesem Kapitel

Beispiel-Skript: 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)
```

Close

bool Close(TLuaSFTPClientHandle FileHandle)

Rückgabewert

Wenn die Funktion erfolgreich war, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch".

Parameter

- FileHandle – Handle einer bereits geöffneten Datei.

CloseDir

bool CloseDir(TLuaSFTPClientDirectoryHandle Handle);

Rückgabewert

Wenn die Funktion erfolgreich war, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch". Bei einem erfolgreichen Vorgang wird das Handle TLuaSFTPClientDirectoryHandle geschlossen.

Parameter

- Handle – Handle wurde durch die Funktion OpenDir geöffnet.

Connect

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

Rückgabewert

Bei erfolgreicher Verbindung wird "Wahr" zurückgegeben und andernfalls "Falsch".

Parameter

- sUsername – Benutzername
- sPassword – Passwort
- iPort – Portnummer, die der Server überwacht. Der Standardwert ist 22.
- iTimeout – Zeitlimit in Millisekunden, in dem auf eine Serverantwort gewartet wird. Der Standardwert ist 25.000 (25 Sekunden).

CreateFile

bool CreateFile(string sPath,TLuaSFTPClientHandle hHandle)

Rückgabewert

Wenn die Datei erstellt wurde, wird der Wert als "Wahr" zurückgegeben und als "Falsch", wenn der Vorgang fehlgeschlagen ist. Das Handle TLuaSFTPClientHandle enthält einen Verweis zum Öffnen, wenn der Vorgang erfolgreich war.

Parameter

- sPath – Vollständiger Pfad der zu erstellenden Datei. Die im Pfad enthaltenen Verzeichnisse müssen existieren, damit der Vorgang erfolgreich ausgeführt werden kann.
- hHandle – Handle, um eine Datei zu erstellen.

Hinweise

Die neu erstellte Datei hat Lese- und Schreibzugriffsrechte.

ListDir

bool ListDir(TLuaSFTPClientDirectoryHandle Handle);

Rückgabewert

Wenn die Funktion erfolgreich war, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch". Bei einem erfolgreichen Vorgang stehen Daten zum Abrufen in der Klasse TLuaSFTPClientDirectoryHandle bereit.

Parameter

- Handle – Handle wurde durch die Funktion OpenDir geöffnet.

MkDir

bool MkDir(string sPath)

Rückgabewert

Wenn die Funktion erfolgreich war, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch".

Parameter

- sPath – Pfad zum Verzeichnis, das erstellt wird, einschließlich Name des neuen Verzeichnisses.

Hinweise

Diese Funktion kann rekursiv keine neuen Direktiven erstellen. Es müssen alle Verzeichnisse des letzten Verzeichnisses im Pfad existieren.

OpenDir

bool OpenDir(string sPath,TLuaSFTPClientDirectoryHandle &Handle)

Rückgabewert

Wenn die Funktion erfolgreich war, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch".

Parameter

- sPath – Pfad zum Verzeichnis, das geöffnet wird.
- Handle – Zurückgegebenes Handle, in weiteren Vorgängen zu verwenden.

Hinweise

Diese Funktion "öffnet" ein Verzeichnis, um dessen Inhalt mit der Funktion ListDir aufzulisten.

Open_ForRead

bool Open_ForRead(string _sPath,TLuaSFTPClientHandle hHandle)

Rückgabewert

Wenn die Datei erfolgreich geöffnet wurde, wird der Wert als "Wahr" zurückgegeben und als "Falsch", wenn der Vorgang fehlgeschlagen ist.

Parameter

- sPath – Vollständiger Pfad der Datei.
- hHandle – Handle für die Datei, das in weiteren Vorgängen verwendet wird.

Open_ForWrite

bool Open_ForWrite(string _sPath,TLuaSFTPClientHandle hHandle)

Rückgabewert

Wenn die Datei erfolgreich geöffnet wurde, wird der Wert als "Wahr" zurückgegeben und als "Falsch", wenn der Vorgang fehlgeschlagen ist.

Parameter

- sPath – Vollständiger Pfad der Datei.
- hHandle – Handle für die Datei, das in weiteren Vorgängen verwendet wird.

Open_ForAppend

bool Open_ForAppend(string _sPath,TLuaSFTPClientHandle hHandle)

Rückgabewert

Wenn die Datei erfolgreich geöffnet wurde, wird der Wert als "Wahr" zurückgegeben und als "Falsch", wenn der Vorgang fehlgeschlagen ist.

Parameter

- sPath – Vollständiger Pfad der Datei. hHandle Handle zur Datei, das in weiteren Vorgängen verwendet wird.

Hinweise

Open_ForAppend wird im Schreibmodus geöffnet. Der Unterschied zwischen dieser Funktion und Open_ForWrite ist, dass alle Daten in das Ende der Datei geschrieben werden, auch wenn der Dateizeiger zwischen zwei Schreibvorgänge positioniert wird.

Read

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

Rückgabewert

Wenn die Funktion erfolgreich war, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch".

Parameter

- FileHandle – Handle einer bereits geöffneten Datei.
- iOffset – Offset in Byte, in Bezug auf die Leseposition in der Datei.
- iLen – Länge der zu lesenden Daten
- sData – Variable, in die Daten eingefügt werden.

Hinweise

Mit dieser Funktion können nur Textdateien gelesen werden.

Beispiel

