PIASe Server Programming

The PI Active Server engine enables you to write dynamic script. While static HTML is good to present vast amounts of static information, it can't support the building of HTML blocks based on decisive information submitted to the server either by user interaction or information extracted from database tables for example.

When we started to develop the server back in 1999, we named the project PI Server, hence its name PI. PIASe server programming does not only allow you to program dynamic HTML, it also enables you to write scripts at protocol level (i.e. at the very moment that a connection has been established). In fact, without the protocol level scripts there would be no HTTP server at all.

PIASe may look like classic ASP since we use VBScript and the same tags to embed server-side script into HTML but it isn't classic ASP. ASP is limited concerning file / user security and accessing sources outside the virtual environment and does not allow any modifications at the protocol level. PIASe does not have these limitations.

Events

PIASe Server events are programmed Microsoft's Visual Basic procedures and are called by the server when a client is connecting, requesting a HTTP resource or wants to deliver UDP data-packets. The server will also call event procedures when the UDP timer elapsed, on starting, on stopping, on creation of sessions or when terminating sessions.

Why ?

We have programmed a default HTTP 1.0 compatible set of scripts to deliver you a functional server. You may want to fix problems or bugs, add new, customize or improve existing functionality and that is why we moved some closed source 'core' to external script files to enable you to obtain insights in how the server works at protocol level and most of all, create the possibility to actually change the normally closed source core.

Where ?

The event script files can be found in the */admin/server folder*.

Programming

We do not recommend core coding to novice VB programmers. If you're not familiar with the TCP, UDP and HTTP protocol you may do worse than better and you may compromise the functionality of the server. Core files have a **OS** file extension and cannot be retrieved as a web page using a HTTP client.

Server-side scripts (Scripts that generate dynamically web pages) have a **SSC** extension and incorrect changes made to **SSC** files may cause runtime error but will not compromise the functionality of the server.

What can I program ?

Dynamic web pages, Proxies, Email clients, Peer to Peer sharing programs, Spoken notification programs, Network scan utilities, Web spiders, Relay utilities and much more.

Because Visual Basic implements the OLE server interface to other libraries such a Word, Excel, Outlook and many more office oriented platforms you are able to integrate the server as a non complex stand alone HTTP / UDP server as part of you favorite Visual Basic, VBS and VBA development platform.

The PIASe library

We recommend you to master the PIASe library and server-side script programming first before you consider **OS** programming. Both the language and the library are not difficult to understand but good knowledge about the both prevents reinstall of **OS** files, un-authorized access, damage to your file-system or loss of data.

Pre-processor

Whenever you change an event script file or server-side script file it will be reloaded and if needed pre-processed by the server. You don't have to restart the server to get the modifications in effect when you have changed a script. There is one exception, The **OnServerStart** event will only be executed at startup and to test changes made in the **OnServerStart** event procedure you must restart the server.

Warning

Do not accept event-script files or server-side script files from third parties unless you're absolutely sure that the source can be trusted.

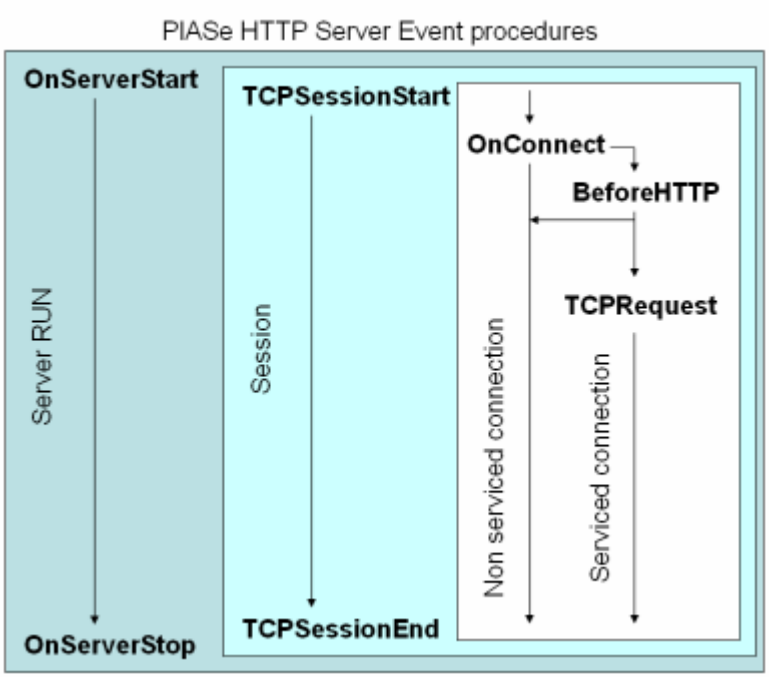
Ready ?

An important thing is to know about event procedures. There are two types of event procedures:

HTTP event procedures that are triggered by incoming requests that are serverd by the web server engine. **UDP event procedures** that are triggered by incoming UDP packets or the UDP event timer serving (optionally) UDP output streams.

The Core part I - TCP

Implementation chart of the HTTP Server Event procedures



Pure Visual Basic

Server Event procedures are programmed in pure Visual Basic ONLY. HTML may exist in pure Visual Basic script ONLY as part of a string expression. Script tag's are forbidden when programming OS scripts.

Firewall interaction

The built-in firewall is called before triggering the **OnConnect** event. Connections discarded by the firewall never trigger the OnConnect event.

Event procedures

location: /admin/server/startstop.os

Sub OnServerStart()

Occurs when the server is started and may be used to create server objects, database connections, etc.

Sub OnServerStop()

Occurs when the server is shutting down and may be used to close database connections and other clean-up tasks.

Location: /admin/server/tcpserver.os

Sub TCPRequest(Index)

Occurs whenever a client is requesting for a resource trough the HTTP protocol. The **Index** parameter addresses the offset in the connection object array that the connection is currently using.

The connection object array contains all input, output and calculated statistical data of the request and is of great importance when retrieving the request and preparing the HTTP output.

This event is already programmed by us and supports the handling of (authorized) HTTP 1.0 compatible requests of (non-encrypted) pages, images and binary objects. The script is also supporting the Base64 and MD5 authentication methods. Modifying this script may cause server malfunction when not handled correctly and requires advanced programming skills.

Sub TcpSessionStart(Id)

Occurs whenever a client is starting a new session. When a client is requesting a resource (any HTTP object) for the first time a so called session cookie will be created. When the session cookie is created successfully the server will call this event. The value of the created session cookie is passed via the **Id** parameter and uniquely identifies a session. This event usually contains initialization code such as the creation of a database connection.

A session is only maintained by the server when a sequence of two or more requests from the same client appears within a certain amount of time (TTL or Time To Live). The server will call the **TcpSessionEnd** event and cleanup session data when the TTL time expires.

Sub TcpSessionEnd(Id)

Occurs whenever a session is expired and may be used to clean-up server objects, memory and temporary files. The **Id** parameter uniquely identifies the expired session.

Location: /admin/server/firewall.os

Sub OnConnect(IP)

This event occurs when a client has connected to the TCP port from the listening HTTP server. The **IP** parameter returns the IP address of the connected client. At the moment of the **OnConnect** event no HTTP request is processed by the server and this event may only be used to determine a valid IP address and if not, a forced disconnection my be applied preventing further processing by the **BeforeHTTP** and **TCPRequest** events.

We didn't program any sample-code in this event because SL Server 4 supports a user-interface to the firewall table. The connection is held against the firewall table just before the **OnConnect** event is triggered. You may use this event to programmatically apply rules that are too complex to add to the firewall table.

Sub BeforeHTTP(Index)

This event occurs after the **OnConnect** event and before HTTP processing by the **TCPRequest** event. Use this event to scan the HTTP request to validate the request. If a request turns out to be invalid, you may terminate the request disallowing further processing by the **TCPRequest** event.

The Core part II - UDP

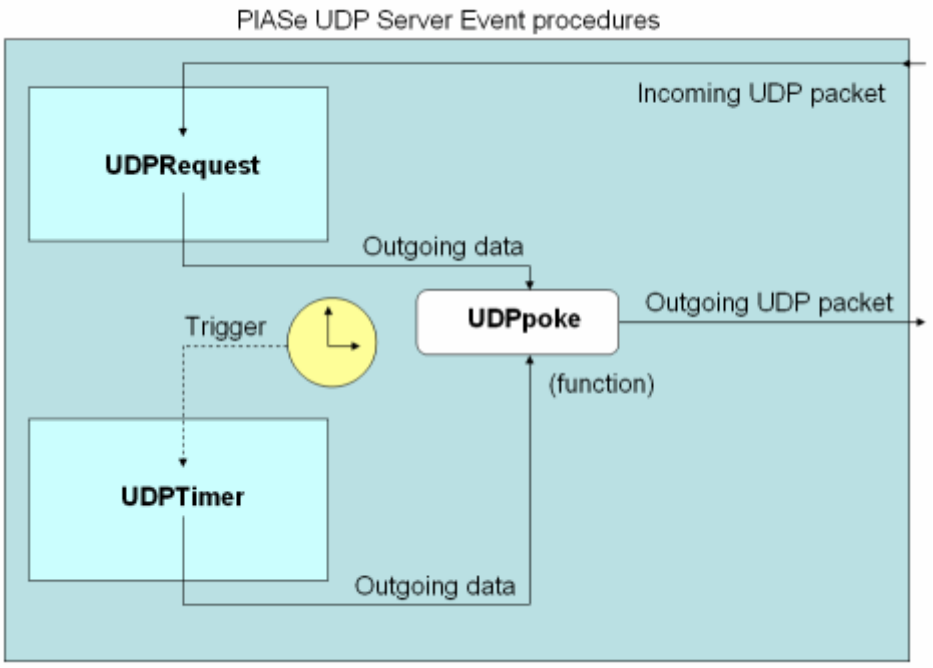
What is UDP broadcasting ?

UDP broadcasting is a system mainly used to notify other services/clients by broadcasting short messages to a group of network stations in a particular network without knowing their host ID number.

A UDP server in common does need not to reply when it received a broadcasted packet but programmable options to respond to individual network station are available within the PIASe programming library of SL Server 4.

SL Server 4 will be installed having no active UDP listener. Keep the UDP listener disabled if you're not planning to use a broadcast system and prevent useless processing consumption.

Implementation chart of the UDP Server Event procedures



Pure Visual Basic

UDP Client/Server Event procedures are programmed in pure Visual Basic ONLY.

Firewall interaction

The built-in firewall is called before triggering the UDPRequest event. Connections discarded by the firewall never trigger the UDPRequest event.

Event Procedures

Location: /admin/server/udpserver.os

Sub UDPRequest(Ip, Data)

Occurs whenever a UDP packet has been arrived at the listening UDP port. The Ip parameter returns the client IP address of the submitting machine and the Data parameter returns the (raw) data that

was submitted by the client. We've programmed a sample how UDP can be used to receive broadcasted ID records from other SL Server 4 instances for identification purposes. Note: The default setting for UDP broadcasting is set to OFF.

Sub UDPTimer()

This event occurs whenever the UDP timer is activated. The timer is used to broadcast UDP packets at regular intervals. We've programmed a sample how UDP can be used to broadcast ID records to other SL Server 4 instances for identification purposes. Note: The default setting for the UDP timer is set to: OFF.

Server Side Scripts

Combining HTML and script

Unlike the pure Visual Basic coding style in event script files, PIASe and HTML code are combined in the same document having an **SSC** extension. To indicate server-side script you must use the <% and %> script **block** tag's. To indicate the insertion of a result from a PIASe expression into the HTML you must use the <%= and %> script **insertion** tag's. When you start writing script you always begin with an block / insertion start tag (<% or <%=) and you end the script with an block / insertion end tag (%>).

A simple example:

```
<%
    Text = "Hello world"
%>
<html>
<head>
<title><%= Text %></title>
</head>
<body>
<p><%= Text %></p>
</body>
</html>
```

The script that's inside the script-tags will be evaluated at server-side and is never visible to the client. The blue marked HTML and the result of the two insertion tag's will be sent to the client. In the first tag the variable *Text* is assigned with the text "Hello world". The content of the variable is then inserted as the title of the document (second tag) and finally inserted as part of the body (third tag).

The client would see this as the source of the received document:

```
<html>
<head>
<title>Hello world</title>
</head>
<body>
<p>Hello world</p>
</body>
</html>
```

Logical flow

HTML can be part of logical program-flow such as the IF / THEN block, DO / LOOP and FOR / NEXT loops and SELECT CASE switches. This feature is very important when building documents that are conditionally based on different text-fragments.

```
<html>
<head>
<title>Conditional building</title>
</head>
<body>
<p>A If/Then:</p>
<%
  SomeVal = 1234
  Test = 2
  If Test = 1 Then
%>
<p>Block one <%= SomeVal %></p>
<%
  ElseIf Test = 2 Then
%>
<p>Block two <%= SomeVal %></p>
<%
  Else
%>
<p>Block three <%= SomeVal %></p>
<%
  End If
%>
<p>A For/Next:</p>
<%
  For N = 1 to 10
    If N > 4 Then
%>
<p>N = <%= N %></p>
<%
    End If
  Next
%>
</body>
</html>
```

Including files

You may, when writing combined script and HTML use the include tag to include files. You may not use an insertion tag inside the include tag to specify dynamically the name of the file to be included. Because including of files is done by the pre-processor and pre-processing is done before the script is executed, the dynamic insertion (that depends on execution of the given expression) will not have any effect.

