Scripting

This section contains information on scripting.

Purpose

This function returns the path to the directory that the last script was run from.

Parameters

None.

Returns

Return value: path
Type: string

Example

hs.WriteLog "Script Path", hs.GetScriptPath

Writes this (example) to the log:

4/1/2004 2:00:00 PM~!~Script Path~!~C:\Program Files\HomeSeer\scripts\

Or for a script run from the scripts\Includes directory:

4/1/2004 2:00:00 PM~!~Script Path~!~C:\Program Files\HomeSeer\scripts\Includes

Purpose

This function indicates if a specified script is currently running.

Parameters

Parameter: script name
Type: string
Description: This is the name of the script to check.

Returns

Return value: status
Type: boolean
Description: This returns TRUE if the specified script is currently running and FALSE if it doesn't.

Example

```
' check if the script "weather.txt" is running

if hs.IsScriptRunning("weather.txt") then
  hs.writelog "info","The weather script is still running"
end if
```

Public Function RunScript(ByVal scr As String, ByVal Wait As Boolean, ByVal SingleInstance As Boolean) As Object

Purpose

This function runs another script. This will also return a value from the called script provided the "Main" procedure is a function.

Scripts must be located in the scripts directory in the HomeSeer application directory (C:\Program Files\HomeSeer 2\Scripts by default).

Parameters

Parameter: Script
Type: String
Description: This is the file name of the script to run. Do not include the path in the script name. The "Main" procedure in the script will be run. If you need to run a specific procedure other than Main, see RunScriptFunc .

Optional Parameter: **Wait**
Type: B**oolean**
Description: When set to TRUE, the script that is calling hs.RunScript will not continue processing commands until the script referenced here is finished. Set this to False to allow the script using hs.RunScript to continue processing commands after launching the additional script.

Optional Parameter: SingleInstance
Type: Boolean
Description: When set to TRUE, only one instance of the script referenced by hs.RunScript can be running at a time, so if there is one instance already running, calling this again will result in an empty/null return and an error message written to the log.

Returns

Return value: Value
Type: Object
Description: This returns any value that the called script returns from the Main function - Sub Main will not return any values.

```
Public Function RunScriptFunc(ByVal Script As String, ByVal Proc As String, _
                              ByVal Params As Object, ByVal Wait As Boolean, _
                              ByVal SingleInstance As Boolean) As Object
```

Purpose

This procedure runs another script and specifies a procedure to run in that script and optional parameters. This will also return a value from the called script.

Scripts must be located in the scripts directory in the HomeSeer application directory (C:\Program Files\HomeSeer 3\Scripts by default).

Parameters

Parameter: Script
Type: String
Description: This is the file name of the script to run. Do not include the path in the script name.

Parameter: Proc
Type: String
Description: This is the name of the procedure (Sub or Function) to execute. If this is left blank, the procedure "Main" will be run.

Parameter: Params
Type: Object
Description: This is a parameter or a set of parameters to send to the procedure. This can be a string or numeric value, or even an array of different values.

Optional Parameter: Wait
Type: Boolean
Description: When set to TRUE, the script that is calling hs.RunScriptFunc will not continue processing commands until the script referenced here is finished. Set this to False to allow the script using hs.RunScriptFunc to continue as soon as the other script is launched.

Optional Parameter: SingleInstance
Type: Boolean
Description: When set to TRUE, only one instance of the script referenced by hs.RunScriptFunc can be running at a time, so if there is one instance already running, calling this again will result in an empty/null return and an error message written to the log.

Returns

Return value: Value
Type: Object
Description: This returns any value (numeric, string, object) that the called script returns if the called procedure is a function.

Purpose

This returns a comma separated list of all of the scripts currently running in the system.

Parameters

None.

Returns

Return value: script list
Type: string
Description: This returns all of the currently running script names, separated by commas.

Purpose

This function will suspend operation of the script and allow the HomeSeer application to run. This is useful if you are waiting for a voice command or some other action that HomeSeer needs to recognize. If this function is not called, a script will time out in 30 seconds and prompt the user to either wait longer or kill the script. If this function is called within the 30 seconds, the script will not time out.

Parameters

None.

Returns

None.

Example

```
Sub Main(ByVal Parms As Object)


    Dim V As Double
    Do
       V = hs.DeviceValueEx(1234)
       If V = 41.66 Then Exit Do
       hs.WaitEvents()
       hs.WaitSecs(2)
    Loop
    hs.WriteLog("My Device", "The device has reached the proper value.")


End Sub
```

Purpose

This function waits a number of seconds. This will also allow other operations to take place in HomeSeer by giving up the CPU. It will also keep a script from timing out. The function will not return until the number of seconds have elapsed.

Parameters

Parameter: seconds
Type: integer
Description: This is the number of seconds to wait.

Returns

None.

Example

```
Sub Main(ByVal Parms As Object)

    Dim V As Double
    Do
        V = hs.DeviceValueEx(1234)
        If V = 41.66 Then Exit Do
        hs.WaitEvents()
        hs.WaitSecs(2)
    Loop
    hs.WriteLog("My Device", "The device has reached the proper value.")

End Sub
```

In This Section	
	Modifying Voice Recognition Commands Getting Last Voice Command Information Controlling Speaker Clients

In This Section	
	AddVoiceCommand ClearAllVoiceCommands

Purpose

This function will add the specified voice command to a new private command list. HomeSeer voice commands are disabled and the computer will only listen for the commands given using this function. When the script is exited, the computer will go back to listening for regular HomeSeer voice commands.

If the script is triggered by a voice command from HomeSeer Phone, make sure you add a system call to clear all voice commands. This will tell HomeSeer Phone to restore the main menu voice commands. The statement is:

system.ClearAllVoiceCommands

Parameters

Parameter: cmd
Type: string
Description: This is the voice command to add.

Parameter: host (optional)
Type: string
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

Return value: voice command
Type: string
Description: This is the specified voice command.

Example

The following script will read your E-mail messages.

```
Sub Main(ByVal Parms As Object)

    Dim Count As Integer
    Count = hs.MailMsgCount
    hs.Speak("You have " & Convert.ToString(Count) & " messages.", False, "")

    ' If no messages, exit.
    If Count < 1 Then Exit Sub



    hs.Speak("Would you like me to read your messages to you?", True, "")



    ' Clear out the last voice command recognized.
    hs.LastVoiceCommand = ""



    ' Create our own private recognition list.
    hs.AddVoiceCommand("Yes")
    hs.AddVoiceCommand("Sure")
    hs.AddVoiceCommand("Please")
    hs.AddVoiceCommand("No")



    Dim Resp As String = ""
    Dim GotResponse As Boolean = False
    Dim Start As Date = Now
    Do
       Resp = hs.LastVoiceCommand
       If Not String.IsNullOrEmpty(Resp.Trim) Then
          GotResponse = True
          Exit Do
       End If
       hs.WaitEvents()
    Loop Until Now.Subtract(Start).TotalSeconds > 15



    If Not GotResponse Then
       hs.ClearAllVoiceCommands()
       hs.Speak("Goodbye.", False, "")
       Exit Sub
    End If



    If Resp.Trim.ToLower = "no" Then
       hs.ClearAllVoiceCommands()
       hs.Speak("OK, perhaps later.", False, "")
       Exit Sub
    End If
```

```
For i As Integer = 0 To Count - 1
    hs.Speak("Message " & Convert.ToString(i), True, "")
    hs.Speak("Left on " & hs.MailDate(i), True, "")
    hs.Speak("The message is from,, " & hs.MailFrom(i), True, "")
    hs.Speak(" and the subject of the message is " & hs.MailSubject(i), True, "")
    hs.Speak(",, would you like me to read you the message?", True, "")

    Resp = ""
    GotResponse = False
    Start = Now
    Do
        Resp = hs.LastVoiceCommand
        If Not String.IsNullOrEmpty(Resp.Trim) Then
            GotResponse = True
            Exit Do
        End If
        hs.WaitEvents()
    Loop Until Now.Subtract(Start).TotalSeconds > 15



    If Not GotResponse Then
        hs.ClearAllVoiceCommands()
        hs.Speak("Goodbye.", False, "")
        Exit Sub
    End If



    Select Case Resp.Trim.ToLower
        Case "yes", "sure", "please"
            hs.Speak(hs.MailText(i), True, "")
            hs.WaitEvents()
    End Select



    hs.WaitSecs(2)



Next



hs.Speak("That was your last message.  Goodbye.", False, "")
hs.ClearAllVoiceCommands()



End Sub
```

Purpose

This function clears all voice commands that were added with AddVoiceCommand.