```
--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)
```

Remove

bool Remove(string sPath);

Rückgabewert

Wenn die Funktion erfolgreich war, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch".

Parameter

- sPath – Pfad der zu entfernenden Datei.

Rename

bool Rename(string sPath,string sNewPath);

Rückgabewert

Wenn die Funktion erfolgreich war, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch".

Parameter

- sPath – Pfad der bestehenden Datei, die umbenannt wird.
- sNew – Pfad mit neuem Dateinamen.

RmDir

bool RmDir(string sPath)

Rückgabewert

Wenn die Funktion erfolgreich war, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch".

Parameter

- sPath – Pfad zum Verzeichnis, das gelöscht wird.

Hinweise

Diese Funktion kann nur leere Verzeichnisse löschen.

Write

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

Rückgabewert

Wenn die Funktion erfolgreich war, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch".

Parameter

- FileHandle – Handle einer bereits geöffneten Datei.
- iOffset – Offset in Byte, wo die Datei geschrieben wird.
- sData – Zeichenfolge des zu schreibenden Texts.

Beispiel

```
--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)
```

Kapitel 17

TLuaSFTPClientAttributes

Die Klasse enthält Attribute, die ein Verzeichnis oder eine Datei beschreiben, die durch die Funktion ListDir abgerufen wurden.

In diesem Kapitel

AccessedTime

bool AccessedTime(TLuaDateTime &Time)

Rückgabewert

Wenn der Wert vorhanden ist, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch".

Parameter

- Time – Enthält den Zeitpunkt, zu dem zuletzt auf die Datei zugegriffen wurde.

CreatedTime

bool CreatedTime(TLuaDateTime &Time)

Rückgabewert

Wenn der Wert vorhanden ist, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch".

Parameter

- Time – Enthält den Zeitpunkt, zu dem die Datei erstellt wurde.

Group

bool Group(string &sGroup)

Rückgabewert

Wenn der Wert vorhanden ist, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch".

Parameter

- sOwner – Enthält den Namen der Dateigruppe oder des Verzeichnisses.

ModifiedTime

bool ModifiedTime(TLuaDateTime &Time)

Rückgabewert

Wenn der Wert vorhanden ist, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch".

Parameter

- Time – Enthält den Zeitpunkt, zu dem die Datei zuletzt geändert wurde.

Owner

bool Owner(string &sOwner)

Rückgabewert

Wenn der Wert vorhanden ist, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch".

Parameter

- sOwner – Enthält den Namen des Besitzers der Datei oder des Verzeichnisses.

PermissionBits

bool PermissionBits(int &iPermissionsBits)

Rückgabewert

Wenn der Wert vorhanden ist, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch".

Parameter

- iPermissionsBits – Enthält einen Dezimalwert, der die Berechtigung der Datei oder des Verzeichnisses darstellt.

Size

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

Rückgabewert

Wenn der Wert vorhanden ist, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch".

Parameter

- iBytesHighDWord – Enthält den oberen Dword-Anteil der 64-Bit-Ganzzahl.
- iBytesLow DWord – Enthält den unteren Dword-Anteil der 64-Bit-Ganzzahl.

Hinweise