Breaking long lines

It is not allowed to use the underscore (_) to break long lines into multiple lines when combining script and HTML. Only when you write script in pure Visual Basic allows the usage of the underscore to break long lines into multiple lines.

Document naming

PIASe server-side script documents have an **SSC** extension. If you wouldn't name the document correctly the server would send your source to the client without pre-processing or running the script.

PIASe server-side script must contain at least one block or insertion tag

If you wouldn't add at least one server-side script tag pair, the pre-processor would be confused and generates an error because it expects a server-side script tag due to the **SSC** extension of the script file (name a document that lacks any block or insertion tag with a **HTM** extension).

Preprocessing

Because of the embedded nature of server-side-script, some Pre-processing must be done whenever you change a script file. Pre-processing takes some time (depending the speed of the computer's processor) and is performed only once until the the next modification.

Pure Visual Basic instead of combined script

While promoting the usage of script and HTML combined in one document it still is possible to write server-side scripts in pure Visual Basic. Pure Visual Basic script must contain a main procedure having one index parameter.

Sub Main(Index)

Note that HTML may exist in pure Visual Basic script ONLY as part of a string expression. Script tag's are forbidden when programming pure Visual Basic scripts. Pure Visual Basic script does’t require pre-processing and execute faster.

Visual Basic script programming

First of all, you need to know Visual Basic script programming. Visual Basic script is an easy to learn programming language and if you don't know VBS please check the internet. There are plenty good (and not so good) online tutorials. If you want to know more about VBS you might want to visit Microsoft's MSDN website and download documentation and help about Visual Basic script. Due to copyrights and trade-mark law, VBScript documentation is not included in this manual.

The library

PIASe has an extensive function library to write advanced server-side script. There are over 200 functions and procedures in the library.

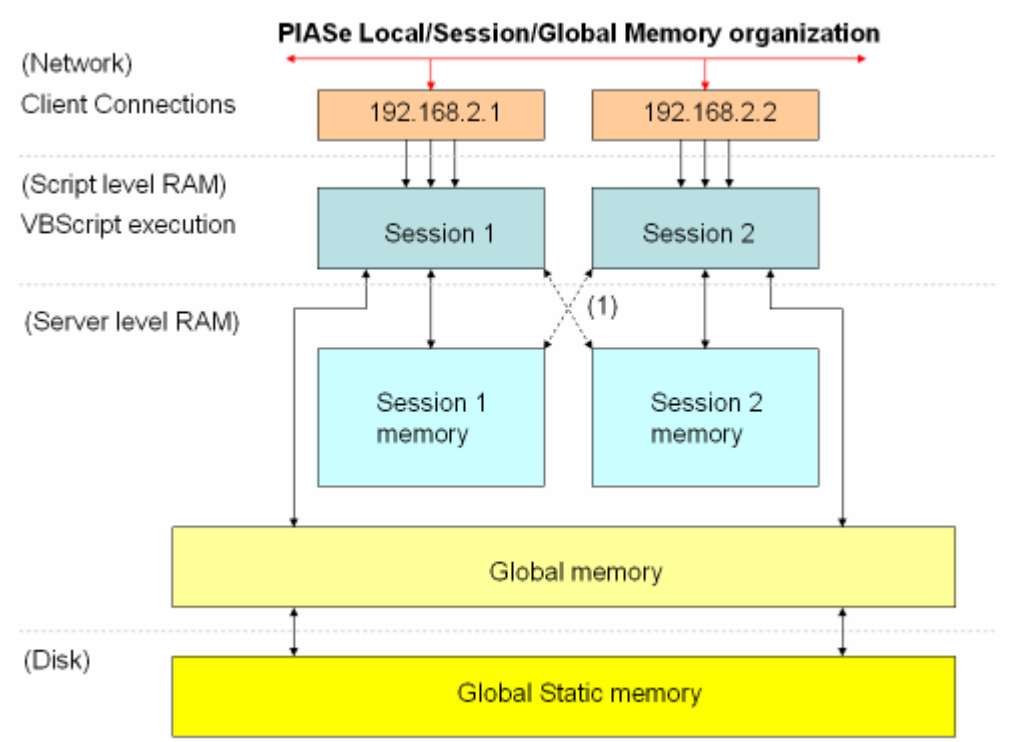
Memory organization

Local and Module level VBScript variables

Incoming connections from a unique client are grouped into a unique client session. When a script is executed all Local and Module level VBScript variables (Script Level RAM) belong exclusively to the executing script and are erased when script execution ends.

Though Local and Module level variables provide common memory tasks during the execution of a script, It will never support sharing between connections, sessions and server runs. To provide such interface the Session memory block and Global memory block were introduced.

Memory organization chart



Session memory block

A Session memory block (Server level RAM) provides the sharing of data during a session. Scripts executing within the scope of a session may also access and alter session memory of other sessions (1).

Global memory block

The Global memory block (Server level RAM) provides a globally shared memory interface. Global Shared variables usually contains values that need to be available throughout all sessions (regardless the type of application) during a server run. Global memory variables are erased when the server program terminates.

Global Static (disk) memory block

Variables in the Global memory block can be set **static** enabling persistent storage to disk and are reloaded from disk every time the server is started. Page counter and application configuration variables usually are global static variables.

The global memory block is not organized as a database nor has it any database features. It is therefore not recommend to use the global memory block as a database. Use the file system or a OLEDB interface to a database connection to store large amounts of data. Heavily packed global memory will have a negative impact on the performance of the server.

PIASe Function Reference

Function ActiveConnectionCount

Description: Returns the number of established connections

For every resource that is downloaded by the client a new connection is created. When clients are downloading resources simultaneously multiple connections are handled at the same time. The value returned by this function represents the actual number of connections currently being served.

Parameters: None

Sub AddFailedLogins

Description: Increase the number of failed logins for a specific session

Use this function to keep track on failed logins when using a custom login form (i.e. not using the standard login dialog of a browser). You can extract the SessionID parameter by calling the function GetSessionID. Retrieve the actual login failure count by calling the GetFailedLogins function. When the standard login dialog is used and a user fails to authenticate correctly, the count is increased by the server using this function. If a user is failing a lot you may (silently) block further access to the server (preventing hack attempts) or may take action to notify the user about excessive login failure (a user may have forgotten the password).

Parameters:

Name	Type	Description
SessionID	string	Session ID

Sub AddIP2

Description: Add a new entry to the firewall

If you wish to enter a single IP adress, use the IPStartingAddress parameter for the actual address and set the IPEndingAddress parameter to 0.0.0.0.

Parameters:

Name	Type	Description
IPStartingAddress	string	Starting IP address-range
IPEndingAddress	string	Ending IP address-range
Block	bool	Set to true for blocking, false for non-blocking

Description	string	Description of the new entry

Function AdminLoginB64

Description: Returns a Base64 encoded authorization header field

It's used to quickly compare the authentication header against the credentials of the administrator account. This function is downward compatible to older versions of SL Server 4 and has become obsolete since the introduction of RSA's MD5 authentication method. It is NOT advised to use the Base64 authentication method at client and server side due to the transfer of "clear text" that is used for encoding the credentials.

To prevent snooping make sure that the server is set to MD5 authentication and that the client indicates a secure login method before you logon as an administrator. Most browsers support MD5 but not all browsers will notify you (visually) about the usage of the MD5 method.

Parameters: None

Function AppendFile

Description: Appends a string to a file

Specify a full path including drive and subfolders of a file you wish to append the string to. If the file doesn't exist it will be created. Use this function for logging events or for writing large dynamically created file-fragments.

The function returns zero when succeeded, non-zero when it failed to write the string.

Parameters:

Name	Type	Description
Filename	string	Absolute path and filename
DataToAppend	string	Data to append at the end of the file

Function AuthenticateHeader

Description: Returns a HTTP authenticate header

Returns a HTTP authenticate (response) header that can be sent back to the client to force a login. Depending the server setting the header will either request a Base64 or MD5 digest login. Note that the header must be responded to the client using the SetDataRawOut procedure.

Use this function to authenticate users when scripts that require authentication are stored in folders having anonymous permissions.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function Base64Decoder

Description: Decode a Base64 encoded string

Base64 is an well-known method to format data in such way that it can be transferred trough all possible channels the internet is using including email. Base64 is not an encryption protocol (mistakenly said because you can't read-back text visually). Base64 does not use any tokens to encode/decode data.

Parameters:

Name	Type	Description
Base64Str	string	Base64 encoded string

Function Base64Encoder

Description: Encode a string using the Base64 format

Base64 is an well-known method to format data in such way that it can be transferred trough all possible channels the internet is using including email. Base64 is not an encryption protocol (mistakenly said because you can't read-back text visually). Base64 does not use any tokens to encode/decode data.

Parameters:

Name	Type	Description
SourceStr	string	String to be encoded

Sub BlockSession

Description: Block session

The session of a client can be blocked for further service when the client is not allowed any new requests because you need to maintain the system or the client is behaving in such way that you don't want to serve any new connections. A currently running connection will be completed before the block is applied.

Note that the block is not effective when the client is starting a new session (opens a new browser window). Use the firewall functions to block persistent clients by adding their IP-address to the firewall table.

Parameters:

Name	Type	Description
SessionID	string	Session ID

Function CachedFilesCount

Description: Returns the number of cached files

SL Server 4 is using a dynamic file-cache to improve performance. This function returns the actual number of cached files currently loaded into memory.

Parameters: None

Sub ClearCache

Description: Release all cached files

This function will remove all cached files from the file-cache. Normally it is not necessary to clear the cache but when you restored a file in the VFS having a file date older than the currently cached one you must clear the cache to reload the restored file otherwise the newer (and obsolete) file version will be served from the cache.

Parameters: None

Sub ClearFirewall

Description: Clear all firewall entries

This function will remove all firewall entries from the firewall. Most likely, you will not use this function unless you write your own firewall maintenance utility.

Parameters: None

Sub ClearScriptParameters

Description: Clear all script parameters

Script parameters are used when (non-recursively) calling a script from another script to transfer values the way you would when calling sub's and functions. Script parameters are statically stored and must be cleared when exiting the script.

Parameters:

Name	Type	Description

Index	numeric	Reference to a connection object

Function ClosePort

Description: Close a connection

This function will close the connection a client has established when requesting a resource. Normally you do not need to close a connection because the server is closing it for you when completing the execution of a script. There are some exceptions where you must force a close and mostly when you're programming multi-processing scripts (scripts taking over processing for other currently running scripts).

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Sub ClrGlobal

Description: Clear all global variables from memory

Global memory can be accessed by all sessions and if set, be persistent (global static) between different runs of the server. Because of the persistent nature you may need to clear it once in a while to free-up memory. Note that when appications are using global static variables that they must also support detection of non-initialized (cleared) global variables and if necessary reset non-initialized global variables to their initial values. It's not advised to clear global memory without knowing if installed scripts that use global memory will handle reset and initialization of global variables.

Parameters: None

Sub ClsDebug

Description: Clear the debug window

The debug window is the immediate window where you can print results from script expressions for testing purposes.

Parameters: None

Function ConvertImageToJPEG

Description: Converts a bimtmap (BMP) file to a JPEG image-file

Currently, only BMP bitmaps are supported as input file. When using a high compression ratio (up to 100) the image becomes extremely small but may loose details and color definition. The recommended compression value is about 70 for photo’s.

Parameters:

Name	Type	Description
ImageInputFile	string	Path of the input imagefile
ImageOutputFile	string	Path of the output imagefile
CompressionRatio	numeric	Compression ratio

Function CopyFile

Description: Copies a single file

If you don't want to overwrite any existing file, set the OverwriteExistingFile boolean to True. The function will return 58 (File already exists) when it tries to overwrite the file. When the file was copied successfully zero is returned otherwise a non-zero number is returned.

The function is using the Scripting.Filesystem object and you may receive a notification from a virus-scanner that SL Server 4 might be a virus.

Parameters:

Name	Type	Description
SourceFilename	string	The absolute path and filename of the file to copy
DestinationFilename	string	The absolute path and filename of the destination
OverwriteExistingFile	bool	Overwrite existing destination if set to true

Sub CopyScreenAreaToFile

Description: Save partially screen to a bitmap

Saves an area of the screen to a bitmap file. Currently only the BMP format is supported. If you want to clip the entire desktop use function CopyScreenToFile or determine the height of the desktop using function GetDTHeight and width using function GetDTWidth. This function will only at server-side.

Parameters:

Name	Type	Description
X	numeric	Topmost column
Y	numeric	Topmost row
Width	numeric	Width
Height	numeric	Height
BitmapFilename	string	Filename of the bitmap

Sub CopyScreenToFile

Description: Save screen to a bitmap

Saves an full area of the screen to a bitmap file. Currently only the BMP format is supported. This function will only at server-side.

Parameters:

Name	Type	Description
BitmapFilename	string	Filename of the screen-dump bitmap

Function CreateServerObject

Description: Create a server object

The function will create a server object in a reserved object pool and returns a boolean true when succeeded. See function GetServerObjectRef how to retrieve a object link to a local variable for the created server object. You may create a maximum of 32 server objects.

Parameters:

Name	Type	Description
Name	string	Name of the server object to create
Class	string	Object class
Server	optional, string	Servername

Function CryptDecode

Description: Decode a encrypted string

Decodes only strings that are encoded using the CryptEncode function. The methods used to encrypt data (masking of bytes) are very simple and can be decoded easily by specialists. Use the CryptGetKey function to generate an encryption key based on a password. Never use a password directly as a encryption key.