Parameters

Parameter: Host (optional)
Type: String
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

In This Section	
	GetLastVRCollection GetLastVRInfo LastCommandSelected LastVoiceCommand LastVoiceCommandHost LastVoiceCommandInstance LastVoiceCommandPhone LastVoiceCommandRaw

Purpose

This function gets the last voice command recognized by all speaker clients and HomeSeer Phone lines. The return is a simple array of clsLastVR objects.

Parameters

None.

Returns

Return value: LastVR
Type: Array of clsLastVR
Description: This returns the last voice command that HomeSeer recognized on all connected speaker clients and HomeSeer phone lines in an array of clsLastVR objects.

Note - if a given speaker client is connected but has not been used for VR since HomeSeer was started, it will not be a part of the returned array.

See Also:

clsLastVR
GetLastVRInfo
LastCommandSelected
LastVoiceCommand
LastVoiceCommandPhone
LastVoiceCommandHost
LastVoiceCommandInstance
LastVoiceCommandRaw

clsLastVR is an object class used with GetLastVRInfo and GetLastVRCollection and returns information about the last recognized voice command given to HomeSeer.

Here are the properties of the class:

Property	**Type**	**Description**
Raw	String	This is the raw voice command as it was heard and recognized by the VR engine.
Parsed	String	This is the parsed voice command. For HomeSeer generated voice commands, this string will contain special indicators for the matched device or event - it will not match the spoken text.
Host	String	This is the host name of the speaker client the recognized phrase was spoken to, or 'Phone' if it was spoken via HomeSeer Phone's local or remote interaction.

Instance	String	This is the instance name of the speaker client the recognized phrase was spoken to, or the phone line number if it was spoken via HomeSeer Phone's local or remote interaction.
VRTime	Date	This is the date/time the phrase was recognized.
ID	Integer	This is the voice recognition context ID number that was matched for the recognized phrase.

See Also:

GetLastVRInfo
GetLastVRCollection
LastCommandSelected
LastVoiceCommand
LastVoiceCommandPhone
LastVoiceCommandHost
LastVoiceCommandInstance
LastVoiceCommandRaw

Purpose

This function gets the last voice command recognized by a given speaker client/instance. The return is a clsLastVR object matching the speaker client host name and instance provided.

Parameters

Parameter: host
Type: string
Description: This is the host name for the speaker client to retrieve the last recognized VR information from.

Parameter: instance
Type: string
Description: This is the instance name for the speaker client to retrieve the last recognized VR information from.

Note - if a given speaker client is connected but has not been used for VR since HomeSeer was started, it will not be returned with this command.

Returns

Return value: LastVR
Type: clsLastVR
Description: This returns the last voice command that HomeSeer recognized on the given host:instance in a clsLastVR object, or 'nothing' if no matching host:instance was found.

See Also:

clsLastVR
GetLastVRCollection
LastCommandSelected
LastVoiceCommand
LastVoiceCommandPhone
LastVoiceCommandHost
LastVoiceCommandInstance
LastVoiceCommandRaw

clsLastVR is an object class used with GetLastVRInfoand GetLastVRCollectionand returns information about the last recognized voice command givento HomeSeer.

Here are the properties of the class:

Property	Type	Description
Raw	String	This is the raw voice command as it was heard and recognized by the VR engine.
Parsed	String	This is the parsed voice command. For HomeSeer generated voice commands, this string will contain special indicators for the matched device or event - it will not match the spoken text.
Host	String	This is the host name of the speaker client the recognized phrase was spoken to, or 'Phone' if it was spoken via HomeSeer Phone's local or remote interaction.
Instance	String	This is the instance name of the speaker client the recognized phrase was spoken to, or the phone line number if it was spoken via HomeSeer Phone's local or remote interaction.
VRTime	Date	This is the date/time the phrase was recognized.
ID	Integer	This is the voice recognition context ID number that was matched for the recognized phrase.

See Also:

GetLastVRInfo
GetLastVRCollection
LastCommandSelected
LastVoiceCommand
LastVoiceCommandPhone
LastVoiceCommandHost
LastVoiceCommandInstance
LastVoiceCommandRaw

Purpose

This function gets the event name of the last voice command. This works the same as the LastVoiceCommand function except it will return the actual name of the voice command. This is useful if you wanted to do some other action to the event, like delete it or disable it and you need that actual event name. This is a read-only property.

Parameters

None.

Returns

Return value: event name
Type: string
Description: This returns the name of the event that was triggered by the last voice command.

See Also:

clsLastVR
GetLastVRInfo
GetLastVRCollection
LastVoiceCommand
LastVoiceCommandPhone
LastVoiceCommandHost
LastVoiceCommandInstance
LastVoiceCommandRaw

Purpose

This function gets the last voice command recognized by a speaker client. This is a read-only property. This value is the parsed (processed) voice recognition string, which means that some parts of the command may be replaced by values or codes that HomeSeer uses to interpret what was spoken. See LastVoiceCommandRaw to get the unparsed (raw) phrase.

Parameters

None.

Returns

Return value: voice command
Type: string
Description: This returns the last voice command that HomeSeer recognized. This is useful for obtaining the actual voice command when the given voice command contains many optional words.

Example

If a voice command was set to:

tv channel (0|1|2|3|4|5|6|7|8|9)

and the user spoke "*tv channel 4*", this function would return the string "*tv channel 4*"

Create an event name tv channel. Set the voice command to:

tv channel (0|1|2|3|4|5|6|7|8|9)

Set the actions of the event to run the following script. When you speak a phrase like "tv channel 2", a message box will pop up giving you the actual command the system recognized.

```
sub main()

dim v
v=hs.LastVoiceCommand
msgbox "I heard "&v

end sub
```

See Also:

clsLastVR
GetLastVRInfo
GetLastVRCollection
LastCommandSelected
LastVoiceCommandPhone
LastVoiceCommandHost
LastVoiceCommandInstance
LastVoiceCommandRaw

Purpose

This function gets the host name of the speaker client for the last voice command recognized by a speaker client. This is a read-only property. This command will return "Phone" if the last recognized command came from the a HomeSeer Phone line.

Parameters

None.

Returns

Return value: host name
Type: string
Description: This returns the host name where the speaker client is running that the last voice command that HomeSeer recognized originated from. If the phone interface was used, this command returns the text: Phone

See Also:

clsLastVR
GetLastVRInfo
GetLastVRCollection
LastCommandSelected
LastVoiceCommand
LastVoiceCommandPhone
LastVoiceCommandInstance
LastVoiceCommandRaw

Purpose

This function gets the instance name of the speaker client for the last voice command recognized by a speaker client. This is a read-only property. Thi s command will return a phone line number (e.g. "1", "2", etc.) if the last recognized command came from a HomeSeer Phone line.