Die Dateigröße wird in Byte als 64-Bit-Ganzzahl gemeldet. Da Lua keine 64-Ganzzahl-Datentypen hat, wurden die Daten in zwei 32-Bit-Ganzzahlen unterteilt. Wenn die Datei kleiner als 2 GB ist, ist iBytesHighDWord immer null.

SizeMB

bool SizeMB(unsigned int &iSizeMB);

Rückgabewert

Wenn der Wert vorhanden ist, wird der Wert als "Wahr" zurückgegeben und andernfalls als "Falsch".

Parameter

- iSizeMB – Enthält die Dateigröße in Megabytes.

Hinweise

Es handelt sich um eine benutzerfreundliche Alternative zur Funktion Size(), die die Dateigröße abgerundet zurückgibt.

Kapitel 18

TLuaSFTPClientDirectoryHandle

Diese Klasse wird in Verbindung mit den Funktionen OpenDir, ListDir und CloseDir verwendet.

In diesem Kapitel

Next

bool Next(TLuaSFTPClientFile &hFile)

Rückgabewert

Der zurückgegebene Wert ist "Wahr", wenn Daten in der zur Verfügung gestellten TLuaSfPTClientFile enthalten sind.

Hinweise

Führen Sie die Funktion in einer Schleife aus, bis sie den Wert "Falsch" zurückgibt. So werden alle zurückgegebenen Daten der Funktion ListDir abgerufen.

Kapitel 19

TLuaSFTPClientFile

TLuaSFTPClientFile ist eine schreibgeschützte Klasse.

Klassenmitglieder

- string m_sFilename
- string m_sLongFilename
- TLuaSFTPClientAttributes m_Attribs

Kapitel 20

TLuaSNMP

Die Klasse implementiert einen grundlegenden SNMP-Client, der SET- und GET-Vorgänge ausführen kann.

In diesem Kapitel

Beispiel-Skript: 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)

Parameter

- sOID – Ein OID, das den Beginn eines OID Tree Walk darstellt. Beispiel für ein OID: iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable

Hinweise

Bevor die Funktion Walk zum ersten Mal aufgerufen wird, muss das Programm die Funktion BeginWalk aufrufen, um den Beginn für Walk festzulegen. Walk ruft alle untergeordneten und gleichgeordneten Bestandsbezeichner des Start-OIDs auf, der von den Funktionen BeginWalk festgelegt wurde.

Close

Close()

Hinweise

Schließt die SNMP-Verbindung.

Get

string Get(string sOID,int iSyntax)

Rückgabewerte

Eine Zeichenfolge mit dem vom Remote-SNMP-Agent abgerufenen Wert.

Parameter

- sOID – In GET-Vorgängen zu verwendender OID. Wenn eine Schnittstelle angefragt wird, kann @ user zum Festlegen der Schnittstellenliste verwendet werden.

  Beispiel für die Verwendung von @ user:

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

  Beispiel für einen üblichen OID

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

- iSyntax – Legt das Format der zurückgegebenen Daten fest. Eine der folgenden Konstanten ist möglich.
- SNMP_NOSYNTAX
- SNMP_IPADDRESS
- SNMP_INTEGER
- SNMP_UNSIGNED32
- SNMP_COUNTER32
- SNMP_GAUGE32
- SNMP_TIMETICKS
- SNMP_OPAQUE
- SNMP_OCTETSTRING
- SNMP_DATA_AS_HEXSTRING

Lesen von Binärwerten

Einige OID geben eventuell Binärdaten statt z. B. einer Zeichenfolge oder Ganzzahl zurück. Dies kann ein Problem darstellen, da die Funktion Get eine auf null endende Zeichenfolge zurückgibt. Das Problem lässt sich lösen, indem die Variable iSyntax auf SNMP_DATA_AS_HEXSTRING eingestellt wird. Die Funktion gibt anschließend die Binärdaten hexadezimalcodiert zurück.

Beispiel für drei hexadezimalcodierte Byte