Encrypted data cannot be exchanged between different instances of SL Server 4.

Parameters:

Name	Type	Description
EncryptedText	string	Encrypted string
EncryptionKey	string	Key

Function CryptEncode

Description: Create an encrypted string

Use the CryptDecode function to decode the string. The methods used to encrypt data (masking of bytes) are very simple and can be decoded easily by specialists. Use the SHA encryption methods to prevent easy decoding. Use the CryptGetKey function to generate an encryption key based on a password. Never use a password directly as the encryption key.

Encrypted data cannot be exchanged between different instances of SL Server 4.

Parameters:

Name	Type	Description
AsciiText	string	ASCII string
EncryptionKey	string	Key

Function CryptGetKey

Description: Get a encryption key

Returns a encryption key based on a password.

Encrypted data cannot be exchanged between different instances of SL Server 4.

Parameters:

Name	Type	Description
Password	string	A password to de/en-code PIASe native encryption

Function CustomLibInfo

Description: Returns the version number and info of the custom library.
(only available for customized server versions)

If we have developed custom communication tools for you (or your company) you may retrieve custom library information by calling this function to make sure that you are using the latest version of the library - or - using a compatible server version in respect to your script source.

Parameters: None

Sub DataPump

Description: Schedule a file to be transferred in a background process

Can only be used in native VBScript (pure visual basic) The content type and expiration of the file must be set before calling the DataPump function and the sub must be exited after the call. No output may be written preceding the call. SL Server 4 is using the DataPump function whenever a static file exceeds the size of 10 Kb, enabling service to multiple connections at the same time without being stalled by large exclusive transfers. Normally you do not need to call this function when using the standard server OS scripts.

Parameters:

Name	Type	Description
Index	numeric	Index to a connection object
Filename	string	The absolute path and filename to transfer

Function DecodeUrlEncoded

Description: Decode a Url encoded string to a ASCII string

When values are submitted by a client using the POST or GET method you must convert the received data before you actually can use it. A space will not be submitted using ASCII character 32 for example, but will be submitted using the hexadecimal notation (%20).

Parameters:

Name	Type	Description
URLcodedText	string	An URL coded string to encode

Sub Decr

Description: Decrements a numeric variable

Parameters:

Name	Type	Description
Variable	numeric	The variable to decrement

Function DecrFile

Description: Decrypte a file

The methods used to encrypt data (masking of bytes) are very simple and can be decoded easily by specialists. Use the SHA encryption methods to prevent easy decoding. Use the EncrFile function de encrypt files.

Encrypted data cannot be exchanged between different instances of SL Server 4.

Parameters:

Name	Type	Description
Filename	string	Absolute path and filename of the file to process
Password	string	Password

Function DecrFilePub

Description: Decrypt a file (public)

The methods used to encrypt data (masking of bytes) are very simple and can be decoded easily by specialists. Use the SHA encryption methods to prevent easy decoding. Use function EncrFilePub to encrypt files that can be exchanged between SL Server 4s.

Parameters:

Name	Type	Description
Filename	string	Absolute path and filename of the file to process
Password	string	Password

Function DecryptSHA160

Description: Decode a SHA 160 bits encrypted string

This method has a strong encryption algorithm and cannot be decoded easily without knowing the password. Use the CryptGetKey function to create a encryption key based on a password. Use function EncryptSHA160 to encrypt files using SHA 160 bits algorithm.

Encrypted data cannot be exchanged between different instances of SL Server 4.

Parameters:

Name	Type	Description
EncryptedText	string	Encrypted string
EncryptionKey	string	Key

Function DecryptSHA160Pub

Description: Decode a public SHA 160 bits encrypted string

This method has a strong encryption algorithm and cannot be decoded easily without knowing the password. Use the CryptGetKey function to create a encryption key based on a password. Use function EncryptSHA160Pub to encrypt files using SHA 160 bits algorithm.

Encrypted data can be exchanged between different instances of SL Server 4.

Parameters:

Name	Type	Description

EncryptedText	string	Encrypted string
EncryptionKey	string	Key

Function DeleteFile

Description: Removes a file

The file will be deleted from the disk without being placed in the recycle bin. When the file was deleted successfully, A zero is returned otherwise a non-zero number is returned.

Parameters:

Name	Type	Description
FileName	string	Absolute path and filename of the file to delete

Sub DeleteFirewallEntry

Description: Removes a firewall entry

Firewall entries are uniquely addressed by their IP-addresses and the function will only delete an entry when both the IP and IP2 properties match the given IP addresses.

Parameters:

Name	Type	Description
ClientIPaddressStartRange	string	IP address start range to be deleted
ClientIPaddressEndRange	string	IP address end range to be deleted

Function DeleteFolder

Description: Removes a folder

The folder must be empty to remove it and will not be moved to the recycle bin. Returns a zero when the folder was removed successfully or a non-zero number when the function fails to delete the folder.

Parameters:

Name	Type	Description
Foldername	string	Absolute foldername

Function DeleteSession

Description: Deletes a session variable

Returns an nonzero number if the deletion was succeeded.

Parameters:

Name	Type	Description
SessionID	string	Session ID
SessionVariableName	string	Name of

Function DeleteSystemUser

Description: Removes a user record

Use function GetSystemUser to determine the correct offset from the user record you wish to remove from the user table. The server will revoke any permission when removing the user record of a logged on user unless user anonymous has been granted for the same permission.

Parameters:

Name	Type	Description
Offset	numeric	Reference to an entry in the user table

Function DestroyServerObject

Description: Destroy a server object

Returns a boolean true when the object was destroyed successfully. Use whenever you created a server object and the server is shutting down or when the object is not needed anymore.

Parameters:

Name	Type	Description

Name	string	Name of the object

Function DownloadUrl

Description: Download a HTTP resource and save the results to a file.

The function will return True when the operation succeeded or False when it failed to retrieve the HTTP resource.

Name	Type	Description
Url	string	Location of the HTTP resource
OutputFile	string	Filename containing the results

Sub DPrint

Description: Print debug message

The debug window is the immediate window where you can print results from script expressions for testing purposes. If you omit the TextToOutput variable, a blank line will be printed. This function works only in the IDE edition.

Parameters:

Name	Type	Description
TextToOutput	string,optional	Output to the debug console

Function EmbedInHTML

Description: Returns a HTML formatted page

Returns a default formatted HTML page for quick dumping of information. Specify the title (PageTitle), font face (FontFace), font size (FontSize) and set the ConvertCrLfBreak parameter to true if CRLF's inside the data (specified by HTMLDataToEmbed) must be converted to the equivalent HTML tag 'BR'.

Parameters:

Name	Type	Description
PageTitle	string	Pagetitle

FontFace	string	Fontname
FontSize	numeric	Fontsize
HTMLDataToEmbed	string	Html data to insert
ConvertCrLfToBreak	bool	Convert CRLF to BR

Sub EmulateKB

Description: Send keystrokes

This will cause the submission of emulated keystrokes to the active window at server side.

Special keys:

Type a + to indicate a SHIFT pressed state, a ^ to indicate a CTRL pressed state and % to indicate a ALT pressed state.

To emulate the plus, caret and percent signs you must enclose them in braces:

+ or PLUS: {+}

^ or CARET: {^}

% or PERCENT: {%}

As well as other special keys:

BACKSPACE: {BACKSPACE}, {BS}, or {BKSP}

BREAK: {BREAK}

CAPS LOCK: {CAPSLOCK}

DEL or DELETE: {DELETE} or {DEL}

DOWN ARROW: {DOWN}

END: {END}

ENTER: {ENTER} or ~

ESC: {ESC}

HELP: {HELP}

HOME: {HOME}

INS or INSERT: {INSERT} or {INS}

LEFT ARROW: {LEFT}

NUM LOCK: {NUMLOCK}

PAGE DOWN: {PGDN}

PAGE UP: {PGUP}

PRINT SCREEN: {PRTSC}

RIGHT ARROW: {RIGHT}

SCROLL LOCK: {SCROLLLOCK}

TAB: {TAB}

UP ARROW: {UP}

F1: {F1}

F2: {F2}

F3: {F3}

F4: {F4}

F5: {F5}

F6: {F6}

F7: {F7}

F8: {F8}

F9: {F9}

F10: {F10}

F11: {F11}

F12: {F12}

F13: {F13}

F14: {F14}

F15: {F15}

F16: {F16}

Parameters:

Name	Type	Description
Keys	string	Key-strokes
Wait	optional, bool	Wait until keys are processed

Function EncodeUrlDecoded

Description: Encode a ASCII string to a URL coded string

Some ASCII characters cannot be sent using the HTTP protocol. To prevent problems the URL and data enclosed by <input> tags must be sent in a Url encoding format enabling internet devices and clients to interpret the data correctly. Sending two ASCII 13 characters as part of the URL may cause the client to think that the header has ended and the HTTP content has started. Therefore, ASCII character 13 must be submitted as %0D when used in the Url or input values.

Parameters:

Name	Type	Description
AsciiText	string	An ASCII string to encode

Function EncrFile

Description: Encrypt a file

The methods used to encrypt data (masking of bytes) are very simple and can be decoded easily by specialists. Use the SHA encryption methods to prevent easy decoding. Use the DecrFile function de decrypt files.

Encrypted data cannot be exchanged between different instances of SL Server 4.

Parameters:

Name	Type	Description
Filename	string	Absolute path and filename of the file to process
Password	string	Password

Function EncrFilePub

Description: Encrypt a file (public)

The methods used to encrypt data (masking of bytes) are very simple and can be decoded easily by specialists. Use the SHA encryption methods to prevent easy decoding. Use function DecrFilePub to decrypt files that can be exchanged between SL Server 4s.

Parameters:

Name	Type	Description
Filename	string	Absolute path and filename of the file to process
Password	string	Password

Function EncryptSHA160

Description: Encode a SHA 160 bits encrypted string

This method has a strong encryption algorithm and cannot be decoded easily without knowing the password. Use the CryptGetKey function to create a encryption key based on a password. Use function DecryptSHA160 to decrypt files using SHA 160 bits algorithm.

Encrypted data cannot be exchanged between different instances of SL Server 4.

Parameters:

Name	Type	Description
AsciiText	string	ASCII string
EncryptionKey	string	Key

Function EncryptSHA160Pub

Description: Encode a public SHA 160 bits encrypted string

This method has a strong encryption algorithm and cannot be decoded easily without knowing the password. Use the CryptGetKey function to create a encryption key based on a password. Use function DecryptSHA160Pub to decrypt files using SHA 160 bits algorithm. Encrypted data can be exchanged between different instances of SL Server 4.

Parameters:

Name	Type	Description

AsciiText	string	ASCII string
EncryptionKey	string	Key

Sub EndServices

Description: Terminate all services and end program (Remote shutdown)

Check before calling this function for user authorization. It wouldn't be fun when a anonymous user is shutting the server down.

Parameters: None

Sub ExecScript

Description: Executes a script file using it's virtual filename

Note: An 'Exit Sub' statement is required following the ExecScript call.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
ScriptFilename	string	Virtual path and filename of the script
Parameter1	variant, optional	Parameter
Parameter2	variant, optional	Parameter
Parameter3	variant, optional	Parameter
Parameter4	variant, optional	Parameter
Parameter5	variant, optional	Parameter
Parameter6	variant, optional	Parameter
Parameter7	variant, optional	Parameter
Parameter8	variant, optional	Parameter
Parameter9	variant, optional	Parameter

Parameter10	variant, optional	Parameter

Sub ExecScriptAbs

Description: Executes a script file using it's absolute filename

Note: An 'Exit Sub' statement is required following the ExecScriptAbs call.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
ScriptFilename	string	Absolute path and filename of the script
Parameter1	variant, optional	Parameter
Parameter2	variant, optional	Parameter
Parameter3	variant, optional	Parameter
Parameter4	variant, optional	Parameter
Parameter5	variant, optional	Parameter
Parameter6	variant, optional	Parameter
Parameter7	variant, optional	Parameter
Parameter8	variant, optional	Parameter
Parameter9	variant, optional	Parameter
Parameter10	variant, optional	Parameter

Sub ExternalExec

Description: Runs a CLI application

This function is specially designed for use with a CLI application and can't be used as a standard shell. The command requires a input file (specified by InputFilename) supporting the CLI application with it's input (the request header) and produces a output file (specified by parameter OutputFilename) when CLI completed (the HTTP content).

During the execution of the CLI the output file must be locked exclusively for write by the CLI application. The function will exit immediately after calling and the produced CLI output is directed to the client when it becomes available. Set the ResultContainsHeaders parameter to true when the CLI support header fields in the result enabling the server to merge it with already existing fields.

After calling you must exit the script using the 'Exit Sub' or by normal termination. In either case no output data may exist in the output buffer.

The connection stays open during the execution of the CLI application and will be closed by the successful submission of the result, timeout or abort a client side. CLI applications may run simultaneously.

Parameters:

Name	Type	Description
Index	numeric	Index to a connection object
ExternalCommand	string	Shell argument
InputFilename	string	Absolute path to a input file
OutputFilename	string	Absolute path to a output file (created by function, must not exist when calling)
ResultContainsHeaders	boolean, optional	Indicates a header/content result rather than a content only result

Function FileExist

Description: Test for the existence of a file

Returns a boolean true (-1) when the file specified by Filename exist or a boolean false (0) when the file doesn't exist. The test will only work for normal, archives, hidden, read only and system files.