Parameters

None.

Returns

Return value: instance name
Type: string
Description: This returns the instance name where the speaker client is running that the last voice command that HomeSeer recognized originated from. If the phone interface was used, this command returns the HomeSeer Phone line number as text.

See Also:

clsLastVR
GetLastVRInfo
GetLastVRCollection
LastCommandSelected
LastVoiceCommand
LastVoiceCommandPhone
LastVoiceCommandHost
LastVoiceCommandRaw

Purpose

This function gets the last voice command recognized by HomeSeer Phone. This is a read-only property. This value is the parsed (processed) voice recognition string, which means that some parts of the command may be replaced by values or codes that HomeSeer uses to interpret what was spoken. See LastVoiceCommandRaw to get the unparsed (raw) phrase.

HomeSeer Phone is required to do voice recognition over the telephone.

Parameters

None.

Returns

Return value: voice command
Type: string
Description: This returns the last voice command that HomeSeer recognized via the telephone. This is useful for obtaining the actual voice command when the given voice command contains many optional words.

Example

If a voice command was set to:

tv channel (0|1|2|3|4|5|6|7|8|9)

and the user spoke "*tv channel 4*", this function would return the string "*tv channel 4*"

Create an event name tv channel. Set the voice command to:

tv channel (0|1|2|3|4|5|6|7|8|9)

Set the actions of the event to run the following script. When you speak a phrase like "tv channel 2", a message box will pop up giving you the actual command the system recognized.

```
sub main()

dim v
v=hs.LastVoiceCommandPhone
hs.WriteLog "LVCP", "I heard " & v & " from the phone."

end sub
```

See Also:

clsLastVR
GetLastVRInfo
GetLastVRCollection
LastCommandSelected
LastVoiceCommand
LastVoiceCommandPhone
LastVoiceCommandHost
LastVoiceCommandInstance
LastVoiceCommandRaw

Purpose

This function gets the last voice command recognized by a speaker client or HomeSeer phone in raw (unparsed) format. The unparsed format matches the phrase spoken by the user. This is a read-only property.

Parameters

None.

Returns

Return value: voice command
Type: string
Description: This returns the last voice command that HomeSeer recognized in unparsed form. In unparsed form, the spoken phrase "Turn on the Kitchen Light" will return the same text. In parsed form, the phrase might return something like "Turn on DV:5427"

See Also:

clsLastVR
GetLastVRInfo
GetLastVRCollection
LastCommandSelected
LastVoiceCommand
LastVoiceCommandPhone
LastVoiceCommandHost
LastVoiceCommandInstance

In This Section	
	GetListenStatus ListenMode ListenForCommands SetSpeaker StartListen StopListen

Purpose

This function returns the listening status of a specific speaker client (host or host:instance).

Parameters

Parameter: host
Type: string
Description: Leaving this a null string will return the status for the first instance HomeSeer finds, otherwise use the hostname of the computer you are interested in determining the listening status of. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

Return value: listening status
Type: boolean
Description: TRUE indicates that the speaker app instance is listening.

Purpose

This function indicates the current listening mode.

Parameters

None.

Returns

Return value: mode
Type: integer
Description: This returns the current listen mode which is define as:

1 = Not Listening
2 = Listening for commands
3 = Listening for attention

Purpose

This function will switch the computer from either listening for event name commands or listening for the attention phrase.

Parameters

Parameter: action
Type: boolean
Description: Use TRUE to listen for event name commands and FALSE to listen for the attention phrase.

Returns

None.

Example

```
sub main()

    ' listen only for attention phrase
 hs.ListenForCommands FALSE

end sub
```

Purpose

This procedure changes the speaker profile on one or more Speaker clients to the profile name provided.

Parameters

Parameter: speaker_name
Type: string
Description: This is the name of the speaker profile to switch to. The speaker profile name must match one of the available speaker profiles on the computer that the HomeSeer Speaker client is running on.

Parameter: host (optional)
Type: string
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

Purpose

This function starts the voice recognition engine if it is not already started. For scripts that are to be used over the phone, use the *System* functions.

Parameters

Parameter: host (optional)
Type: string
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

Purpose

This function stops the voice recognition engine if it is not already stopped. For scripts that are to be used over the phone, use the *System* functions.

Parameters

Parameter: host (optional)
Type: string
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

In This Section	
	Global Variables Encryption Counters Timers

In This Section	
	CreateVar DeleteVar GetVar SaveVar

Purpose

This function creates a new global variable. The variable may be accessed by the functions SaveVar and GetVar. The variable is global in scope and can only be destroyed with the DeleteVar function or exiting the application.

The variable created is an object and can be used to hold any variable type, including references to objects.

Parameters

Parameter: name
Type: string
Description: This is the name of the variable.

Returns

Return value: error code
Type: string
Description: This is an empty string if there's no error. Otherwise, an error string will be returned if the variable already exists.

Example

```
dim errst

errst = CreateVar("myvar")

if errst <> "" then
msgbox "Error creating variable"
end if
```

Purpose

This function deletes a global variable or reference to an object that was created by CreateVar. If the variable does not exist, the function does nothing.

Parameters

Parameter: name
Type: string
Description: This is the name of the variable.

Returns

None.

Purpose

This function finds the variable associated with the name parameter and returns it.

Parameters

Parameter: name
Type: string
Description: This is the name of the variable.

Returns

Return value: variable item
Type: variant
Description: This returns the variable saved.

Example

```
dim myvar

myvar = hs.GetVar("myvar")

' if "myvar" is an object, then get the variable with:

set myvar = hs.GetVar("myvar")
```

Purpose

This function saves the variable contained in the obj parameter. The parameter may be any variable type such as a string or integer, or it may be a reference to an object created with CreateObject.

Parameters

Parameter: name
Type: string
Description: This is the name of the variable.

Parameter: obj
Type: object
Description: This is the object to be saved.

Returns

Return value: error code
Type: string
Description: This returns an empty string if no error occurred and returns an error string if one did occur.

Example

```
dim errst
dim myvalue

myvalue = 10
errst = hs.SaveVar("myvar",myvalue)
```

In This Section	
	EncryptString EncryptStringEx DecryptString

Purpose

This function encrypts a string using an encryption password that you specify. Although many unprintable characters can be written to a text file successfully, Windows terminates a text string with a carriage return/line-feed character combination. The bRecurse parameter is provided to cause the function to recursively encrypt the data until it detects no carriage return or line-feed characters in it. The string may then be written to a text file such as when you save it in an INI file using SaveINISetting. Using bRecurse on a large amount of text is NOT recommended as it may recursively encrypt for a long time in an attempt to remove carriage return and line-feed characters, or the function may result in an error due to too many attempts to recursively encrypt. Another solution for writing encrypted data to a text file safely is to convert it to a text representation of HEX data. See the example below.

Parameters

Parameter: sToEncrypt
Type: string
Description: This is the text that you want encrypted.

Parameter: sPassword
Type: string
Description: This is the user-created text string to encrypt the text with.

Returns

Return value: data
Type: string
Description: This returns a string containing an encrypted form of sToEncrypt, encrypted using sPassword. This string is not limited to printable characters only, so care should be taken in the storage of this data in files.