```
49 4E4D
```

Open

bool Open(string sCommunity, int iPort=161)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sCommunity – Name der Community, in der Regel öffentlich
- iPort (Optional) Legen Sie die Portnummer fest, wenn Sie einen anderen Port als den Standardport (Port 161) verwenden müssen.

Set

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

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null.

Parameter

- sOID – In SET-Vorgängen zu verwendender OID.
- sData – Textdaten, die in SET-Vorgängen verwendet werden.
- iSyntax – Legt das Format des Parameters sData fest. Eine der folgenden Konstanten ist möglich.
  - ✓ SNMP_NOSYNTAX
  - ✓ SNMP_IPADDRESS
  - ✓ SNMP_INTEGER
  - ✓ SNMP_UNSIGNED32
  - ✓ SNMP_COUNTER32
  - ✓ SNMP_GAUGE32
  - ✓ SNMP_TIMETICKS
  - ✓ SNMP_OPAQUE
  - ✓ SNMP_OCTETSTRING

Walk

TLuaSNMPResult TLuaSNMP::Walk()

Rückgabewerte

Gibt den Bezeichner des nächsten OID in der Membervariable m_sOID in der Struktur TLuaSNMPResult zurück. Gibt nicht die vollständige OID-Zeichenfolge zurück. Wenn das Ende erreicht wird, ist Member m_sOID der Struktur TLuaSNMPResult leer.

Hinweise

Bevor die Funktion Walk zum ersten Mal aufgerufen wird, muss das Programm die Funktion BeginWalk aufrufen, um den Beginn für Walk festzulegen. Walk ruft alle untergeordneten Bezeichner und die Objektbezeichner des Start-OIDs auf, der von den Funktionen BeginWalk festgelegt wurde.

Beispiel

```
--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 ist eine schreibgeschützte Klasse, die von der Funktion Walk zurückgegeben wird. Eine Änderung löst eine Ausnahme aus.

Klassenmitglieder

- string m_sOID
- string m_sData
- int m_iSyntax

Kapitel 21

TLuaSSH2Client

Die Klasse implementiert einen 2.0 SSH-Client, der auf einem Remoteserver Befehle ausführen kann.

In diesem Kapitel

Beispiel-Skript: 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*/)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sCommand – Zeichenfolge mit dem Befehl, der auf dem Remotehost ausgeführt wird.
- dwWait – (Optional) Wartezeit für die Beendigung der Ausführung, Standard 25 Sekunden.

GetErrorDescription

string GetErrorDescription(void)

Rückgabewerte

Gibt die letzte Fehlerbeschreibung zurück, die vom Client als Zeichenfolge erzeugt wurde.

GetStdErr

string GetStdErr(void)

Rückgabewerte

Gibt die Standardfehlerausgabe vom Remotehost zurück.

GetStdOut

string GetStdOut(void)

Rückgabewerte

Gibt die Standardausgabe vom Remotehost zurück.

Open

bool Open(int iPort)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Aufrufen der Funktion GetErrorDescription abgerufen werden. Wenn der Fehler ein Ergebnis des Befehls ist, können weitere Informationen durch Abruf von GetStdErr abgerufen werden.

Parameter

- iPort – SSH-Port, Standard 23.
- sUsername – Benutzername
- sPassword – Passwort

Kapitel 22

TLuaSocket

Diese Klasse stellt grundlegende Socketvorgänge zur Verfügung. Sockets können sowohl im UDP- als auch im TCP-Modus geöffnet werden.