Parameters:

Name	Type	Description
Filename	string	Absolute path and filename of the file to test

Function FirewallCount

Description: Returns the number of firewall entries

Use this function when enumerating entries to determine the last item in the firewall. See function GetFirewallEntry how to retrieve entries. The firewall will only count blocking entries when the optional CountOnlyBlockedEntries parameter is set to true.

Parameters:

Name	Type	Description
CountOnlyBlockedEntries	bool, optional	If set, count only blocked items

Function FlushLogfiles

Description: Commits cached log data to disk

Use this function when you want to retrieve the log files. The logging mechanism of SL Server 4 does write log lines to disk every time the logger is hit but queues it before writing a larger block of log data to disk. If you want to be able to fetch the last set of log events you must flush the logger's memory to disk.

Parameters: None

Function Fmt

Description: Visual Basic Format function

Executes the VB format function using one format section

Formatting commands

:

Time separator. In some locales, other characters may be used to represent the time separator. The time separator separates hours, minutes, and seconds when time values are formatted. The actual character used as the time separator in formatted output is determined by your system settings.

/

Date separator. In some locales, other characters may be used to represent the date separator. The date separator separates the day, month, and year when date values are formatted. The actual character used as the date separator in formatted output is determined by your system settings.

C

Display the date as ddddd and display the time as t t t t t, in that order. Display only date information if there is no fractional part to the date serial number; display only time information if there is no integer portion.

D

Display the day as a number without a leading zero (1 - 31).

Dd

Display the day as a number with a leading zero (01 - 31).

Ddd

Display the day as an abbreviation (Sun - Sat).

dddd

Display the day as a full name (Sunday - Saturday).

ddddd

Display the date as a complete date (including day, month, and year), formatted according to your system's short date format setting. The default short date format is m/d/yy.

dddddd

Display a date serial number as a complete date (including day, month, and year) formatted according to the long date setting recognized by your system. The default long date format is mmmm dd, yyyy.

w

Display the day of the week as a number (1 for Sunday through 7 for Saturday).

ww

Display the week of the year as a number (1 - 53).

m

Display the month as a number without a leading zero (1 - 12). If m immediately follows h or hh, the minute rather than the month is displayed.

mm

Display the month as a number with a leading zero (01 - 12). If m immediately follows h or hh, the minute rather than the month is displayed.

mmm

Display the month as an abbreviation (Jan - Dec).

mmmm

Display the month as a full month name (January - December).

q

Display the quarter of the year as a number (1 - 4).

y

Display the day of the year as a number (1 - 366).

yy

Display the year as a 2-digit number (00 - 99).

yyyy

Display the year as a 4-digit number (100 - 9666).

h

Display the hour as a number without leading zeros (0 - 23).

hh

Display the hour as a number with leading zeros (00 - 23).

n

Display the minute as a number without leading zeros (0 - 59).

nn

Display the minute as a number with leading zeros (00 - 59).

s

Display the second as a number without leading zeros (0 - 59).

ss

Display the second as a number with leading zeros (00 - 59).

ttttt

Display a time as a complete time (including hour, minute, and second), formatted using the time separator defined by the time format recognized by your system. A leading zero is displayed if the leading zero option is selected and the time is before 10:00 A.M. or P.M. The default time format is h:mm:ss.

AM/PM

Use the 12-hour clock and display an uppercase AM with any hour before noon; display an uppercase PM with any hour between noon and 11:59 P.M.

am/pm

Use the 12-hour clock and display a lowercase AM with any hour before noon; display a lowercase PM with any hour between noon and 11:59 P.M.

A/P

Use the 12-hour clock and display an uppercase A with any hour before noon; display an uppercase P with any hour between noon and 11:59 P.M.

a/p

Use the 12-hour clock and display a lowercase A with any hour before noon; display a lowercase P with any hour between noon and 11:59 P.M.

AMPM

Use the 12-hour clock and display the AM string literal as defined by your system with any hour before noon; display the PM string literal as defined by your system with any hour between noon and 11:59 P.M. AMPM can be either uppercase or lowercase, but the case of the string displayed matches the string as defined by your system settings. The default format is AM/PM.

0 (Zero)

Display a digit or a zero. If the expression has a digit in the position where the 0 appears in the format string, display it; otherwise, display a zero in that position. If the number has fewer digits than there are zeros (on either side of the decimal) in the format expression, display leading or trailing zeros. If the number has more digits to the right of the decimal separator than there are zeros to the right of the decimal separator in the format expression, round the number to as many decimal places as there are zeros. If the number has more digits to the left of the decimal separator than there are zeros to the left of the decimal separator in the format expression, display the extra digits without modification.

#

Display a digit or nothing. If the expression has a digit in the position where the # appears in the format string, display it; otherwise, display nothing in that position. This symbol works like the 0 digit placeholder, except that leading and trailing zeros aren't displayed if the number has the same or fewer digits than there are # characters on either side of the decimal separator in the format expression.

.

In some locales, a comma is used as the decimal separator. The decimal placeholder determines how many digits are displayed to the left and right of the decimal separator. If the format expression contains only number signs to the left of this symbol, numbers smaller than 1 begin with a decimal separator. If you always want a leading zero displayed with fractional numbers, use 0 as the first digit placeholder to the left of the decimal separator instead. The actual character used as a decimal placeholder in the formatted output depends on the Number Format recognized by your system.

%

The expression is multiplied by 100. The percent character (%) is inserted in the position where it appears in the format string.

,

In some locales, a period is used as a thousand separator. The thousand separator separates thousands from hundreds within a number that has four or more places to the left of the decimal separator. Standard use of the thousand separator is specified if the format contains a thousand separator surrounded by digit placeholders (0 or #). Two adjacent thousand separators or a thousand separator immediately to the left of the decimal separator (whether or not a decimal is specified) means "scale the number by dividing it by 1000, rounding as needed." You can scale large numbers using this technique. For example, you can use the format string "##0,," to represent 100 million as 100. Numbers smaller than 1 million are displayed as 0. Two adjacent thousand separators in any position other than immediately to the left of the decimal separator are treated simply as specifying the use of a thousand separator. The actual character used as the thousand separator in the formatted output depends on the Number Format recognized by your system.

E- E+ e- e+

If the format expression contains at least one digit placeholder (0 or #) to the right of E-, E+, e-, or e+, the number is displayed in scientific format and E or e is inserted between the number and its exponent. The number of digit placeholders to the right determines the number of digits in the exponent. Use E- or e- to place a minus sign next to negative exponents. Use E+ or e+ to place a minus sign next to negative exponents and a plus sign next to positive exponents.

-+$()

To display a character other than one of those listed, precede it with a backslash (\) or enclose it in double quotation marks (" ").

\

Many characters in the format expression have a special meaning and can't be displayed as literal characters unless they are preceded by a backslash. The backslash itself isn't displayed. Using a backslash is the same as enclosing the next character in double quotation marks. To display a backslash, use two backslashes (\\). Examples of characters that can't be displayed as literal characters are the date- and time-formatting characters (a, c, d, h, m, n, p, q, s, t, w, y, and /:), the numeric-formatting characters (#, 0, %, E, e, comma, and period), and the string-formatting characters (@, &, <, >, and !).

"ABC"

To include a string in format from within code, you must use Chr(34) to enclose the text (34 is the character code for a double quotation mark).

@

Display a character or a space. If the string has a character in the position where the @ appears in the format string, display it; otherwise, display a space in that position. Placeholders are filled from right to left unless there is an ! character in the format string. See below.

&

Display a character or nothing. If the string has a character in the position where the & appears, display it; otherwise, display nothing. Placeholders are filled from right to left unless there is an ! character in the format string. See below.

<

Display all characters in lowercase format.

>

Display all characters in uppercase format.

!

The default is to fill from right to left.

Currency

Shows negative numbers inside parentheses.

Fixed

Shows at least one digit.

Standard

Uses a thousands separator.

Percent

Multiplies the value by 100 with a percent sign at the end.

Scientific

Uses standard scientific notation.

General Date

Shows date and time if expression contains both. If expression is only a date or a time, the missing information is not displayed.

Long Date

Uses the Long Date format specified in the Regional Settings dialog box of the Microsoft Windows Control Panel.

Medium Date

Uses the dd-mmm-yy format (for example, 03-Apr-93)

Short Date

Uses the Short Date format specified in the Regional Settings dialog box of the Windows Control Panel.

Long Time

Shows the hour, minute, second, and "AM" or "PM" using the h:mm:ss format.

Medium Time

Shows the hour, minute, and "AM" or "PM" using the "hh:mm AM/PM" format.

Short Time

Shows the hour and minute using the hh:mm format.

Yes/No

Any nonzero numeric value (usually - 1) is Yes. Zero is No.

True/False

Any nonzero numeric value (usually - 1) is True. Zero is False.

On/Off

Any nonzero numeric value (usually - 1) is On. Zero is Off.

Parameters:

Name	Type	Description
Value	numeric	Value to format
FormattingCommands	string	Formatting commands

Function FreeCache

Description: Returns the offset of a free slot in the file cache

Parameters: None

Function FromHex

Description: Convert a hexadecimal string to ascii

Note that 2 hex digits form one character.

Parameters:

Name	Type	Description
Hexed	string	char-hexed string

Function GetAbsVPath

Description: Returns an absolute virtual path to a destination path starting from the current path

This function resolves path traversals such as '../..' by replacing them by their correct folder names. If the function returns a blank, the root '/' must be considered as the resolved path. The result of this function cannot go below the root. Parameter DestinationPath is the path to be 'traversed' and the CurrentPath is the path where traversing starts. Always use this function to convert paths submitted by users and automatically prevent access to other folders then the ones inside the VFS.

Parameters:

Name	Type	Description
DestinationPath	string	Destination folder
CurrentPath	string	Current folder

Function GetAdminPassword

Description: Returns the password of the administrator.

The username of the administrator is fixed by the system and set to "admin".

Parameters: None

Function GetAllFlagKeys

Description: Returns all access permission flags used by the system

SL Server 4 assigns permissions per folder to each user by using flags. A user is granted a certain action if the corresponding flag is returned by function GetPermission for a particular virtual folder. The user should not be granted to execute the action if the flag does not exist for the requested folder. A great deal of permission testing is done by SL Server 4 but you may want to specify new flags (see SetSystemUser) for special functions and granting access (or not) is all up to you.

The function returns a series of characters and each character indicates the unique key of a flag. Per key you can use function GetFlagDescription to retrieve the description. See also topic permissions how to add new flags, change existing flags or remove flags.

Parameters: None

Function GetCacheEntry

Description: Returns a property from a cached item

filename - The absolute path of the cached file

filedate - The date that the file was modified

data - The content of the cached file

csize - The size of the compressed data

fsize - The file size

hits - The number of times that the file was 'hit'

age - The number of seconds the cached file wasn't 'hit'

cachedate - The date that the file was loaded by the cache

compressed - Boolean indicating compressed data

Parameters:

Name	Type	Description
Offset	numeric	Offset to the cached file
PropertyName	string	Name of a file cache property (filename, filedate, data, csize, fsize, hits, age, cachedate or compressed)

Function GetCFGKey

Description: Read a configuration key from the configuration table

Parameters:

Name	Type	Description
KeyName	string	Name of the key to retrieve

Function GetClientSessionCount

Description: Count the number of sessions one client has established

The function will count all sessions that are in use by one machine. The machine is identified by it's IP address.

Parameters:

Name	Type	Description
IPaddress	string	IP address

Function GetContent

Description: Returns the content of the HTTP request

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function GetCookie

Description: Returns a cookie-value from a string

This function does not extract a cookie directly from the request header. You must extract an array of cookies from the request header using the function GetHeaderField before you can extract a cookie value from the returned field using this function.

Parameters:

Name	Type	Description
CookieData	string	Cookie array
CookieName	string	Name of the cookie

Function GetDataIn

Description: Returns the full HTTP request

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function GetDataOut

Description: Returns the current output buffer

The function will only return the content part of the page, the header is omitted.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function GetDir

Description: Returns the content of a folder

The function returns a string containing the file index from a folder specified by variable TargetFolder where each file entry is delimited by vbCrLf (&h 0D0A). The TargetFolder variable may contain a valid mask (*.scc, *.t?t, etc). When the AbsolutePath variable is set to true any folder inside your file-system (or mapped network drives) may be read. When it is set to false any folder inside the virtual file system may be read (PIASe will automatically prefix the root-folder).

Parameters:

Name	Type	Description
TargetFolder	string	Foldername and mask
AbsolutePath	bool	Set to true when reading an absolute directory else reading virtual

Function GetDTHeight

Description: Returns the height of the server-side desktop

Parameters: None

Function GetDTWidth

Description: Returns the width of the server-side desktop

Parameters: None

Function GetFailedLogins

Description: Returns the number of failed logins from a specific session

The number of failed logins may, If values are extreme high for this session, require steps to investigate either a user having problems to login (forgot the username or password) or somebody is trying to hack the account.

Parameters:

Name	Type	Description
SessionID	string	Session ID

Function GetFileDateCreated

Description: Returns the file creation date of a file

When output directly, the format of the date and time is according your system locales. Use function Fmt to convert the date and time to another format.

The function is using the Scripting.Filesystem object and you may receive a notification from a virus-scanner that SL Server 4 might be a virus.

Parameters:

Name	Type	Description
Filename	string	The absolute path and filename of a file

Function GetFileDateLastModified

Description: Returns the file modification date of a file

When output directly, the format of the date and time is according your system locales. Use function Fmt to convert the date and time to another format.