Example

```
Sub Main(ByVal Parms As Object)

    If Parms Is Nothing Then Exit Sub

    ' Encrypt the combination to my vault full of money.  The combination
    '   that I just entered is stored in the variable sCombEntered.
    Dim sCombEntered As String = Convert.ToString(Parms)

    Dim sCombination As String = ""
    sCombination = hs.EncryptString(sCombEntered, "For Spouse Only Spend Wisely", False)

    ' I now have my encrypted combination in sCombination.  I must remember
    '   to use HomeSeer's or Microsoft's script encrypters on this script
    '   since my password string is plainly visible above!

    ' I want to store the combination in a text file, so let's Base64 encode it.
    Dim bteArray() As Byte
    bteArray = Encoding.ASCII.GetBytes(sCombination)
    Dim sOutput As String = ""
    sOutput = Convert.ToBase64String(bteArray, Base64FormattingOptions.None)

    ' Now I have sOutput as a text representation of bytes, I can write that to an INI
    '   file and reverse the process of encoding it to Base64 to unencode it.
    hs.SaveINISetting("Passwords", "Vault", sOutput, "MyPasswords.ini")

End Sub
```

See Also:

EncryptStringEx
DecryptString

Purpose

This function encrypts a string using an encryption password that you specify.

Notes

Encrypted strings using this function are encrypted using strong (AES/Rijndael) encryption - the data can NOT be recovered if the password(s) are lost. The resulting data may have unprintable characters, so you may not be able to save it using INI functions. Another solution for writing encrypted data to a text file safely is to convert it to a text representation of HEX data. See the example used in EncryptString.

Parameters

Parameter: Text
Type: String
Description: This is the text that you want encrypted.

Parameter: Password
Type: String
Description: This is the user-created text string to encrypt the text with.

Parameter: KeyModifier
Type: String
Description: This parameter may be used to provide further user-specific encryption of the data - it is a modifier used with the password parameter to create the encryption key.

Returns

Return value: data
Type: string
Description: This returns a string containing an encrypted form of Text, encrypted using Password (and KeyModifier if provided). This string is not limited to printable characters only, so care should be taken in the storage of this data in files.

See Also:

EncryptString
DecryptString

Purpose

This function decrypts a string using a decryption password that you specify.

Parameters

Parameter: sToDecrypt
Type: String
Description: This is the text that you want decrypted (unencrypted).

Parameter: sPassword
Type: String
Description: This is the user-created text string to encrypt the text with.

Parameter: KeyModifier (Optional)
Type: String
Description: This optional parameter is the modifier text to use with the password to create the key - if EncryptStringEx was used to encrypt the string and a key modifier was used, you must specify the same key modifier here.

Returns

Return value: Data
Type: String
Description: This returns a string containing a decrypted form of sToDecrypt, decrypted using sPassword. Only the same value of sPassword used to encrypt the string will return the original string in this function.

Example

```
Sub Main(ByVal Parms As Object)


    ' Decrypt the combination to my vault full of money.
```

```
    ' First I have to read the encrypted string from a file and unencode it.
    ' The string was Base64 encoded so that it could be safely written to a text file.
    Dim sCombination As String = ""
    sCombination = hs.GetINISetting("Passwords", "Vault", "NOTHING", "MyPasswords.ini")
    If sCombination Is Nothing OrElse String.IsNullOrEmpty(sCombination.Trim) Then
        hs.WriteLog("Error", "Encrypted combination was not read from the INI file properly.")
        Exit Sub
    End If
    If sCombination.Trim.ToLower = "nothing" Then
        hs.WriteLog("Warning", "The encrypted vault password was not found in our passwords INI file.")
        Exit Sub
    End If



    ' Now decode the string back into an array of bytes.
    Dim bteArray() As Byte
    bteArray = Convert.FromBase64String(sCombination)
    Dim sCombEntered As String = ""
    sCombEntered = Encoding.ASCII.GetString(bteArray)



    ' Now we have the encrypted combination, so let's decrypt it. (We'll re-use sCombination)
    sCombination = hs.DecryptString(sCombEntered, "For Spouse Only Spend Wisely")



    ' I now have my decrypted combination in sCombination.  I must remember
    '   to use HomeSeer's or Microsoft's script encrypters on this script
    '   since my password string is plainly visible above!


End Sub
```

See Also:

EncryptString
EncryptStringEx

Counters are created from within HS3 either from the Global Timers Counters page or from within an event. The functions in this section allow scripts or plugins to manipulate the counters.

Purpose

Return the value of a counter.

Parameters

Parameter: CounterName
Type: **String**
Description: The name of the counter.

Returns

Return value: **Result**
Type: Double
Description: Returns the counter value.

Example

```
dim value as Double = hs.CounterValue("mycounter")
```

Purpose

Resets a counter to 0. If any events are waiting on the change of the counter, they may trigger when this sub is called.

Parameters

Parameter: **CounterName**
Type: **String**
Description: The name of the counter to reset.

Returns

Return value: **Nothing**
Description: This is a Sub and does not return a value.

Example

```
hs.CounterReset("mycounter")
```

Purpose

Increments a counter. If any events are waiting on a change to the counter, they may trigger.

Parameters

Parameter: **CounterName**
Type: **String**
Description: The name of the counter to increment.

Returns

Return value: **Nothing**
Description: This is a Sub and does not return a value;

Example

```
hs.CounterIncrement("mycounter")
```

Purpose

Decrementsa counter. If any events are waiting on a change to the counter, they may trigger.

Parameters

Parameter: **CounterName**
Type: **String**
Description: The name of the counter to decrement.

Returns

Return value: **Nothing**
Description: This is a Sub and does not return a value;

Example

hs.CounterDecrement("mycounter")

Timers are created from the Global Timers Counters page or from within an event. The following functions can be used from within a script or plugin to manipulate the timers.

Purpose

Retrieve the value of a specific named timer.

Parameters

Parameter: TimerName
Type: **String**
Description: The name of the timer to retrieve the value from.

Returns

Return value: **Timer Value**
Type: **TimeSpan**
Description: A TimeSpan object that represents the timer.

Example

dim ts as TimeSpan = hs.TimerValue("mytimer")

Purpose

Reset a timer to 0.

Parameters

Parameter: **TimerName**
Type: **String**
Description: The name of the timer to reset.

Returns

Return value: **Nothing**
Description: This is a Sub and does not return a value.

Example

hs.TimerReset("mytimer")

In This Section	
	Time Related Calendar Related

In This Section	
	LocalTimeZone SolarNoon Sunrise SunriseDt Sunset SunsetDt TimeZoneName

Purpose

This function returns an offset in minutes from UTC (Universal Time Coordinate) for your time zone.

The offset is based upon UTC, which is the time standard used since 1972, and not GMT, which was the previous standard.

Parameters

None.

Returns

Return value: Offset
Type: Integer
Description: This returns the current time zone offset from UTC for the time zone set on your HomeSeer computer.

Example

hs.WriteLog "TimeZone","My timezone offset here in Eastern Daylight Time from UTC is " & CStr(hs.LocalTimeZone / 60) & " hours."

Results in this being written to the log:

4/14/2004 3:00:00 PM~!~TimeZone~!~My timezone offset here in Eastern Daylight Time from UTC is 5 hours.

Purpose

This function returns the time of solar noon. This is a read-only property.

Parameters

None.

Returns

Return value: solar noon time
Type: date
Description: This is a date item representing the time of solar noon, the period at which the sun appears directly overhead a location.

Example

```
sub main()

dim t

t=hs.SolarNoon
msgbox "Solar Noon is at " & FormatDateTime(t, vbLongTime)

end sub
```

Purpose

This function returns the time of sunrise. This is a read-only property.

Parameters

None.

Returns

Return value: sunrise time
Type: string
Description: This is a string representing the time of sunrise. The string returned is formatted according to your system's setting for time display but with seconds removed (e.g., if there are three fields separated by colons, the third one is removed).

Example

```
sub main()

dim t

t=hs.Sunrise
msgbox "Sunrise is at " & t

end sub
```

Purpose

This function returns the time of sunrise. This is a read-only property.