In diesem Kapitel

Beispiel-Skript: 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()
```

Close

int Close()

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Hinweise

Schließt das Socket, das bereits mit OpenTCP oder OpenUDP geöffnet wurde.

OpenTCP

int OpenTCP(int iPort)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- iPort – Der zu öffnende Port.

Hinweise

Öffnet einen TCP-Socket unter Verwendung einer angegebenen Portnummer.

OpenUDP

int OpenUDP(int iPort)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- iPort – Ein bestimmter Port, der mit dem Socket verwendet wird.

Hinweise

Öffnet einen UDP-Socket unter Verwendung einer angegebenen Portnummer.

Read

string Read(int iSize,int iTimeout=1)

Rückgabewerte

Ein Datenarray, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null, wenn keine Daten gelesen werden konnten. Ein bestimmter Fehlercode kann durch Aufrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- iSize – Wenn der Aufruf einer Funktion zurückkehrt, wird die Variable auf die Größe der gelesenen Daten festgelegt. Wenn keine Daten gelesen wurden, ist dieser Wert null.
- iTimeout – Die Wartezeit in Sekunden bis die Daten am Socket eintreffen. Der Standardwert ist eine Sekunde.

Hinweise

Die Funktion blockiert die Ausführung nur für den Zeitraum, der im Zeitlimit festgelegt ist. Wenn keine Daten in diesem Zeitraum empfangen werden, gibt die Funktion den Wert null zurück.

Write

int Write(string Data,int iSize)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sData – Ein Array mit den zu versendenden Daten.

- iSize – Die Größe der Daten im Array.

Kapitel 23

TLuaSocketSecure

Die Klasse stellt grundlegende sichere Socketvorgänge zur Verfügung, die gemeinhin als Transport Layer Security (TLS) oder als dessen Vorgänger Secure Sockets Layer (SSL) bezeichnet werden.

In diesem Kapitel

Beispiel-Skript: 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
```

Open

int Open(int iPort)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- iPort – Der zu öffnende Port

Close

int Close()

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Read

string Read(int iSize)

Rückgabewerte

Ein Datenarray, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null, wenn keine Daten gelesen werden konnten. Ein bestimmter Fehlercode kann durch Aufrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- iSize – Wenn der Aufruf einer Funktion zurückkehrt, wird die Variable auf die Größe der gelesenen Daten festgelegt. Wenn keine Daten gelesen wurden, ist dieser Wert null.

Write

int Write(string Data,int iSize)

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null und ein bestimmter Fehlercode kann durch Abrufen der globalen Funktion GetLastError abgerufen werden.

Parameter

- sData – Ein Array mit den zu versendenden Daten.
- iSize – Die Größe der Daten im Array.

GetCertificateExpiryDate

TLuaDateTime GetCertificateExpiryDate()

Rückgabewerte

Eine TLuaDateTime-Struktur, die das Ablaufdatum des Zertifikats des Remotehosts enthält. Wenn der Aufruf Connect() fehlschlägt, enthält die Struktur das Datum null.

Hinweise

Diese Funktion kann verwendet werden, um zu bestimmen, ob ein Zertifikat bald ablaufen wird oder bereits abgelaufen ist.

Kapitel 24

TLuaStorage

Mit dieser Klasse können Textdaten zwischen Skriptsitzungen gespeichert werden. Dies kann sinnvoll sein, wenn Sie eine aktuelle Skriptiteration auf ein vorhergehendes Ergebnis gründen wollen oder wenn Sie zwischen zwei unabhängigen Skripten kommunizieren möchten.

In diesem Kapitel

CreateItem

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

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null.

Parameter

- sName – Eindeutiger Elementname. Wenn der Name bereits erstellt wurde, schlägt diese Funktion fehl.
- sKey – Schlüsselname des Elements, der eindeutig sein muss. Wenn er bereits existiert, schlägt die Funktion fehl.
- sData – Optionale Daten, die mit dem Element verknüpft werden
- iSize – Datengröße. Nur erforderlich, wenn mit dieser Funktion Daten zur Verfügung gestellt werden.