The function is using the Scripting.Filesystem object and you may receive a notification from a virus-scanner that SL Server 4 might be a virus.

Parameters:

Name	Type	Description
Filename	string	The absolute path and filename of a file

Function GetFileExt

Description: Returns the file extension of the requested URL

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function GetFilename

Description: Returns the URL filename

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function GetFileSize

Description: Returns the size of a file in bytes

The function is using the Scripting.Filesystem object and you may receive a notification from a virus-scanner that SL Server 4 might be a virus.

Parameters:

Name	Type	Description
Filename	string	The absolute path and filename of a file

Function GetFileType

Description: Returns the file type description of a file

Parameters:

Name	Type	Description
Filename	string	The absolute path and filename of a file

Function GetFirewallEntry

Description: Returns a property from a firewall entry

Returns the following properties:

IP - Starting IP-address range

IP2 - Ending IP-address range (0.0.0.0 when blocking/passing a single IP-address)

blocked - Boolean true when the rule is blocking or false when the rule must pass any match

description - Description of the rule

Parameters:

Name	Type	Description
EntryOffset	numeric	Offset to the firewall entry
PropertyName	string	Name of the property to retrieve (ip, ip2, blocked, description)

Function GetFlagDescription

Description: Returns a flag description

The function returns the description based on the key of the flag. See also function GetAllFlagKeys how to retrieve all keys currently in use by the server.

Parameters:

Name	Type	Description
FlagKey	string	Flag key

Function GetFolders

Description: Returns a list of sub folders from the specified folder

The function returns a string containing the sub folder index from a folder specified by variable TargetFolder where each folder entry is delimited by vbCrLf (&h 0D0A). When the AbsolutePath variable is set to true any folder inside your file-system (network drives, CD-rom, etc) may be read. When it is set to false any folder inside the virtual file system may be read (PIASe will automatically prefix the root-folder).

Parameters:

Name	Type	Description
TargetFolder	string	Folder to query
AbsolutePath	bool	Set to true when reading an absolute directory else reading virtual

Function GetGlobal

Description: Returns a global variable

Parameters:

Name	Type	Description
VariableName	string	Name of

Function GetGlobalByRef

Description: Returns a global variable by reference

Use function GetGlobalCount to determine the offset of the last global variable. The offset of the global memory array starts at 1.

Parameters:

Name	Type	Description
Reference	numeric	Reference of

Function GetGlobalCount

Description: Returns the number of global variables in use

Parameters: None

Function GetGlobalNameByRef

Description: Get a global variable name by reference

Use function GetGlobalCount to determine the offset of the last global variable. The offset of the global memory array starts at 1.

Parameters:

Name	Type	Description
Reference	numeric	Reference of

Function GetGlobalRef

Description: Returns the offset of a global variable by name

Parameters:

Name	Type	Description
VariableName	string	Name of

Function GetHeaderField

Description: Returns the value of a header-field

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
HeaderFieldName	string	The name of a header-field

Function GetIP

Description: Returns the remote IP address of a client

The IP address of the remote client is retrieved from the socket. If a client in connecting trough a proxy you may need to retrieve the additional "Forwarded for" IP address that is included in the HTTP header by non-anonymous proxies to determine a valid IP address. Anonymous proxies do not forward IP addresses of their clients.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function GetIPRangeID

Description: Return network identification related to a given IP address

The file **DS.IpRangeID.dat** located in the executable folder of the server contains information and ranges of IP addresses that you can query based on an IP address using this function. If the given IP address is matching a range it will return the corresponding record. Multiple records are returned if more IP ranges were matched. Each record is terminated by an CRLF. Use function `LeftDeli(ReturnedRecords, vbCrLf)` to return a single record and `LeftDeli(Record, ";")` to return each field inside the record.

Format of the IP range table:

```
Orgname1; Place1; IpStartRange1; IpEndRange1; Tag1 <CRLF>
Orgname2; Place2; IpStartRange2; IpEndRange2; Tag2 <CRLF>
Orgname3; Place3; IpStartRange3; IpEndRange3; Tag3 <CRLF>
```

If Orgname, Place and Tag is left empty a previous occurrence will be used. In this way you can create a slim table preserving memory. The table can hold no more that 10240 records. The table will be loaded when the server is starting up and the server needs to be restarted when you modified the table.

In pratice:

```
John's network; Amsterdam; 192.168.0.10; 192.168.0.20; AMS
Alice's network; Rotterdam; 192.168.0.21; 192.168.0.30; RTD
Marcie's network; Utrecht; 192.168.0.31; 192.168.0.40; UTR
Sales network; Netherlands; 192.168.0.10; 192.168.0.40; SALES
```

`GetIPRangeID( "192.168.0.11")` will return the following records:

```
John's network; Amsterdam; 192.168.0.10; 192.168.0.20; JOHN
Sales network; Netherlands; 192.168.0.10; 192.168.0.255; SALES
```

`GetIPRangeID( "192.168.0.55")` will return the following record:

```
Sales network; Netherlands; 192.168.0.10; 192.168.0.255; SALES
```

The table and function enables you to identify networks and based on the returned records you may programmatically decide what type of service to deliver or not. For example you could add IP ranges that explicitly have access to the server while non matched IP addresses (returned as an empty string) are blocked for servicing.

Parameters:

Name	Type	Description
IP address	String	The IP address to query

Function GetMethod

Description: Returns the requested method

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function GetParam

Description: Returns a parameter value from the requested URL

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
ParameterName	string	Parameter name

Function GetParameters

Description: Returns all parameters form the requested URL

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function GetPathRights

Description: Returns a string containing flag keys assigned to a user

A user has no permission for a certain action if the corresponding key does not exist in the returned string. This function is equivalent to function GetPermission.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
Username	string	A username

FolderName	string	Absolute virtual path of the folder to test

Function GetPermission

Description: Returns a string containing flag keys assigned to a user

A user has no permission for a certain action if the corresponding key does not exist in the returned string. This function is equivalent to function GetPathRights.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
Username	string	A username
FolderName	string	Absolute virtual path of the folder to test

Function GetPiPath

Description: Returns the executable path of the SL Server 4 program

Parameters: None

Function GetPiPort

Description: Returns the TCP port number of the HTTP server

Parameters: None

Function GetPiRoot

Description: Returns the absolute path to the virtual root-folder

Pi is the project ID the initial developer Marcel Blokker gave the server engine back in 1999 when he started writing the server engine. Tough many IDE programs has been written using a different name, This function may indicate the actual usage of the same server engine.

Parameters: None

Function GetPostedParam

Description: Returns a posted parameter

Posted parameters are sent to the server using the method "POST" Note that the content of a posted parameter is Url encoded. Use function DecodeUrlEncoded to convert the content it to ASCII. When

sending parameters back to the client as part of a INPUT element your need to Url encode it using function EncodeUrlDecoded.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
ParameterName	string	Posted parameter name

Function GetPostedParamArray

Description: Returns all parameters from a parameter array

A posted parameter can contain more than one value. The function will retrieve all values delimited by the character(s) specified in parameter Delimiter. Note that the content of a posted parameter is Url encoded. Use function DecodeUrlEncoded to convert the content it to ASCII. When sending parameters back to the client as part of a INPUT element your need to Url encode it using function EncodeUrlDecoded.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
ParameterName	string	Parameter name
Delimiter	string	A string that separates each value in the returned result

Function GetProgramCaption

Description: Returns the program caption and version number as stated in fileinfo

Parameters: None

Function GetProtocol

Description: Returns the HTTP Protocol and version from the requested URL

Parameters:

Name	Type	Description

Index	numeric	Reference to a connection object

Function GetProxyPort

Description: Returns the HTTP port used to access the internet

Parameters: None

Function GetRandomNumber

Description: Generate a random number between 1 and MaxValue

Parameters:

Name	Type	Description
MaxValue	numeric	Maximum value

Sub GetScriptParameters

Description: Returns script parameters supporting the RunScript, ExecScript and ExecScriptAbs functions

Script parameters are used when (non-recursively) calling a script from another script to transfer values the way you would when calling sub's and functions.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
Parameter1	variant, optional, output	Parameter
Parameter2	variant, optional, output	Parameter
Parameter3	variant, optional, output	Parameter
Parameter4	variant, optional, output	Parameter
Parameter5	variant, optional, output	Parameter
Parameter6	variant, optional, output	Parameter
Parameter7	variant, optional, output	Parameter

Parameter8	variant, optional, output	Parameter
Parameter9	variant, optional, output	Parameter
Parameter10	variant, optional, output	Parameter

Function GetServerCaption

Description: Returns the server header-field

The header field will be sent to the client when a HTTP request is made and contains the Manufacturer, Product name, Product version, Product website and User-defined ID. To change the User-defined ID see the ServerID configuration field.

Parameters: None

Function GetServerId

Description: Returns the user-defined server ID

See also ServerID how to change the ID.

Parameters: None

Function GetServerObjectRef

Description: Get a reference from an server object

Use the 'Set' statement to link the server object to a local object variable. When ending the procedure that was using the local object variable you must destroy the local object variable before exit.

Parameters:

Name	Type	Description
Name	string	Name of the server object

Function GetServerVersion

Description: Returns the version number of the server

Parameters: None

Function GetSession

Description: Returns a session variable by name

Supply the session id (see function GetSessionID) of the current connection and the name of the variable to retrieve the value from the correct session memory block.

When a user connects to the server a session cookie will be created for the user. Every time when the user is reconnecting and sending the same session cookie a link is made to the same session memory block. This enables you to store variables and keep them shared between the different connections of one user.

One condition for maintaining session memory is the acceptance of the session cookie on initial connect at client side. When a client refuses the session cookie no service will be possible. Session cookies but also other cookies created by SL Server 4 exist only during a session at client side.

When a user is connecting via a proxy it may be possible that the proxy assigns different IP addresses between different connections. SL Server 4 considers this (odd) switching to be "IP spoofing" and creates a new session or in the worse case denies service to the client.

Parameters:

Name	Type	Description
SessionID	string	Session ID
SessionVariable	string	Name of

Function GetSessionID

Description: Returns the session id of a session

The session ID is a unique key that is generated once per session. The session ID is used as a reference to session related functions. The session ID is sent back and forth between client and server by means of a cookie. This might explain to you that a client must support cookies in order to maintain a session.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function GetSessionOffset

Description: Returns an index to a session object

Parameters:

Name	Type	Description

SessionID	string	Session ID

Function GetSessionReceived

Description: Returns the number of bytes the server has received from a client

Parameters:

Name	Type	Description
Index	numeric	Index to a connection object

Function GetSessionsCount

Description: Returns the number of active sessions

Parameters: None

Function GetSessionSent

Description: Returns the number of bytes the server has sent to a client

Parameters:

Name	Type	Description
Index	numeric	Index to a connection object

Function GetSessionsProperty

Description: Retrieves a session property

The function passes the value of the property by the parameter PropertyValue, not by the function return value. The function returns a boolean true when the property was retrieve successfully or a boolean false if it doesn't exist.

Properties:

failed - Number of failed logons

ip - The IP address of the client

id - The session ID of the client

ttl - Number of seconds before a session expires

url - The URL of the last requested resource

user - The username of the client

usertype - 0 for anonymous, 1 for a user and 2 for the admin user

bytessent - The number of bytes sent to the client

bytesreceived - The number of bytes received from the client

hits - The number of successful hits

sessionstart - Date and time the session started

sessionupdate - Date and time of the last hit

blocked - Returns a boolean true when the session is denied services otherwise a boolean false is returned

Parameters:

Name	Type	Description
SessionOffset	numeric	Offset to an entry in the session object
PropertyName	string	Name of (failed, ip, id, ttl, url, user, usertype, bytessent, bytesreceived, hits, sessionstart, sessionupdate or blocked)
PropertyValue	variant, output	Value of

Function GetSessionTTL

Description: Returns the session TTL (time to live) in seconds

Parameters:

Name	Type	Description
SessionID	string	Session ID

Function GetSessionUserName

Description: Returns the session user name

Parameters:

Name	Type	Description
SessionID	string	Session ID

Function GetSessionUserType

Description: Returns the session user type

Return values:

0 - Anonymous

1 - User

2 - Admin

Parameters:

Name	Type	Description
SessionID	string	Session ID

Function GetSessionVarName

Description: Return a session variablename by offset

Supply the session id (see function GetSessionID) of a connection and the offset pointing to the variable to retrieve the name of the variable.

Parameters:

Name	Type	Description
SessionID	string	Session ID
Offset	numeric	Reference to the session variable

Function GetSessionVarValue

Description: Returns a session variable by offset

Supply the session id (see function GetSessionID) of the current connection and the offset pointing to the variable to retrieve the value from the correct session memory block.

Parameters:

Name	Type	Description
SessionID	string	Session ID
Offset	numeric	Reference to the session variable

Function GetStrFileExt

Description: Returns the file extension of a filename

Returns the bare extension only without a period. Only the last part will be returned when a filename contains more than one period. A blank will be returned when a filename has a blank extension or no extension at all.

Parameters:

Name	Type	Description
Filename	string	A valid dos/ntfs/virtual filename

Sub GetSystemUser

Description: Returns a user record based on the user name

Returns a user record based on the user name (UserName parameter). The Failed parameter returns a boolean true when the user record could not be retrieved successfully.