Parameters

None.

Returns

Return value: sunrise time
Type: date
Description: This is a date type representing the time of sunrise.

Example

```
sub main()

dim t

t=hs.SunriseDt
msgbox "Sunrise is at " & FormatDateTime(t, vbLongTime)

end sub
```

Purpose

This function returns the time of sunset. This is a read-only property.

Parameters

None.

Returns

Return value: sunset time
Type: string
Description: This is a string representing the time of sunset. The string returned is formatted according to your system's setting for time display but with seconds removed (e.g., if there are three fields separated by colons, the third one is removed).

Example

```
sub main()

dim t

t=hs.Sunset
msgbox "Sunset is at " & t

end sub
```

Purpose

This function returns the time of sunset. This is a read-only property.

Parameters

None.

Returns

Return value: sunset time
Type: date
Description: This is a date type representing the time of sunset.

Example

```
sub main()

dim t

t=hs.SunsetDt
msgbox "Sunset is at " & FormatDateTime(t, vbLongTime)

end sub
```

Purpose

This function returns the name of the PC's time zone setting. This is a read-only property.

Parameters

None.

Returns

Return value: time zone
Type: string
Description: This is the name of the time zone as read from the operating system.

Example

```
sub main()

dim t

t=hs.TimeZoneName
msgbox "The TimeZone is " & t

end sub
```

In This Section	
	DaylightSavings DaysInMonth DaysLeftInMonth DaysLeftInYear EvenOddMonth EvenOddDay GetLastWeekday GetSpecialDay IsSpecialDay IsWeekday IsWeekend Moon Weekdays WeekEndDays WeekNumber WeekNumberEx WeeksLeftInYear WeeksLeftInYearEx

Purpose

This function returns whether daylight savings is currently active. This is a read-only property.

Parameters

None.

Returns

Return value: Currently in daylight savings
Type: Boolean
Description: The return value (TRUE or FALSE) indicates whether the current date falls under daylight savings time as reported by the operating system.

Daylight savings is not used in all locations.

Example

```
sub main()

if hs.DayLightSavings then
 hs.WriteLog "We are currently in daylight savings!"
end if

end sub
```

Purpose

This function returns the number of days in the month of a date value supplied to it.

Parameters

Parameter: Date
Type: Date
Description: This is a date object for which you wish to know how many days are in that month. The day of the month in the date object is ignored.

Returns

Return value: number of days
Type: Integer

Example

```
Dim dte As Date = DateTime.Parse("April 1, 2006")
hs.WriteLog("Info","There are " & hs.DaysInMonth(dte).ToString & " days in the month of April, 2006")
```

See Also

DaysLeftInMonth
DaysLeftInYear
WeekNumber
WeeksLeftInYear

Purpose

This function returns a value indicating how many days are left in the current month.

Parameters

None.

Returns

Return value: Number of days
Type: Integer
Description: The number of days remaining in the current month.

Example

```
hs.WriteLog("Info","There are " & hs.DaysLeftInMonth.ToString & " days left in the month.")
```

See Also

DaysInMonth
DaysLeftInYear
WeekNumber
WeeksLeftInYear

Purpose

This function returns a value indicating how many days are left in the current year.

Parameters

None.

Returns

Return value: Number of days
Type: Integer
Description: The number of days remaining in the current year.

Example

```
hs.WriteLog("Info","There are " & hs.DaysLeftInYear.ToString & " days left in the year.")
```

See Also

DaysInMonth
DaysLeftInMonth
WeekNumber
WeeksLeftInYear

Purpose

This function returns a value indicating whether the provided day of the month is even or odd.

Parameters

Parameter: date
Type: date
Description: This is the date that you wish to check for being even or odd for the month.

Returns

Return value: CD_DAY_EvenOdd
Type: Enum (Integer) 0 = Even, 1 = Odd
Description: The return is a .NET Enum equivalent to an integer value.
The return value converted to string with the .ToString method will return the word Even or Odd,
but when converted to an integer value and then to a string it will display 0 or 1.

Example

```
Dim dte As Date = DateTime.Parse("April 1, 2006")
hs.WriteLog("Info","April 1 of 2006 is an " & hs.EvenOddMonth(dte).ToString & " day of the month.")
```

See Also

EvenOddDay

Purpose

This function returns a value indicating whether the provided day of the year is even or odd.
(The day of the month may be odd, but it may still be an even number for the year.)

Parameters

Parameter: date
Type: date
Description: This is the date that you wish to check for being even or odd for the year.

Returns

Return value: CD_DAY_EvenOdd
Type: Enum (Integer) 0 = Even, 1 = Odd
Description: The return is a .NET Enum equivalent to an integer value.
The return value converted to string with the .ToString method will return the word Even or Odd,
but when converted to an integer value and then to a string it will display 0 or 1.

Example

```
Dim dte As Date = DateTime.Parse("April 1, 2006")
hs.WriteLog("Info","April 1 of 2006 is an " & hs.EvenOddDay(dte).ToString & " day.")
```

See Also

EvenOddMonth

Purpose

This function returns a date representing the last weekday of the month from the date provided.

Parameters

Parameter: date
Type: date
Description: This is a date in the month for which you wish to know the date of the last weekday of that month.

Returns

Return value: last weekday date
Type: date
Description: The date of the last weekday of the month.

Example

```
Dim dte As Date = DateTime.Parse("April 1, 2006")
hs.WriteLog("Info","The last weekday of April 2006 is " & hs.GetLastWeekday(dte).ToShortDateString)
```

See Also

IsWeekday
IsWeekend
Weekdays
WeekendDays

Purpose

This function returns a date object representing the requested special date. e.g. The Third Thursday of November.

Parameters

Parameter: DOW
Type: DayOfWeek (Enum - Integer)
Description: This is the day of the week value you are looking for. The values for the Enum are as follows:
- SUNDAY = 0
- MONDAY = 1
- TUESDAY = 2
- WEDNESDAY = 3
- THURSDAY = 4
- FRIDAY = 5
- SATURDAY = 6
- WEEKDAY = 7
- WEEKEND_DAY = 8

Parameter: Instance
Type: CD_DAY_IS_Type (Enum - Integer)
Description: This is the instance day that you wish to retrieve, using these values:
- FIRST = 1
- SECOND = 2
- THIRD = 3
- FOURTH = 4
- LAST = 5

Parameter: For Month
Type: date
Description: This date object references the month you are requesting the special date for - the day component of the month is not used. For example, to get the third Thursday in November of 2006, provide a date object set to any day/time in the month of November, 2006.

Optional Parameter: GetNext
Type: Boolean (Default value if not provided = False)
Description: If the requested special date has already passed and GetNext is True, then the next instance of the requested special day will be returned. (See the example below)

Returns

Return value: date requested
Type: date
Description: This is a date object with the month, day, year components for the special day requested.

Example

```
Sub Main(parm as object)

  Dim DOW as Integer = 3    ' Wednesday
  Dim Inst as Integer = 3      ' Third instance (e.g. Third Wednesday of the month)
  Dim ForMonth As Date = DateTime.Parse("February 3, 2006")
  Dim dteReturn As Date

  dteReturn = hs.GetSpecialDay(DOW, Inst, ForMonth, False)
  hs.WriteLog("Test","With GetNext False, Result is " & dteReturn.ToShortDateString)

  dteReturn = hs.GetSpecialDay(DOW, Inst, ForMonth, True)
  hs.WriteLog("Test","With GetNext True, Result is " & dteReturn.ToShortDateString)

End Sub
```

The example above returns:

```
 ~!~Test~!~With GetNext False, Result is 2/15/2006
~!~Test~!~With GetNext True, Result is 3/15/2006
```

See Also

IsSpecialDay

Purpose

This function returns a Boolean (True/False) indicating if a date provided is the special day indicated.