Hinweise

Die Funktion erstellt ein Element und ein untergeordnetes Element, der als "Schlüssel" bezeichnet wird. Der Benutzer kann mit diesem Schlüssel Daten verknüpfen. Die Daten können zu einem späteren Zeitpunkt über die Funktion FindItem erworben werden.

UpdateItem

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

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null.

Parameter

- sName – Eindeutiger Name des Elements. Ein Element mit diesem Namen muss bereits vorhanden sein.
- sKey – Schlüsselname des Elements. Ein Schlüssel mit diesem Namen muss bereits vorhanden sein.
- sData – Optionale Daten, die mit dem Element verknüpft werden. Die Daten ersetzen die aktuell im Element gespeicherten Daten (falls vorhanden).
- iSize – Datengröße. Nur erforderlich, wenn mit dieser Funktion Daten zur Verfügung gestellt werden.

Hinweise

Die Funktion aktualisiert ein bereits erstelltes Element. Wenn die Element/Schlüssel-Kombination nicht existiert, schlägt diese Funktion fehl.

DeleteItem

void DeleteItem(string sName,string sKey);

Parameter

- sName – Name des Elements.

- sKey – Name des zu löschenden Schlüssels.

Hinweise

Die Funktion löscht eine Element/Schlüssel-Kombination. Mit dem Schlüssel verknüpfte Daten werden ebenfalls gelöscht.

FindItem

TLuaStorageItem FindItem(string sName,string sKey);

Rückgabewerte

Der Wert ist ungleich null, wenn die Funktion erfolgreich ist, andernfalls ist der Wert null.

Parameter

- sName – Eindeutiger Name des Elements. Ein Element mit diesem Namen muss bereits vorhanden sein.
- sKey – Schlüsselname des Elements. Ein Schlüssel mit diesem Namen muss bereits vorhanden sein.

Hinweise

Die Funktion ruft ein gespeichertes Element ab. Die zurückgegebene Klasse enthält die Element-/Schlüsselnamen sowie die mit dem Element verknüpften Daten.

TLuaStorageItem

TLuaStorageItem ist eine schreibgeschützte Klasse. Eine Änderung löst eine Ausnahme aus.

Klassenmitglieder

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

Kapitel 25

TLuaTimer

Die Klasse stellt einen Timer zur Verfügung, der auf die Millisekunde genau ist.

In diesem Kapitel

Beispiel-Skript: TLuaTimer

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

Start

Start()

Hinweise

Die Funktion startet die Zeit. Ein weiterer Aufruf der Funktion Stop() gibt die Zeit zwischen den Aufrufen der Funktionen Start und Stop zurück. Rufen Sie nach dem Aufruf von Stop erneut Start auf, um den Timer zurückzusetzen und einen neuen Zeitraum zu beginnen.

Stop

int Stop()

Rückgabewerte

Gibt die Anzahl der Millisekunden zurück, die seit dem Aufruf der Funktion Start() vergangen sind.

Kapitel 26

TLuaWinperf

Die Klasse stellt Funktionen zur Abfrage von numerischen Werten im Windows-Leistungsregister zur Verfügung. Es wird als benutzerfreundliche Alternative zur erweiterten Klasse TLuaWMIQuery zur Verfügung gestellt. Die Klasse wird im Sicherheitskontext des Prozesses oder im Thread, welches das Skript gestartet hat, ausgeführt. In IDE wird der Sicherheitskontext vom Desktop vererbt. Bei Ausführung durch Lua-Skriptmonitoring kann der Sicherheitskontext festgelegt werden, indem ein Standardkonto in der Monitor-Eigenschaftenseite ausgewählt wird.

In diesem Kapitel

Beispiel-Skript: 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()

Rückgabewerte

Gibt eine Zeichenfolge zurück, die den letzten Fehler beschreibt, der beim Aufruf einer Funktion in der Klasse gefunden wurde.

GetResult

double GetResult()

Rückgabewerte

Gibt einen numerischen Leistungsindikatorwert zurück. Wenn die vorherige Abfrage von Query() fehlgeschlagen ist, gibt diese Funktion null zurück.