Parameters:

Name	Type	Description
Offset	numeric, output	Reference to an entry in the user table
UserName	string, input	Username
Fullname	string, output	Full name of the user
Information	string, output	Additional user information
UserPassword	string, output	Password
Permissions	string, output	Permissions

Failed	bool, output	True when a user doesn't exist

Function GetSystemUserCount

Description: Returns the number of user records

Parameters: None

Sub GetSystemUserENum

Description: Returns a user record based on record offset

Enumerates the user table based on a record offset (Offset parameter) rather than a username. Determine the number of records with function GetSystemUserCount. The offset of a user record starts on 1 and may not exceed the maximum count. The Failed parameter returns a boolean true when the user record could not be retrieved successfully.

Parameters:

Name	Type	Description
Offset	numeric	Reference to an entry in the user table
UserName	string, output	Username
Fullname	string, output	Full name of the user
Information	string, output	Additional user information
UserPassword	string, output	Password
Permissions	string, output	Permissions
Failed	bool, output	True when a user doesn't exist

Function GetUDPInterval

Description: Returns the assigend UDP loopback interval in miliseconds

Parameters: None

Function GetUDPPort

Description: Returns the UDP listener port number

Parameters: None

Function GetURL

Description: Returns the requested URL

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function GetUrlNoParm

Description: Returns the requested URL without any parameters

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function GetURLPath

Description: Returns the virtual folder of the requested URL without filename and parameters

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Sub GetUserPassB64

Description: Extract a Base64 encoded authentication string

The method returns a username and password from a string that was base64 encoded according the USERNAME:PASSWORD format that is used by the Base64 authentication method submitted by the client as a header field.

Parameters:

Name	Type	Description
Base64AuthHeader	string	The base64 formatted auth header

Username	string, output	Username
Password	string, output	Password

Function GetVirtualSource

Description: Returns the absolute path and filename of a mapped virtual file

A virtual source is a file that doesn't exist in the root folder (or sub folders) of SL Server 4. By mapping the file using function SetVirtualSource you are able to mount the file from any disk and hook it into the virtual file system.

Parameters:

Name	Type	Description
VirtualFilename	string	Path and filename of the virtual file

Sub HOut

Description: Add a string terminated by a BR to the HTTP output buffer

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
HTTPOutputData	string	Data to add

Function HttpPending

Description: Checks whether the HTTP proxy function still is pending

Checking the pending status of the connection that the current script is using will always return a boolean false because the HTTP proxy function will not return until the resource is loaded or when a timeout has occurred. You can only check the pending status of other pending connections. The function returns a boolean true when the proxy is pending or a boolean false when the proxy has completed the call.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function HttpProxy

Description: Returns a resource from a remote web server

Enables you to retrieve data from another web server. The proxy function does not support SSL. Any redirection or authentication request from the remote server must be handled manually to end-up at the initially requested resource. Enable permission flag "access proxy" at root for user anonymous (all users) or any specific user you want to grant the usage of this function.

Note that the remote cookies field is replaced by the cookies field of the local request to maintain local session recognition. The replaced remote cookies field is saved per server and used again when the next requests goes out to maintain the remote session. The swapping of the cookies fields enables you to feed the result directly to the client without modification of any header fields to maintain the local session. See also the experimental proxy server script (/admin/server/proxy.os) how to implement such direct feeds for proxy serving.

The function will not return unless the resource has been loaded or when a timeout has occurred.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
HTTPportNumber	numeric	Remote port to access
Server	string	Name or IP address of the server
VirtualPath	string	Remote folder
Document	string	Page to retrieve
Method	string	HTTP method (get or post)
PostedData	string	Any posted data required to support the request
sHeaderData	string, optional	Any required header
sAgent	string, optional	User-agent

ConnectionTimeOut	numeric. optional	Connection timeout value in miliseconds
Retries	numeric, optional	Number of retries after a timeout
TransferTimeOut	numeric. optional	Transfer timeout (time without receiving any data) value in seconds

Function IIF

Description: Immediate IF

Returns the TruePart expression when expression Expr evaluates True otherwise the FalsePart expression will be returned.

Parameters:

Name	Type	Description
Expr	expression	Expression to evaluate
TruePart	optional, expression	Result to be returned if Expr is True
FalsePart	optional, expression	Result to be returned if Expr is False

Sub Incr

Description: Increments a numeric variable

Parameters:

Name	Type	Description
Variable	numeric	The variable to increment

Function InFileCache

Description: Search for a file in the cache

Returns a zero when the file was not found or a non-zero offset address to a slot in the file cache.

Parameters:

Name	Type	Description
Filename	string	The absolute path filename of the file to test

Function IPInstances

Description: Returns the number of connections a client has established

Use this function to determine the number of connections a client has established and to decide whether a client is 'abusing' the service.

Parameters:

Name	Type	Description
ClientIPaddress	string	IP address of the connections to count

Function IpMatched

Description: Compare a IP address with a masked IP address

Use a * to mask IP subfields. For example, the function will return true when IP addresses 1.2.3.4, 4.5.3.4 and 9.8.3.4 are compared with the IP mask *.*.3.4. The function will return false when the IP mask is set to *.*.5.6.

Parameters:

Name	Type	Description
ClientIPaddress	string	IP address to compare
MaskedIPrange	string	IP address range to compare against

Function IpRangeMatched

Description: Compares a IP address against a range

Returns a boolean true when a IP address is in range compared to the given IP address range and a boolean false when the IP address is outside the scope of the range.

Parameters:

Name	Type	Description
IpCompare	string	IP-address to compare
IpStartRange	string	IP-address start-range
IpEndRange	string	IP-address end-range

Function IsBlocked

Description: Returns the blocking-state of an IP address

Returns a boolean true when the firewall is blocking the given IP address or a boolean false when the firewall allows the IP address to pass.

Parameters:

Name	Type	Description
ClientIPaddress	string	IP address of the connections to test

Function IsConnected

Description: Returns the connection status of a client

When a script file has a long processing time you might want to check the connection status of the client to determine whether you need to abort the execution of the script. The function returns a True value when the client is still connected and False when the client has disconnected.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function IsInFireWall

Description: Checks for a rule that applies exclusively to a IP address

Returns a boolean true when the first IP property of a rule matches the given IP address (either blocking or passing). This function does not search the ip2 property, it is excluded from the search.

Parameters:

Name	Type	Description
ClientIPaddress	string	IP address to test

Function IsSystemUser

Description: Authenticate a user

Use this function to authenticate users listed in the user table. The function will return a boolean true when the credentials match a user in the user table. Note that the administrator credentials cannot be checked using this function because the administrator credentials are not maintained in the user table. Valdate the administrative user by using the AdminLoginB64 or GetAdminPassword functions.

Parameters:

Name	Type	Description
Username	string	Username
Password	string	Password

Function IsSystemUserB64

Description: Returns True when an auth header matches an entry in the user table

When a client has sent it's credentials using the Base64 authentication method, you may authenticate the retrieved Base64 encoded field-value directly without converting it to ASCII first.

Parameters:

Name	Type	Description
Base64AuthHeader	string	The base64 formatted auth header

Function IsValidID

Description: Checks whether a connection has a valid Session ID

This function is used to determine whether a connection bound to a valid session and returns a boolean true when it does or a boolean false when it doesn't. If the function returns false, A possible

attempt to spoof the session ID is in progress and service should be aborted. See also the configuration field CheckIPSecurity how to control the behavior of the server in this matter.

Parameters:

Name	Type	Description
Index	numeric	Index to a connection object

Function LeftDeli

Description: Returns a sub-string starting from the left

The function returns the left part of a string until a delimiter is encountered. The delimiter is not included and will be removed from the source unless the boolean parameter KeepSource is set to true.

Parameters:

Name	Type	Description
SourceText	string, output	Source
Delimiter	string	Delimiter
KeepSource	optional, bool	Do not remove the parsed result from the source when set to true

Sub LockIP

Description: Lock an IP address

Add or set a single IP address for blocking. This function does not add a range or description to the firewall entry. Use the AddIP2 function to add ranges and/or descriptions.

Parameters:

Name	Type	Description
ClientIPaddress	string	IP address of a client

Sub LockUser

Description: User lockout

Use this function to lockout a user during the current server run. This function works only for logged on users. When the server is restarted all locks are cleared.

Parameters:

Name	Type	Description
Username	string	Username of the user to lockout

Sub LOut

Description: Add a string terminated by a CRLF to the HTTP output buffer

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
HTTPOutputData	string	Data to add

Function MakeFolder

Description: Creates a new folder

Returns zero when succeeded or a non-zero number when the function failed to create the new folder.

Parameters:

Name	Type	Description
NewFolderName	string	The absolute name of the new folder to create

Function MakeServerDate

Description: Create a server date field

Returns a current date / time string formatted according the HTTP header specifications.

Parameters:

Name	Type	Description
Date	string	Date and time

Sub MakeSfwFile

Description: Create a encrypted script file

This function provides a script file not readable or editable using the SHA 160 bits encryption algorithm. The server still is able to run the script and pre-processing of the script becomes obsolete. You do not need to provide a password for this type of encryption. This type of file cannot be exchanged between different instances of SL Server 4.

WARNING: This function will overwrite your source-file. Make a copy of your source file before you apply this function!

Parameters:

Name	Type	Description
ScriptFilename	string	Absolute path and filename of the script to process

Function Md5AuthLogin

Description: Returns True when MD5 Authentication is required by configuration

When the function returns True, a submitted authentication response must be formatted using the MD5 digest method. A client however may choose to use the Base64 method whenever the client does not support MD5 digest method. Even when Base64 does not provide a secure login method, the server must (by protocol standards) accept the Base64 method and thus authenticate the client regardless the authentication method.

Parameters: None

Function MD5Hash

Description: Returns the RSA MD5 Hash of an input string

MD5 is an excellent authentication method when writing scripts supporting their own secure login. A sample how to use RSA's MD5 login at client and server side can be found in the RADM scripts: /radm/login.ssc. This function is also used by SL Server 4 to support secure client logon.

Parameters:

Name	Type	Description

InputString	string	The data to hash

Sub MouseLeftClick

Description: Emulate a left-mouse-click event

This function is only applicable at server side and sends the mouse message to the current active window.

Parameters:

Name	Type	Description
AbsX	numeric	Absolute X coordinate
AbsY	numeric	Absolute Y coordinate

Sub MouseLeftDown

Description: Emulate a left-mouse-down event

This function is only applicable at server side and sends the mouse message to the current active window.

Parameters:

Name	Type	Description
AbsX	numeric	Absolute X coordinate
AbsY	numeric	Absolute Y coordinate

Sub MouseLeftUp

Description: Emulate a left-mouse-up event

This function is only applicable at server side and sends the mouse message to the current active window.

Parameters:

Name	Type	Description

AbsX	numeric	Absolute X coordinate
AbsY	numeric	Absolute Y coordinate

Sub MouseRightClick

Description: Emulate a right-mouse-click event

This function is only applicable at server side and sends the mouse message to the current active window.

Parameters:

Name	Type	Description
AbsX	numeric	Absolute X coordinate
AbsY	numeric	Absolute Y coordinate

Sub MouseRightDown

Description: Emulate a right-mouse-down event

This function is only applicable at server side and sends the mouse message to the current active window.

Parameters:

Name	Type	Description
AbsX	numeric	Absolute X coordinate
AbsY	numeric	Absolute Y coordinate

Sub MouseRightUp

Description: Emulate a right-mouse-up event

This function is only applicable at server side and sends the mouse message to the current active window.

Parameters:

Name	Type	Description
AbsX	numeric	Absolute X coordinate
AbsY	numeric	Absolute Y coordinate

Function MouseX

Description: Returns the absolute current mouse column

This function returns only the mouse coordinate from server side.

Parameters: None

Function MouseY

Description: Returns the absolute current mouse row

This function returns only the mouse coordinate from server side.

Parameters: None

Sub MoveFwPriorityDown

Description: Move a firewall-rule one row down

The firewall evaluates the rules-list from top till bottom. If an IP address is not filtered by a list rule, the default option is used. If an IP address is matched by more than one rules, the result of the last matching rule is the final result of the query. Moving the rules up and down the list will change the evaluation order of the list.

For example: If you add a non-blocking rule for domain 192.160.*.* but you wish to block station 192.160.1.5 you must move the second rule to a position that is after the non-blocking first rule. In that case all IP addresses of domain 192.160.*.* may pass with the exception of IP address 192.160.1.5.

If you would swap these rules the outcome would be initial a block on IP address 192.160.1.5 but by evaluating the second rule (192.160.*.*) it would allow the same IP address to connect since that is the result of the last evaluation.

Parameters:

Name	Type	Description
Offset	numeric	Offset to the firewall-rule

Sub MoveFwPriorityUp

Description: Move a firewall-rule one row up

The firewall evaluates the rules-list from top till bottom. If an IP address is not filtered by a list rule, the default option is used. If an IP address is matched by more than one rules, the result of the last matching rule is the final result of the query. Moving the rules up and down the list will change the evaluation order of the list.

For example: If you add a non-blocking rule for domain 192.160.*.* but you wish to block station 192.160.1.5 you must move the second rule to a position that is after the non-blocking first rule. In that case all IP addresses of domain 192.160.*.* may pass with the exception of IP address 192.160.1.5.

If you would swap these rules the outcome would be initial a block on IP address 192.160.1.5 but by evaluating the second rule (192.160.*.*) it would allow the same IP address to connect since that is the result of the last evaluation.