Parameters

Parameter: In Date
Type: date
Description: This date object references the date you wish to check.

Parameter: DOW
Type: DayOfWeek (Enum - Integer)
Description: This is the day of the week value you are looking for. The values for the Enum are as follows:
SUNDAY = 0
MONDAY = 1
TUESDAY = 2
WEDNESDAY = 3
THURSDAY = 4
FRIDAY = 5
SATURDAY = 6
WEEKDAY = 7
WEEKEND_DAY = 8

Parameter: Instance
Type: CD_DAY_IS_Type (Enum - Integer)
Description: This is the instance day that you wish to compare, using these values:
FIRST = 1
SECOND = 2
THIRD = 3
FOURTH = 4
LAST = 5

Parameter: For Month
Type: date
Description: This date object references the month you are requesting the special date for - the day component of the month is not used. For example, to get the third Thursday in November of 2006, provide a date object set to any day/time in the month of November, 2006.

Returns

Return value: Is Special
Type: Boolean (True/False)
Description: If In Date matches the special day information indicated with the other three parameters, then Is Special will be True, otherwise False.

Example

```
Sub Main(parm as object)

 Dim DOW as Integer = 3    ' Wednesday
 Dim Inst as Integer = 3      ' Third instance (e.g. Third Wednesday of the month)
 Dim ForMonth As Date = DateTime.Parse("February 3, 2006")
 Dim InDate As Date = DateTime.Parse("February 15, 2006")
 Dim bReturn As Boolean

 bReturn = hs.IsSpecialDay(InDate, DOW, Inst, ForMonth)
 If bReturn = True Then
    hs.WriteLog("Test","February 15 is the third Wednesday of February, 2006")
 Else
    hs.WriteLog("Test","February 15 is the third Wednesday of February, 2006")
 End If

End Sub
```

See Also

GetSpecialDay

Purpose

This function returns a Boolean (True/False) value indicating whether the date provided is a weekday or not.

Parameters

Parameter: date
Type: date
Description: This is the date that you wish to check.

Returns

Return value: Weekday
Type: Boolean
Description: If the date provided falls upon a weekday (Monday through Friday), then this return will be True, otherwise False.

See Also

GetLastWeekday
IsWeekend
Weekdays
WeekendDays

Purpose

This function returns a Boolean (True/False) value indicating whether the date provided is a weekend day or not.

Parameters

Parameter: date
Type: date
Description: This is the date that you wish to check.

Returns

Return value: Weekend
Type: Boolean
Description: If the date provided falls upon a weekend day (Saturday or Sunday), then this return will be True, otherwise False.

See Also

GetLastWeekday
IsWeekday
Weekdays
WeekendDays

Purpose

This subroutine accepts an input date, and returns into the variables you provide, the values of various moon phase data points including the dates of the new and full moon, the current cycle value, and the moon phase description.

Parameters

Parameter: dateStart
Type: Date
Description: This is the date for which you wish to retrieve moon phase information.

Parameter: NewMoon
Type: Date
Description: After the sub-routine completes, this variable will contain the date of the new moon relative to the provided starting date.

Parameter: FullMoon
Type: Date
Description: After the sub-routine completes, this variable will contain the date of the full moon relative to the provided starting date.

Parameter: Cycle
Type: Integer
Description: After the sub-routine completes, this variable will contain the value of the moon cycle on the date provided by the starting date.

Parameter: Description
Type: String
Description: This is the name of the current moon cycle.

Returns

None

Example

```
Sub Main(parm as object)

   Dim dtStart as Date = Now
  Dim NMoon as Date
  Dim FMoon as Date
  Dim CurCycle as Integer
  Dim sDesc as String = ""

      hs.Moon(dtStart, NMoon, FMoon, CurCycle, sDesc)
     hs.WriteLog("Moon","New on " & NMoon.ToShortDateString & ", Full on " & FMoon.ToShortDateString & _
                  ", Cycle is " & CurCycle.ToString & " = " & sDesc)

End Sub
```

The above example returns:

Moon - New on 8/24/2006, Full on 9/7/2006, Cycle is 24 = Waning Crescent

Purpose

This function returns the number of weekdays between two dates, inclusive of the end date.

Parameters

Parameter: Date Start
Type: date
Description: This is the starting date.

Parameter: Date End
Type: date
Description: This is the ending date.

Returns

Return value: Weekdays
Type: integer
Description: This is the number of weekdays between the two dates including the ending date if it is a weekday.

Example

```
Sub Main(parm as object)

   Dim dtStart as Date = DateTime.Parse("8/1/2006")
  Dim dtEnd as Date = DateTime.Parse("8/10/2006")
  Dim iResult as Integer

  iResult = hs.Weekdays(dtStart, dtEnd)
  hs.WriteLog("Weekdays","There are " & iResult.ToString & " weekdays between the dates.")

End Sub
```

The above example returns this result:

Weekdays - There are 7 weekdays between the dates.

See Also

GetLastWeekday
IsWeekday
IsWeekend
WeekendDays

Purpose

This function returns the number of weekend days between two dates, inclusive of the end date.

Parameters

Parameter: Date Start
Type: date
Description: This is the starting date.

Parameter: Date End
Type: date
Description: This is the ending date.

Returns

Return value: Weekdays
Type: integer
Description: This is the number of weekend days between the two dates including the ending date if it is a Saturday or Sunday.

Example

```
Sub Main(parm as object)

  Dim dtStart as Date = DateTime.Parse("8/1/2006")
 Dim dtEnd as Date = DateTime.Parse("8/13/2006")
 Dim iResult as Integer

 iResult = hs.WeekEndDays(dtStart, dtEnd)
 hs.WriteLog("WeekEndDays","There are " & iResult.ToString & " weekend days between the dates.")

End Sub
```

The above example returns this result:

WeekEndDays - There are 4 weekend days between the dates.

See Also

GetLastWeekday
IsWeekday
IsWeekend
Weekdays

Purpose

This function returns the week number of the year for the given date, and assumes the first full week starting on Sunday of the year as Week 1. For other options on the first week of the year, use WeekNumberEx.

Parameters

Parameter: In Date
Type: date
Description: This is the date for which you wish to know the week number.

Returns

Return value: WeekNumber
Type: short integer
Description: This is the week number of the year for the given date.

See Also

DaysInMonth
DaysLeftInMonth
DaysLeftInYear
WeeksLeftInYear
WeeksLeftInYearEx
WeekNumberEx

Purpose

This function returns the week number of the year for the given date, just like WeekNumber, except that you can specify the conditions for determining the first week of the year.

Parameters

Parameter: In Date
Type: date
Description: This is the date for which you wish to know the week number.

Parameter: Week Mode
Type: Integer
Description: This specifies how the first week of the year is determined, according to the following table:

Week Mode Value	Result
1	The first week of the year starts with the first calendar day of the year, regardless of the day of the week it falls upon.
4	The first week of the year is determined by the first week with at least four days in the new year.
(Anything Else)	The first week of the year is determined by the first full week starting on Sunday in the new year.

Returns

Return value: WeekNumber
Type: Integer
Description: This is the week number of the year for the given date and Week Mode.

See Also

DaysInMonth
DaysLeftInMonth
DaysLeftInYear
WeeksLeftInYear
WeeksLeftInYearEx
WeekNumber

Purpose

This function returns the number of weeks left in the current year based upon the first week of the year being the first full week starting on a Sunday. For other starting week options, see WeeksLeftInYearEx.

Parameters

None.