Query

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

Rückgabewerte

Wahr: die Abfrage wurde erfolgreich ausgeführt. Falsch: ein Fehler ist aufgetreten.

Parameter

- sDeviceName – Eine Zeichenfolge mit dem Namen des Bestands, der den abzufragenden Leistungsindikator enthält.
- sCounterName – Eine Zeichenfolge mit dem Namen des abzufragenden Leistungsindikators
- sInstanceName – (Optional) Eine Zeichenfolge mit dem Namen des Leistungsindikators.

Hinweise

Bestands-, Leistungsindikator- und Instanznamen können sowohl durch Klicken auf die Aufzählungsschaltfläche des Monitors Winperf von **Network Monitor** erhalten werden oder über die Windows-Anwendung perfmon.exe. Rufen Sie, nachdem die Funktion beendet wurde, GetResult() auf, um den Wert abzurufen.

Kapitel 27

TLuaWMIQuery

Die Klasse stellt Funktionen zur Verfügung, um WMI-Eigenschaften abzufragen. Die Klasse wird im Sicherheitskontext des Prozesses oder im Thread, welches das Skript gestartet hat, ausgeführt. In IDE wird der Sicherheitskontext vom Desktop vererbt. Bei Ausführung durch Lua-Skriptmonitoring kann der Sicherheitskontext festgelegt werden, indem ein Standardkonto in der Monitor-Eigenschaftenseite ausgewählt wird. Das Konto muss für die Delegierung aktiviert werden.