Parameters:

Name	Type	Description
Offset	numeric	Offset to the firewall-rule

Sub OpenConsoleWindow

Description: Opens the console window when the IDE is hidden

This function provides access to the IDE when the windows start bar is ignoring mouse messages addressed to the system tray. This will happen sometimes and may indicate that the start bar, the desktop program or another program having it's handle hooked to the system tray is in a possible crash state. Write a small script using a normal editor and save it to the admin folder of the virtual root. Open your browser and surf to the virtual location of the script file to open the IDE. Save your work (if any), shutdown SL Server 4 and Restart windows when this happens.

Parameters: None

Sub Out

Description: Add a string to the HTTP output buffer

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
HTTPOutputData	string	Data to add

Function Param

Description: Returns a parameter value from a url formatted string

URL Parameters in a URL formatted string specified in UrlFormattedString can be extracted by specifying the name of the URL parameter in ParameterName.

Parameters:

Name	Type	Description
UrlFormattedString	string	Url formatted string
ParameterName	string	Name of

Function PiBuild

Description: Returns the application build number

Parameters: None

Sub PlaySound

Description: Playback sound file

This will playback media files at server side. De media is played in the background when PlayInBackGround is set to true.

Parameters:

Name	Type	Description
SoundFile	string	Absolute path nd filename of the soundfile
PlayInBackGround	optional, boolean	Playback of media in the background when set to True

Function PortStatus

Description: Returns the connection status

The following codes are returned:

0 - The connection is idle

1 - The connection is receiving data

2 - The connection is transmitting data

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function ProcessScriptCode

Description: Executes a string containing PIASe script and returns the result

SL Server 4 is able to run script dynamically. For example, you could build script using another script, store script in a database, load and run it -or- let a client use the post method to submit a script fragment and run it. The HTTP request data (the HTTP header that was sent by the client) will be inherited from the caller (the parent script) and will be available for normal script processing. This function is very powerful but also dangerous when you implement the function as a interface for your client without sufficient security. Lack of security could compromise your file system and ultimately your entire system and network. Use with extreme caution!

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
ScriptCode	string	Scriptcode

Function ProcessScriptFile

Description: Executes a file containing PIASe script and returns the result

SL Server 4 is able to run script dynamically. For example, you could build script using another script, save the script to a file and run it -or- let a client use the post method to submit a script fragment, save it to a file and run it. The HTTP request data (the HTTP header that was sent by the client) will be inherited from the caller (the parent script) and will be available for normal script processing. This function is very powerful but also dangerous when you implement the function as a interface for your client without sufficient security. Lack of security could compromise your file system and ultimately your entire system and network. Use with extreme caution!

Parameters:

Name	Type	Description

Index	numeric	Reference to a connection object
ScriptFilename	string	Name of the file to process

Function QuickSortStr

Description: Returns a ascending sorted string

Each item must be delimited with a unique character such as CRLF. The function will return a sorted string using the same delimiter. Use function LeftDeli to extract the items ascending or RightDeli to extract the items ascending.

Parameters:

Name	Type	Description
UnSortedString	string	Delimited items
Delimiter	string	Delimiter

Function RAdmin

Description: Returns the remote administrator policy

Returns a boolean true when the admin user may access the server remotely or false when it is denied by configuration. See also RemoteAdministration to set this policy. It's for safety reasons not advised to allow the admin user to manage the server remotely when the server is taking requests from the internet directly.

Parameters: None

Function ReadFile

Description: Read the full content of a file into a string

Returns zero when succeeded or a non-zero number when failed. Specify the absolute path of the file to be loaded in variable Filename. De variable FileData contains the full content of the file. It is not recommended to load extreme large files due to high memory consumption.

Parameters:

Name	Type	Description
Filename	string	Absolute path and filename of the file to read and returns zero

		if succeeded
FileData	string, output	The content of a file

Sub Redirect

Description: Prepares the server to redirect

Prepares a redirection response header (status 304, Moved temporarily) requesting the client to load another location. The location may be any HTTP or client compatible location. A exit sub statement is required following the call.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
UrlToRedirectTo	string	Redirection URL

Function ReleaseCache

Description: Release the less used cached file

When the file cache is full you may release the less used file (lowest number of hits) from the cache. The function returns an offset to the freed slot.

Parameters: None

Function RenameFile

Description: Rename a file

Returns a zero when succeeded or a non-zero number when failed. When renamed on the same disk volume a move will be applied. When a mask is specified a multiple rename is applied [n = RenameFile("*.txt", *.ssc")].

Parameters:

Name	Type	Description
SourceFilename	string	Absolute path and filename of the file to rename
RenameToFilename	string	New absolute path and filename

Function RenameFile

Description: Rename a file

Returns a zero when succeeded or a non-zero number when failed. When renamed on the same disk volume a move will be applied. When a mask is specified a multiple rename is applied [n = RenameFile("*.txt", *.ssc")].

Parameters:

Name	Type	Description
SourceFilename	string	Absolute path and filename of the file to rename
RenameToFilename	string	New absolute path and filename

Function Repl

Description: Replace a sub string with new sub string

Specify the string containing the items to be replaced in Target, The SearchFor parameter with the string you want to replace and the ReplaceWith parameter with the replacement string. Prevent an empty SearchFor parameter or values that may cause a endless loop.

Parameters:

Name	Type	Description
Target	string	String to search and replace
SearchFor	string	String to look for
ReplaceWith	string	Replacement

Sub ResizeImage

Description: Resize an image

Loads any BMP or JPeg image and resizes it. The output file will always be in the JPeg format using a 70% compression ratio.

Parameters:

Name	Type	Description
Width	numeric	Output width, If 0 the width is relative to height
Height	numeric	Output height, If 0 the height is relative to width
InputFilename	string	Name of the image to resize
JPegOutputFile	string	Name of the resulting bitmap image

Function ResolveHost

Description: Returns the hostname of a machine

Used to determine the FQDN (Fully Qualified Domain Name) of a machine based on the IP address of the machine.

Parameters:

Name	Type	Description
IpAddress	string	IP address of the domain to resolve

Function ResolveIP

Description: Returns an IP address of a machine

Used to determine the IP address of a machine based on the FQDN (Fully Qualified Domain Name) of the machine.

Parameters:

Name	Type	Description
DomainName	string	The name of the domain (FQDN)

Function RightDeli

Description: Returns a sub-string starting from the right

The function returns the right part of a string until a delimiter is encountered. The delimiter is not included and will be removed from the source unless the boolean parameter KeepSource is set to true.

Parameters:

Name	Type	Description
SourceText	string, output	Source
Delimmiter	string	Delimiter
KeepSource	optional, bool	Do not remove the parsed result from the source

Sub RShell

Description: Executes a program in a shell at server side

It is not advised to run a shell unless you know that the application can run in a unattended environment, behave as predicted, has not output to the UI (messages, etc) and terminates as predicted regardless any error state the application may return.

Parameters:

Name	Type	Description
ShellArgument	string	Shell argument
WindowState	numeric, optional	Type of window
WaitForCompletion	boolean, optional	Wait for the completion of the shell (default: True)

Sub RunScript

Description: Stop executing the current script and start another one

Note: An 'Exit Sub' statement is required following the RunScript call.

Parameters:

Name	Type	Description

Index	numeric	Reference to a connection object
ScriptFilename	string	Script file to run

Function SaveFile

Description: Save the content of a string to a file

Returns a zero when succeeded or a non-zero number when failed. The function will overwrite any existing file.

Parameters:

Name	Type	Description
Filename	string	Absolute path and filename of the file
FileData	string	New file contents

Function SeekExactFirewallEntry

Description: Returns the offset to a entry in the firewall table

Returns the offset to a firewall entry when the first IP property of a rule matches the given IP address (either blocking or passing). This function does not search the ip2 property, it is excluded from the search.

Parameters:

Name	Type	Description
ClientIPaddress	string	IP address to look for

Function SeekFirewallEntry

Description: Returns the offset to a entry in the firewall table

Returns the offset to a firewall entry when the first field of a rule matches the given masked IP address (either blocking or passing). This function does not search the ip2 property, it is excluded from the search.

Parameters:

Name	Type	Description
ClientIPaddress	string	Masked IP address range to look for

Function SeekReadFile

Description: Reads a portion of a file

Returns zero when succeeded or a non-zero number when failed. The function returns -1 when the Offset variable is exceeding the length of the file. The minimal value to be used with the Offset variable is 1. The data read by the function will not be cached by the file cache.

Parameters:

Name	Type	Description
Filename	string	Absolute path and filename
Offset	numeric	The absolute position in the file
Size	numeric	The number of bytes to read
DataToRead	string	The portion of the file

Function SendMail

Description: Send an email message

This function will send an email message using a standard SMTP server on port 25 of the indicated SMTP server. Use this procedure to submit administrative messages concerning server / user state. The function returns True when succeeded and False when it could not deliver the message.

If you do not wish to attach files, set parameter AttFile to an empty string (“”). Separate multiple files with a semicolon (“;”) if you want to attach more than one file. If a file is not found, the function will return False and the message will not be submitted.

Name	Type	Description
To	string	Email address of the recipient
From	string	Email address of the sender
Subject	string	Subject of the message

Message	string	Message body
AttFiles	string	Absolute path and filenames of files to be attached
SMTPServer	string	Name or IP address of the SMTP server
Username	string	SMTP username
Password	string	SMTP password

Function SendPort

Description: Stream raw data out

Submits data directly to the current open TCP port. The HTTP engine will be by-passed and you must submit the page including protocol, location, status, header, header fields and authorization. Before the transmission you need check whether the client still is connected using the IsConnected function and whether the connection is not sending or receiving data using the PortStatus function. A TerminateRequest is required to complete the transmission.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
StrData	string	String containing the data

Function ServerDate

Description: Calculate a server date field

Returns a calculated date / time string formatted according the HTTP header specifications.

Parameters:

Name	Type	Description
Days	numeric	Number of days to add (negative to substract)

Function SessionExpired

Description: Returns True if a session has expired

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function SessionVarCount

Description: Returns the number of session variables in use

Supply the session id (see function GetSessionID) of the current connection to retrieve the variable count from the correct session memory block.

Parameters:

Name	Type	Description
SessionID	string	Session ID

Function SessionVarOffs

Description: Returns a offset of a session variable by name

Supply the session id (see function GetSessionID) of the current connection and the name of the variable to retrieve the offset from the correct session memory block.

Class: Session memory

Parameters:

Name	Type	Description
SessionID	string	Session ID
VariableName	string	Name of

Sub SetAdminPassword

Description: Changes the administrator password

Parameters:

Name	Type	Description
AdministratorPassword	string	The new password

Sub SetCache

Description: Add or refresh cached item

Normally the cache will add or refresh cached items automatically using this function and it is not necessary to do this manually.

Parameters:

Name	Type	Description
Offset	numeric	Offset to a file cache slot
FileToCache	string	Absolute path and filename of the file to cahce
ContentOfTheFile	string	Content of the file
Compiled	bool, optional	Set to True if the file is a compiled result of the PIASe engine

Sub SetContentType

Description: Set the content-type

Content types most commonly used by browsers and supported by SL Server 4:

text/html, image/jpeg, image/gif, application/zip and application/octet-stream.

Set the correct type depending the object you want to submit. Please lookup manufacturers specifications when using special object formats such as office documents or postscript files.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
ContentType	string	Content type

Sub SetCookie

Description: Set or change a cookie-value

Set or change a cookie value of the response header. Cookies are related to the root (/) of the server. Note that SL Server 4 cookies are only maintained by the client during a session. Most clients (browsers) are blocking persistent cookies by default and it would not be advised to use persistent cookies to store values for the long term. If you need to maintain some sort of history, create an account and relate (database) data to a unique user name or construct the response header manually.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
CookieName	string	Name of the cookie
CookieValue	string	New value of the cookie

Sub SetDataOut

Description: Output all content at once

The protocol, location, status, header and header fields will be added by the system. The function overrides all existing output.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
HTTPOutputData	string	Data to send

Sub SetDataRawOut

Description: Output all data at once

You must supply the protocol, location, status, header, header fields (including authorization fields), cookies (session ID) and the HTTP content.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
HTTPOutputData	string	Full HTTP envelope (Header and Page)

Function SetExpired

Description: Set the expiration date of a Page, a Picture or other object

The output will expire immediately when specifying a zero or negative number in the Days parameter.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
Days	numeric	Days to add or subtract counted from now

Sub SetExtraHeaders

Description: Set additional response headers

Because header fields are used to inform the client program about the content you may add extra header fields when needed. Each name-value pare must be delimited by a CRLF (&h 0D0A and you can't override any header that is set exclusively by the server engine.

Class: HTTP output

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
HeaderData	string	Header field(s) including CRLF per name-value pair

Sub SetFileDateCreated

Description: Set the creation date/time of a file

Parameters:

Name	Type	Description
Filename	string	Absolute filename
Year	numeric	Year
Month	numeric	Month
Day	numeric	Day
Hour	numeric	Hour
Minute	numeric	Minute
Second	numeric	Second

Sub SetFileDateLastModified

Description: Set the modification date/time of a file

Parameters:

Name	Type	Description
Filename	string	Absolute filename
Year	numeric	Year
Month	numeric	Month
Day	numeric	Day
Hour	numeric	Hour
Minute	numeric	Minute
Second	numeric	Second

Sub SetGlobal

Description: Create or Set a global variable

Global memory can be accessed by all sessions and if the boolean Static parameter is set to true, be persistent (static) between different runs of the server. Global memory has 10240 memory slots available for storage. It's not advised to use static global memory as your database because it will slow down overall performance.