Returns

Return value: Weeks Left
Type: Integer
Description: This number represents the number of weeks remaining in the current year based upon the first week being the first full week starting on Sunday of the year.

See Also

DaysInMonth
DaysLeftInMonth
DaysLeftInYear
WeekNumber
WeekNumberEx
WeeksLeftInYearEx

Purpose

This function returns the number of weeks left in the current year, based upon the starting week mode provided as a parameter.

Parameters

Parameter: Week Mode
Type: integer (Optional)
Description: This specifies how the first week of the year is determined, according to the following table:

Week Mode Value	Result
1	The first week of the year starts with the first calendar day of the year, regardless of the day of the week it falls upon.
4	The first week of the year is determined by the first week with at least four days in the new year.
(Anything Else)	The first week of the year is determined by the first full week starting on Sunday in the new year.

Returns

Return value: Weeks Left
Type: short
Description: This number represents the number of weeks remaining in the current year as determined by the week mode parameter.

See Also

DaysInMonth
DaysLeftInMonth
DaysLeftInYear
WeekNumber
WeekNumberEx
WeeksLeftInYear

Text to Speech (TTS) and media are handled independently by the speaker clients. It is possible to have a media file playing and at the same time, have TTS being generated. If the computer that the speaker client is installed on only has one sound output, then both sounds are heard at the same time, mixed together. If the system has more than one sound output/resource, and Windows Media Player is set to use a different one than the default for audio, then it is possible to have the output from TTS and Media functions go their separate ways.

The TTS channel can also play WAV media files through PlayWavFile or PlayWavFileVol, so it is possible to have WAV audio play on the TTS channel in the event that the TTS channel and MEDIA channel are routed out separate sound devices.

This section covers the script commands for generating TTS, playing/controling media files, and controlling speaker clients.

HSTouch clients, as a speaker client, are more limited in their scope - they can only process TTS and will ignore the media related commands in this section.

In This Section	
	GetInstanceList IsSpeakerBusy SpeakToFile Speaker Client Global Audio Media Only Procedures Text-to-Speech Only Procedures

Purpose

This function retrieves a comma separated list of host:instance names for Speaker client instances currently connected to HomeSeer.

Parameters

None.

Returns

Return value: instance list
Type: string
Description: The returned instance list is a comma separated list of host:instance pairs as in this example:

Bandit:Default,Johnny:Default,Race:Music,Race:Default

Purpose

This function can let you know if a specific speaker client (host or host:instance) is currently busy speaking or playing WAV audio.

Parameters

Parameter: Host (Optional)
Type: String
Description: Leaving this a null string will return the busy status for the first instance HomeSeer finds, otherwise use the hostname of the computer you are interested in determining the busy status of. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

Return value: busy status
Type: boolean
Description: TRUE indicates that the speaker application instance is busy.

Purpose

This function speaks some text and saves the result in a WAV file.

Parameters

Parameter: Text
Type: String
Description: This is the string you want to speak.

Parameter: Voice
Type: String
Description: This is the name of the voice you want to use for speaking. This string must match the voice name exactly. Voice names can be found in the Speaker Client. If the name is omitted, the default voice as specified in the computer's speech control panel is used.

Parameter: Filename
Type: String
Description: This is the full path to the file where the voice output will be saved.

Returns

None.

Example

```
sub main()

   hs.SpeakToFile "Hello from a file!", "ATT DTNV 1.3 Crystal","c:\voice.wav"

end sub
```

See Also

Using Replacement Variables

In This Section	
	SetVolume GetVolume GetMuteStatus GetPauseStatus MuteAudio PauseAudio UnMuteAudio UnPauseAudio

Purpose

This function sets the master volume of the system sound device that the speaker client(s) are using. This can be used to set the volume of the text-to-speech output. The volume level must be in a range between 0 and 100, where 100 is the maximum volume.

To change the volume for the MEDIA functions, use MediaVolume.

Parameters

Parameter: Level
Type: Integer
Description: This is the volume level, from 0 to 100.

Parameter: Host (Optional)
Type: String
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

Example

```
sub main()


hs.SetVolume 90

hs.speak "I am speaking louder",TRUE

hs.SetVoume 20, "Kitchen"

hs.speak "I am speaking softer on the Kitchen computer than on the others.",TRUE


end sub
```

Purpose

This function returns the volume level of an instance of the Speaker client program running on a computer.

Parameters

Parameter: Host (Optional)
Type: String
Description: Leaving this a null string will return the volume level for the first instance HomeSeer finds, otherwise use the hostname of the computer you are interested in determining the volume level of. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

Return value: Volume level
Type: Integer
Description: The volume level is returned using a 0-100 scale, 100 being full volume.

Purpose

This function returns the mute status of a specific speaker client (host or host:instance).

Parameters

Parameter: Host (Optional)
Type: String
Description: Leaving this a null string will return the status for the first instance HomeSeer finds, otherwise use the hostname of the computer you are interested in determining the listening status of. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

Return value: Mute Status
Type: Boolean
Description: TRUE indicates that the speaker app instance is muted.

Purpose

This function returns the "pause" status of a specific speaker client (host or host:instance).

Parameters

Parameter: Host (Optional)
Type: String
Description: Leaving this a null string will return the status for the first instance HomeSeer finds, otherwise use the hostname of the computer you are interested in determining the pause status of. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

Return value: Pause Status
Type: Integer
Description: Bit encoded value indicating the status as follows:

Bit Values	Status
Bit 1 = 0	No Wavefile Instance
Bit 1 = 1	Wavefile Present
Bit 2 = 0	Wavefile Paused
Bit 2 = 1	Wavefile Playing
Bit 3 = 0	TTS Present
Bit 3 = 1	No TTS Present
Bit 4 = 0	TTS is currently speaking
Bit 4 = 1	TTS is not speaking

Purpose

This function mutes all speech and audio of a specific speaker client (host or host:instance).

Parameters

Parameter: Host (Optional)
Type: String
Description: Leaving this a null string will mute the first instance HomeSeer finds, otherwise use the hostname of the computer you are interested in muting. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

Purpose

This function pauses the audio currently playing at a specific speaker client (host or host:instance).

Parameters

Parameter: **Host (Optional)**
Type: String
Description: Leaving this a null string will pause the audio for the first instance HomeSeer finds, otherwise use the hostname of the computer you are interested in pausing audio on. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

Purpose

This function resumes all speech and audio of a specific speaker client (host or host:instance).

Parameters

Parameter: Host (Optional)
Type: String
Description: Leaving this a null string will mute the first instance HomeSeer finds, otherwise use the hostname of the computer you are interested in muting. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

Purpose

This function resumes the audio currently playing at a specific speaker client (host or host:instance).

Parameters

Parameter: Host (Optional)
Type: String
Description: Leaving this a null string will pause the audio for the first instance HomeSeer finds, otherwise use the hostname of the computer you are interested in pausing audio on. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

In This Section	
	MediaFilename MediaPlay MediaPause MediaMute MediaIsPlaying MediaStop MediaUnPause MediaVolume

Purpose

This is a read/write property. This function sets the file name that is to played using the speaker client. Call MEDIAPlay to actually start playing the selection.

This property may be read to get the selection currently playing.

Parameters

Parameter: filename
Type: string
Description: This sets the file name of the media selection to play. The file name may be any valid file supported by the Windows® Media Player.

Parameter: host (optional)
Type: string
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

Purpose

This function starts playing the selection as specified with the hs.MEDIAFilename property.

Parameters

Parameter: filename (optional)
Type: string
Description: This is the path and filename of the file to be played. If it is omitted here, it must have been previously set using the MEDIAFilename property.