In diesem Kapitel

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
```

Execute

bool Execute(const char *pWQL)

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

Rückgabewerte

Wahr: die Abfrage wurde erfolgreich ausgeführt. Falsch: ein Fehler ist aufgetreten.

Parameter

- sWQL – Eine Zeichenfolge, die eine WQL-Abfrage enthält.

Hinweise

Führt eine WQL-Abfrage (WMI Query Language) aus. Die Abfragen Next() und GetProperty() können verwendet werden, um das Ergebnis heranzuholen.

GetErrorDescription

string GetErrorDescription()

Rückgabewerte

Gibt eine Zeichenfolge zurück, die den letzten Fehler beschreibt, der beim Aufruf einer Funktion in der Klasse gefunden wurde. Sinnvoll beim Debuggen einer WMI-Abfrage.

GetProperty

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

Rückgabewerte

Wenn die Funktion erfolgreich war, werden der Wert als "Wahr" und ein Wert in einer Zeichenfolge zurückgegeben. Wenn die Funktion fehlschlägt, wird der Wert als "Falsch" zurückgegeben und die Zeichenfolge ist leer. Weitere Informationen zum Fehler können durch den Aufruf von GetError() abgerufen werden.

Parameter

- sPropertyName – Name der abzufragenden Eigenschaft.

- sReturnValue – Definierte Zeichenfolge, die den Rückgabewert empfängt. Der Rückgabewert ist immer eine Zeichenfolge, auch wenn der Eigenschaftstyp z. B. eine Ganzzahl oder eine reelle Zahl ist.

Hinweise

Fragt einen Eigenschaftswert im aktuellen Ergebnis ab. Rufen Sie die Funktion NextInstance() auf, um den nächsten Wert derselben Eigenschaft abzurufen. Wenn NextInstance() als falsch zurückgegeben wird, gibt es keine weiteren Werte.

NextInstance

bool NextInstance()

Rückgabewerte

Wahr: ein neues Ergebnis wurde abgerufen. Falsch: es gibt keine weiteren Ergebnisse für die Abfrage.

Hinweise

Die Funktion ruft ein neues Ergebnis ab, dass bei einem bereits durchgeführten Abruf der Execute-Funktion erzeugt wurde. Diese Funktion muss vor dem ersten Abruf der Funktion GetProperty abgerufen werden.

SetNamespace

SetNamespace(string sNamespace)

Parameter

- sNamespace – Zeichenfolge mit WMI-Namespace, die in allen zukünftigen Aufrufen zu verwenden ist.

Hinweise

Der standardmäßige Namespace, der von der Klasse TLuaWMIQuery verwendet wird lautet root\cvmi2.

Kapitel 28

TLuaXMLNode

Die Klasse stellt ein XML-Element dar und kann ein oder mehrere untergeordnete Elemente enthalten.

In diesem Kapitel

FindAttribute

string FindAttribute(string sName)

Rückgabewerte

Die Funktion gibt eine Zeichenfolge mit dem Wert des Attributs zurück. Wenn das Attribut nicht auffindbar ist, ist die zurückgegebene Zeichenfolge leer.

Parameter

- sName – Der Name des Attributs.

FindChildNode

TLuaXMLNode FindChildNode(string sElementName, int iOffset)

Rückgabewerte

Die Funktion gibt einen gültigen Bestand vom Typ TLuaXMLNode zurück, wenn das Element gefunden wurde.

Parameter

- sElementName – Der Name des untergeordneten Elements in diesem Knoten.
- iOffset – Ein nullbasierter Index, um untergeordnete Elemente mit demselben Namen im Knoten abzufragen.

Hinweise

Die Funktion kann verwendet werden, um eine Anzahl an untergeordneten Elementen mit demselben Namen zu durchlaufen. Erhöhen Sie den Versatzparameter, um das nächste Element heranzuholen.

GetData

string GetData()

Rückgabewerte

Die Funktion gibt die Daten im Element zurück.

GetTag

string GetTag()

Rückgabewerte

Die Funktion gibt den Tagnamen des Elements zurück.

GetParentNode

TLuaXMLNode GetParentNode()

Rückgabewerte

Die Funktion gibt das übergeordnete Element des aktuellen XML-Dokumentelements zurück.

IsValid

bool IsValid()

Rückgabewerte

Die Funktion wird bei gültigen Knoten mit dem Wert "Wahr" zurückgegeben und als "Falsch", wenn der Knoten ungültig ist.

Hinweise

Alle Suchfunktionen geben einen Bestand vom Typ TLuaXMLNode zurück; die Funktion IsValid() wird verwendet, um zu bestimmen, ob die Suche erfolgreich war.

Kapitel 29

TLuaXMLReader

Die Klasse bietet eine grundlegende Funktionalität, um XML-Dokumente zu analysieren und zu durchsuchen.

In diesem Kapitel

FindChildNode

TLuaXMLNode FindChildNode(string sElementName, TLuaXMLNode ParentNode)

Rückgabewerte

Die Funktion gibt einen gültigen Bestand vom Typ TLuaXMLNode zurück, wenn das Element gefunden wurde.

Parameter

- sElementName – Der Name des untergeordneten Elements von ParentNode.
- ParentNode – Der zu suchende übergeordnete Knoten.

Hinweise

Beachten Sie, dass die Funktion das erste Element mit dem angegebenen Namen zurückgibt.

FindNode

TLuaXMLNode FindNode(string sElementName, TLuaXMLNode RootNode)

Rückgabewerte

Die Funktion gibt einen gültigen Bestand vom Typ TLuaXMLNode zurück, wenn das Element gefunden wurde.

Parameter

- sElementName – Der Name des untergeordneten Elements von ParentNode.
- RootNode – Der übergeordnete Knoten, von dem aus die Suche beginnt.

Hinweise

Die Funktion sucht das XML-Dokument rekursiv mit RootNode als Ausgangspunkt der Suche.

FromXML

bool FromXML(string XML)

Rückgabewerte

Wahr: Der Vorgang wurde erfolgreich ausgeführt. Falsch: Ein Fehler ist aufgetreten.

Parameter

- sXML – Ein zu analysierendes XML-Dokument.

Hinweise

Beachten Sie, dass der Parser das Dokument nicht als Schema analysiert.

GetRootNode

TLuaXMLNode GetRootNode()

Rückgabewerte

Die Funktion gibt das Stammelement des XML-Dokuments zurück.

Inhaltsverzeichnis

U

W