Parameters:

Name	Type	Description
VariableName	string	Name
Value	string	The value of
PreserveDB	optional, bool	preserve static in datatbase

Sub SetGlobalByRef

Description: Set a global variable by reference

Use this function to set an existing global variable by referring by it's offset rather than it's name. Global memory can be accessed by all sessions and if initialized be persistent (static) between different runs of the server. Global memory has 10240 memory slots available for storage.

Parameters:

Name	Type	Description
Reference	numeric	Reference of
NewValue	string	Value of

Sub SetGlobalNameByRef

Description: Set a global variable name by reference

Use this function to change the name an existing global variable by referring by it's offset. Use this function with care, SL Server 4 will not check for duplicate names. Global memory can be accessed by all sessions and if initialized be persistent (static) between different runs of the server. Global memory has 10240 memory slots available for storage.

Parameters:

Name	Type	Description
Reference	numeric	Reference of
NewName	string	Value of

Sub SetIncommingContent

Description: Change the full content of a incoming response

Use this function when the content of a request needs adjustments (most likely retrieved and set in the BeforeHTTP event procedure) before it is processed by the TCPRequest event procedure.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
NewContent	string	Replacement or modified content

Sub SetLocation

Description: Change the Location-header field

This function is used by the server engine when proxy serving data or running inline script and has no effect on standard browsers or clients. Normally the location as stated in the status (first line of the header) must be the same as the location stated in the location header field.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
NewURLlocation	string	New URL

Sub SetScriptParameters

Description: Set the script parameters to support the RunScript function

Script parameters are used when (non-recursively) calling a script from another script to transfer values the way you would when calling sub's and functions.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
Parameter1	variant, optional	Parameter
Parameter2	variant, optional	Parameter
Parameter3	variant, optional	Parameter
Parameter4	variant, optional	Parameter
Parameter5	variant, optional	Parameter
Parameter6	variant, optional	Parameter
Parameter7	variant, optional	Parameter
Parameter8	variant, optional	Parameter
Parameter9	variant, optional	Parameter
Parameter10	variant, optional	Parameter

Sub SetSession

Description: Add or set a session variable

Supply the session id (see function GetSessionID) of the current connection and the name of the variable to set the value in the correct session memory block. Session variables are of the variant type and can be used to store numeric, boolean or string values during a session.

When a user connects to the server a session cookie will be created for the user. Every time when the user is reconnecting and sending the same session cookie a link is made to the same session memory block. This enables you to store variables and keep them shared between the different connections of one user.

One condition for maintaining session memory is the acceptance of the session cookie on initial connect at client side. When a client refuses the session cookie no service will be possible. Session cookies but also other cookies created by SL Server 4 exist only during a session at client side.

When a user is connecting via a proxy it may be possible that the proxy assigns different IP addresses between different connections. SL Server 4 considers this (odd) switching to be "IP spoofing" and creates a new session or in the worse case denies service to the client.

Parameters:

Name	Type	Description
SessionID	string	Session ID
VariableName	string	Name of
Value	variant	Value of

Sub SetSessionTTL

Description: Set the session TTL

Parameters:

Name	Type	Description
SessionID	string	Session ID
TimeToLive	numeric	Number of seconds before a session ends

Sub SetSessionUserName

Description: Set the session username

Parameters:

Name	Type	Description
SessionID	string	Session ID
Username	string	Username

Sub SetSessionUserType

Description: Set session user type

Parameters:

Name	Type	Description

SessionID	string	Session ID
UserType	numeric	Usertype (0=Anonymous, 1=User, 2=Admin)

Function SetSystemUser

Description: Add or set a user record

If the user (Username parameter) does not exist a new record will be added to the user table. Specify the full name of the user (FullName parameter), Some (but not required) personal information (Information parameter), A password (UserPassword parameter) and the permission flags (Permissions parameter). The function will return a boolean true when changes or additions are executed successfully.

Permissions

The permissions parameter contains a series of comma delimited records. Each record contains a series of permission flags and a folder name using the following format: abc:/folder/[*] where "abc" represents the permission flags, "/folder/" the folder name on which the permissions should be applied and optionally (don't include the brackets) a "*" indicating that child folders must inherit the permissions too.

The record set "rx:/help/*, r:/help/read/" means that a user may read files (r), run scripts (x) inside the /help/ folder and all child folders (*) of the /help/ folder except for the /help/read/ folder, Only the reading of files (r) is permitted here effectively revoking the run scripts (x) permission.

See also permissions in the Configure users manual.

Parameters:

Name	**Type**	**Description**
UserName	string	Username
Fullname	string	Full name of the user
Information	string	Additional user information
UserPassword	string	Password
Permissions	string	Permissions

Sub SetVirtualSource

Description: Maps a file to a virtual location

You could map media files stored on a CD for example to a location inside the virtual file system of the server. In that way a client can access the media by addressing the virtual location of the media without accessing physically the CD-rom player. Virtual mapping will be lost when the server is shut down and need to be re-mapped again on restart.

A maximum of 1024 'external' files can be mapped.

Parameters:

Name	Type	Description
VirtualFilename	string	The virtual path and filename
AbsoluteFilename	string	The absolute path and filename of the file to be mapped

Sub ShareEvents

Description: Share events with other processes

Use this function when running scripts that a polling for a long time (over one second) to enable other scripts to run too.

Parameters:

Name	Type	Description
Seconds	numeric, single, optional	Number of seconds to wait (min 0.10) If ommited, 128 "DoEvent" messages will be sent

Sub ShowDebug

Description: Show debug window

The debug window is the immediate window where you can print results from script expressions for testing purposes. If you need to see results from the debug window without opening or locating the debug window this method will open it as the top-most window. This function works only in the IDE edition.

Parameters: None

Sub Speak

Description: Speak using Microsoft Speech Synthesizer

This function is only applicable at server side and requires Microsoft speech synthesizer to be installed. The default agent is used to synthesize the string. You can download the SDK from Microsoft's website if you don't have the synthesizer installed on your system.

Parameters:

Name	Type	Description
TextToSpeak	string	Text to speak

Sub StartEditor

Description: Opens the IDE editor

Works only with the IDE version. Check before you open the editor that the request came from the local server.

Parameters:

Name	Type	Description
Filename	string	Virtual path of the document

Function StrVal

Description: Convert a string value to a numeric value

Non numeric charachters will result in zero.

Parameters:

Name	Type	Description
StrData	string	String containg numeric data

Function TcpClearData

Description: Clear the current TCP queue

Use this function to clear the TCP queue after you retrieved data from it with function TcpGet or TcpGetData.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function TcpClose

Description: Close the TCP connection

This function will close the socket. Use when transactions are complete. Make sure the socket is closed before the script is ending.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Function TcpContinue

Description: Send a next portion of data to the current TCP connection

This function will enable you to create TCP transactions using multiple (local) requests. Most likely you will call this function in a looping structure and it is advised to call function ShareEvents frequently preventing traffic jam at the HTTP and UDP ports.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
PostedData	string	Any posted data required to support the request

Function TcpGet

Description: Returns raw TCP data from a remote server

This function is used to initialize a raw (not HTTP protocol bound) TCP connection with another server (Server parameter). Because the function is not bound to the HTTP protocol you may specify any port (PortNumber parameter) to connect with. Most TCP servers expect some sort of data (PostedData parameter) to initialize the connection such as a login or a request command. You could

either wait for a response to arrive (returned by the function) or exit the function (WaitForResult parameter) and poll for data when it becomes available using function TcpGetData. Use function TcpContinue when a transaction require multiple submission of data between remote responses.

To make sure that the function does not end up in a stall you may specify the time a connection attempt may take (ConnectionTimeOut parameter), The number of connection retries (Retries parameter) and the time a transfer may stay idle measured from the last received packet (TransferTimeOut parameter). The timing and retry parameters are optional and if omitted default values are used.

When looped TCP transactions run for a long time it is advised to call function ShareEvents frequently to prevent traffic jam at the HTTP and UDP ports.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object
PortNumber	numeric	Remote port to access
Server	string	Name or IP address of the server
PostedData	string	Any posted data required to support the request
WaitForResult	boolean	Force TcpGet to wait for any result and close the connection
ConnectionTimeOut	numeric. optional	Connection timeout value in miliseconds
Retries	numeric, optional	Number of retries after a timeout
TransferTimeOut	numeric, optional	Transfer timeout (time without receiving any data) value in seconds

Function TcpGetData

Description: Retrieve the current TCP queue

Clear the TCP queue using function TcpClearData when the data is retrieved or wait and poll later when the data is only partially arrived. In any case you must clear the queue when the received data appears to be complete.

Do not keep stacking the queue when receiving large amounts of data but retrieve it, clear the queue and save the received data to a disk using the AppendFile function. SL Server 4 can handle large queues but performance will be less or even down to zero when windows need to page-swap memory because of extreme stacking.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Sub TerminateRequest

Description: Terminate a request

When it is not possible to produce standard HTTP output using the server-engine or the client has no permission to connect or request a certain resource you may terminate the request using this method. An "exit sub" must follow the preceding call.

Parameters:

Name	Type	Description
Index	numeric	Reference to the connection object

Sub TerminateSession

Description: Terminate and cleanup a session

Removes a session from memory freeing up a session slot for the next one. Sometimes sessions may take a long time to expire when the session TTL (Time-To-Live) is set to extreme high values and it may be necessary to remove it rather than waiting for removal due to expiration.

Parameters:

Name	Type	Description
SessionID	string	Session ID

Function ToHex

Description: Convert a string to a hexadecimal string

Note that per charachter 2 hexadecimal digits will be used.

Parameters:

Name	Type	Description
Chars	string	Char based input string

Function TraceRt

Description: Returns trace route information

Returns a CRLF delimited string containing the status, time (milliseconds) and IP address of the specified hop number.

The status can be 'OK' indicating that the router returned a response or 'TIMEOUT' indicating that there wasn't a response. Any other status in indicates a non existing router of error in common. A TIMEOUT status does not imply a non existing router and may return a valid IP address. Prevent hopping to the same IP address by breaking the trace loop.

If the returned IP address is valid (status 'OK' or 'TIMEOUT') you must execute the next hop by increasing the hop number specified by parameter HopNumber. A hop number starts always on 1. Never use the returned IP as your starting IP address specified by parameter IpAddress.

Use functions Ping and ResolveHost to complete the route information.

Class: Network

Parameters:

Name	Type	Description
IpAddress	string	IP address of the remote site
HopNumber	numeric	The hopnumber of the route

Function UDPpoke

Description: Broadcast a UDP message

The User Datagram Protocol or UDP for short provides a procedure for application programs to send messages to other programs with a minimum of protocol mechanism. The protocol is transaction oriented, and delivery and duplicate protection are not guaranteed. UDP is commonly used in P2P and messenger programs to notify status and broadcast short messages.

SL Server 4 provides a UDP listener and by using UDPpoke'ed messages you could enable a P2P network of SL Server 4s sharing information with each other. With the UDPpoke function you can send messages (UDPDataPacket parameter) to a computer by specifying it's IP address (IpAddress

parameter) and the port number (PortNumber parameter) the specified computer has a active UDP listener on.

You can send messages to multiple computers in the same network or even in the same gateway segment by masking the IP sub-fields with the value 255. Masking all four sub-fields of an IP address would cause a broadcast going out over the full segment of the gateway but would never pass the gateway and therefore a WWW-wide broadcast is not possible without having a proxy to relay the broadcast to another gateway using a direct IP addressing method.

Parameters:

Name	Type	Description
IpAddress	string	IP address of remote target
PortNumber	string	UDP portnumber
UDPDataPacket	string	UDP data packet

Sub UnBlockSession

Description: UnBlock session

Use to unblock a previous made block on a session. See also BlockSession.

Parameters:

Name	Type	Description
SessionID	string	Session ID

Sub UnLockIP

Description: Unlock a IP address

Add or set a single IP address for passing the firewall. This function does not add a range or description to the firewall entry. Use the AddIP2 function to add ranges and/or descriptions.

Parameters:

Name	Type	Description
ClientIPaddress	string	IP address of a client

Sub UnlockUser

Description: Undo user lockout

Use this function to undo a user lockout. This function works only for logged on users.

Parameters:

Name	Type	Description
Username	string	Username of the locked user

Function UpdateSession

Description: Update a session (ttl and time)

Returns a nonzero number when a session is updated successfully. This function will keep (artificially) the session alive by preventing a TTL expiration.

Parameters:

Name	Type	Description
Index	numeric	Reference to a connection object

Sub WriteToLog

Description: Writes a single line of text to the server log

The logged text is listed bold in the log viewer having the tag "[Script]" in collumn "IP address". The text line can be found in collumn "Url / Mod. Msg".

Parameters:

Name	Type	Description
LogLine	string	Text to write to the log

Function ZCompress

Description: Compresses a string using the Zlib method

Parameters:

Name	Type	Description

UncompressedData	string	Uncompressed data

Function ZCompressFile

Description: Compress a file using the Zlib method

Parameters:

Name	Type	Description
DeCompressedFile	string	Filename of the decompressed file
CompressedFile	string	Filename of the compressed file

Function ZDeCompress

Description: Uncompresses a Zlib compressed string

Parameters:

Name	Type	Description
CompressedData	string	Compressed data

Function ZDeCompressFile

Description: Decompress a file using the Zlib method

Parameters:

Name	Type	Description
CompressedFile	string	Filename of the compressed file
DeCompressedFile	string	Filename of the decompressed file