Parameter: host (optional)
Type: string
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

Purpose

This function instructs the Windows® Media Player to pause the currently playing selection. The selection may be resumed by calling the hs.MediaPlay function.

Parameters

Parameter: host (optional)
Type: string
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

Purpose

This function mutes the media selection that's currently playing. The selection continues to play, but sound is not heard.

Parameters

Parameter: mute
Type: boolean
Description: Use TRUE to mute the selection and FALSE to unmute it.

Parameter: host (optional)
Type: string
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

Example

```
sub main()


' mute the Windows Media Player

hs.MediaMute TRUE


end sub
```

Purpose

This function checks if the media player is currently playing a selection.

Parameters

Parameter: host (optional)
Type: string
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

Return value: status
Type: boolean
Description: This returns TRUE if a media selection is currently playing and the sound card is most likely busy, and returns FALSE if a media selection is not playing and the sound is most likely free.

Purpose

This function instructs the Windows® Media Player to stop playing the current selection.

Parameters

Parameter: host (optional)
Type: string
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

Purpose

This function instructs the Windows® Media Player to resume the currently playing selection.

Parameters

Parameter: host (optional)
Type: string
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

Purpose

This is a read/write property. It sets and gets the current volume level of the playing media selection.

Parameters

Parameter: Level
Type: Integer (property)
Description: This sets the volume level. 100=full volume and 0 is the lowest volume.

Parameter: Host (optional)
Type: String
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

Example

```
sub main()


' get the current volume level

dim level

level = hs.MediaVolume


' set the volume to full

hs.MediaVolume = 100


end sub
```

In This Section	
	Speak SpeakEx SpeakProxy GetVoiceName MuteSpeech SetSpeakingSpeed SetVoice StopSpeaking PlayWavFile PlayWavFileVol

Purpose

This function speaks some text.

Parameters

Parameter: Text
Type: String
Description: This is the string you want to speak. It may also be the complete path to a WAV file to be played.

Parameter: Wait (Optional)
Type: Boolean
Description: If set to TRUE, the function will not return until the system finishes speaking. This is useful if you are switching between speaking and listening. You cannot listen and speak at the same time on some systems. If this parameter is missing, the system will not wait.

Parameter: Host (Optional)
Type: String
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

Example

```
Sub Main()

' speak and wait
hs.speak "hello there", True
hs.speak "Hello people in the kitchen.", True, "Kitchen:*"

End Sub
```

See Also

"Using Replacement Variables" in the HomeSeer help file.

Purpose

This function speaks some text and sends the output to the indicated output device. This function can be used to speak out other sound devices other than the normal sound card. For systems with multiple sound cards, this function can be used to select the specific card.

Parameters

Parameter: device
Type: integer
Description: This is the device number of the output device. Device 0 is usually the computer speakers and the default sound card.

Parameter: text
Type: string
Description: This is the text you want to speak.

Parameter: wait
Type: boolean
Description: If set to TRUE, the function will not return until the system finishes speaking. This is useful if you are switching between speaking and listening. You cannot listen and speak at the same time on some systems. If this parameter is missing, the system will not wait.

Parameter: Host (Optional)
Type: String
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

See Also

"Using Replacement Variables" in the HomeSeer help file.

Purpose

This function speaks some text after being handled by a speaker proxy program or plug-in.

This command passes along speak commands received from HomeSeer as a registered speaker proxy handler, and this is where the values for the parameters are provided. Therefore, this command is generally NOT used by a script and is primarily for plug-ins and applications.

Parameters

Parameter: Device
Type: Integer
Description: This is the sound device ID number for the TTS to be spoken at.

Parameter: Text
Type: String
Description: This is the string you want to speak. It may also be the complete path to a WAV file to be played.

Parameter: Wait
Type: Boolean
Description: If set to TRUE, the function will not return until the system finishes speaking. This is useful if you are switching between speaking and listening. You cannot listen and speak at the same time on some systems. If this parameter is missing, the system will not wait.

Parameter: **Host (Optional)**
Type: String
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

Purpose

This function returns the voice name of a specific speaker client (host or host:instance).

Parameters

Parameter: Host (Optional)
Type: String
Description: Leaving this a null string will return the voice name for the first instance HomeSeer finds, otherwise use the hostname of the computer you are interested in determining the voice name being used. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

Return value: voice name
Type: string

Purpose

This function temporarily mutes the speech output. By setting this property to FALSE, all speech output is silenced until this property is set back to TRUE. This mutes ALL speech, including speech generated from scripts.

This is a read/write property.

Parameters

Parameter: Mode
Type: Boolean
Description: Use TRUE to have speech output silenced and FALSE to have it enabled.

Returns

None.

Example

```
' stop all speech output

sub main()

hs.MuteSpeech = TRUE

end sub


' enable all speech output

sub main()

hs.MuteSpeech = FALSE

end sub
```

Purpose

This function sets the rate of HomeSeer's speech.

Parameters

Parameter: Speed
Type: Integer
Description: This is the speed parameter to be set. The range is -10 to 10. A value of zero is the normal rate of speaking.

Parameter: Host (Optional)
Type: String
Description: Leaving this a null string will set the speaking speed for the first instance HomeSeer finds, otherwise use the hostname of the computer you are interested in setting the speaking speed on. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

Return value: previous speed or 99
Type: integer
Description: This returns the previous speed setting or returns 99 if the input parameter was invalid. This is useful for returning the speaking speed to its previous value after making an adjustment.

Example

Purpose

This command changes the voice of a speaker client instance to the voice name provided.

Parameters

Parameter: VoiceName
Type: String
Description: This is the voice name string of the voice you wish to change the speaker client to use - it is not case sensitive but must match one of the voice names in your system. (See the Speaker Client for a list of voice names.)

Parameter: **Host (Optional)**
Type: String
Description: Leaving this a null string will change the voice for the first instance HomeSeer finds, otherwise use the hostname of the computer you are interested in changing the voice of. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

Return value: return status
Type: integer (.NET Short)
Description: Zero (0) means the voice was not found, One (1) indicates success.

Purpose

This function causes any speaking to stop immediately.

Parameters

Parameter: Host (Optional)
Type: String
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Returns

None.

Example

hs.StopSpeaking

Purpose

This function plays a specific WAV file out the default audio device. For more control over playing WAV files, see *PlayWavFileEx*.

Parameters

Parameter: FileName
Type: String
Description: This is the complete path to the WAV file to play.

Parameter: **Host (Optional)**
Type: String
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Parameter: **Wait (Optional)**
Type: Boolean
Description: Setting this to True will cause the command to wait until the WAV file is done playing before continuing. By default, it will not wait.

Returns

None.

Purpose

This function plays a WAV file and allows playing the WAV file in the background and setting the volume level.

Parameters

Parameter: Filename
Type: String
Description: This is the complete path to the WAV file to play.

Parameter: **volume (left)**
Type: **Integer**
Description: This is the volume level to use when playing. The range is 0 to 100. Set the value to -1 if you want to use the currently set volume level. I n previous versions of HomeSeer this was the LEFT volume level only - note that this is now the one and only volume level.

Parameter: volume (right)
Type: **Integer**
Description: This parameter is obsolete and remains for backward compatibility with previous versions of HomeSeer.

Parameter: Host (optional)
Type: String
Description: Leaving this a null string will apply the command to the first instance HomeSeer finds, otherwise use the hostname of the computer for this command. If more than one instance of the Speaker application is running on "host" then you may need to specify the instance as well in the format host:instance.

Parameter: Wait
Type: Boolean
Description: Use TRUE to not return until the WAV file has finished playing and FALSE to play the WAV file in the background. The function returns immediately.

Returns

None.