## Table of Contents

**Chapter 1: Welcome HS3**  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Things First</td>
<td>2</td>
</tr>
<tr>
<td>Installing Hardware Interfaces</td>
<td>3</td>
</tr>
<tr>
<td>Installing Software Interfaces</td>
<td>4</td>
</tr>
<tr>
<td>Creating Devices</td>
<td>5</td>
</tr>
<tr>
<td>Creating Events</td>
<td>6</td>
</tr>
<tr>
<td>Remote Access</td>
<td>9</td>
</tr>
</tbody>
</table>

**Chapter 2: QuickStart**  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Things First</td>
<td>2</td>
</tr>
<tr>
<td>Installing Hardware Interfaces</td>
<td>3</td>
</tr>
<tr>
<td>Installing Software Interfaces</td>
<td>4</td>
</tr>
<tr>
<td>Creating Devices</td>
<td>5</td>
</tr>
<tr>
<td>Creating Events</td>
<td>6</td>
</tr>
<tr>
<td>Remote Access</td>
<td>9</td>
</tr>
</tbody>
</table>

**Chapter 3: Using Events**  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Triggers</td>
<td>10</td>
</tr>
<tr>
<td>Event Actions</td>
<td>11</td>
</tr>
</tbody>
</table>

**Chapter 4: Setup**  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>General</td>
<td>13</td>
</tr>
<tr>
<td>Network</td>
<td>13</td>
</tr>
<tr>
<td>Email</td>
<td>14</td>
</tr>
<tr>
<td>Voice</td>
<td>17</td>
</tr>
<tr>
<td>Custom</td>
<td>17</td>
</tr>
</tbody>
</table>

**Chapter 5: Text-To-Speech**  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using Replacement Variables</td>
<td>19</td>
</tr>
</tbody>
</table>

**Chapter 6: Voice Recognition**  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>About Scripts</td>
<td>26</td>
</tr>
<tr>
<td>Creating A Script</td>
<td>26</td>
</tr>
<tr>
<td>Debugging Scripts</td>
<td>27</td>
</tr>
<tr>
<td>Executing Single Script Statements</td>
<td>28</td>
</tr>
<tr>
<td>VB.NET Scripts and NameSpaces</td>
<td>28</td>
</tr>
<tr>
<td>CSharp Scripting</td>
<td>29</td>
</tr>
<tr>
<td>Common Scripting Questions</td>
<td>30</td>
</tr>
<tr>
<td>Applications and Plugins</td>
<td>30</td>
</tr>
<tr>
<td>System Information</td>
<td>31</td>
</tr>
<tr>
<td>AppStarting</td>
<td>31</td>
</tr>
<tr>
<td>DebugMode</td>
<td>32</td>
</tr>
<tr>
<td>GetAppPath</td>
<td>33</td>
</tr>
<tr>
<td>InterfaceVersion</td>
<td>33</td>
</tr>
<tr>
<td>IsLicensed</td>
<td>34</td>
</tr>
<tr>
<td>ShuttingDown</td>
<td>35</td>
</tr>
<tr>
<td>SystemUptime</td>
<td>35</td>
</tr>
<tr>
<td>SystemUpTimeTS</td>
<td>36</td>
</tr>
</tbody>
</table>
Version .................................................................................................................. 36
System Functions .................................................................................................... 37
BackupDB ............................................................................................................... 38
PowerFailRecover .................................................................................................. 39
ScheduleFile ......................................................................................................... 39
Shutdown ............................................................................................................... 40
System .................................................................................................................. 40
INI File Editing ..................................................................................................... 41
ClearINISection ...................................................................................................... 41
GetINISection ....................................................................................................... 42
GetINISectionEx .................................................................................................... 42
GetINISetting ........................................................................................................ 43
SaveINISetting ..................................................................................................... 44
Plug-Ins ................................................................................................................ 44
GetHSPRef ............................................................................................................. 45
PluginFunction ....................................................................................................... 45
PluginPropertyGet ................................................................................................. 46
PluginPropertySet ................................................................................................. 47
GetPluginsList ....................................................................................................... 48
RegisterLinkEx ...................................................................................................... 48
Logging .................................................................................................................. 49
ClearLog ................................................................................................................ 49
GetLog .................................................................................................................... 50
LogEntry Structure ............................................................................................... 50
GetLog_FullFilter ................................................................................................. 51
GetLog_Date .......................................................................................................... 52
GetLog_Date_Text ................................................................................................. 53
GetLog_Date_Priority ............................................................................................ 54
GetLog_Date_ErrorCode ....................................................................................... 55
LogGet ................................................................................................................... 55
NoLog ..................................................................................................................... 56
WriteLog ............................................................................................................... 57
WriteLogEx ........................................................................................................... 57
WriteLogDetail ..................................................................................................... 58
Web Pages .............................................................................................................. 59
GetPageFooter ...................................................................................................... 59
GetPageHeader ..................................................................................................... 60
WebValidateUser ................................................................................................. 61
WebStatsPageViews ............................................................................................. 62
WebServerSSLPort .............................................................................................. 63
WebServerPort ..................................................................................................... 63
WebLoggedlnUser ................................................................................................. 64
GetUsers ................................................................................................................. 64
GetPlugLinks ........................................................................................................ 66
RegisterHelpLink .................................................................................................. 67
WebPageDesc Object ........................................................................................... 67
RegisterLinkEx ...................................................................................................... 68
WebPageDesc Object ........................................................................................... 69
UnRegisterHelpLinks ............................................................................................ 69
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callbacks</td>
<td>70</td>
</tr>
<tr>
<td>RegisterStatusChangeCB</td>
<td>70</td>
</tr>
<tr>
<td>UnRegisterStatusChangeCB</td>
<td>71</td>
</tr>
<tr>
<td>Launch</td>
<td>72</td>
</tr>
<tr>
<td>SendMessage</td>
<td>73</td>
</tr>
<tr>
<td>ReplaceVariables</td>
<td>73</td>
</tr>
<tr>
<td>Using Replacement Variables</td>
<td>74</td>
</tr>
<tr>
<td>Computer</td>
<td>75</td>
</tr>
<tr>
<td>Serial Port Communication</td>
<td>76</td>
</tr>
<tr>
<td>OpenComPort</td>
<td>76</td>
</tr>
<tr>
<td>OpenComPortTerm</td>
<td>77</td>
</tr>
<tr>
<td>OpenComPortEx</td>
<td>78</td>
</tr>
<tr>
<td>SetComPortRTS/DTR</td>
<td>80</td>
</tr>
<tr>
<td>SendToComPort</td>
<td>80</td>
</tr>
<tr>
<td>GetComPortCount</td>
<td>81</td>
</tr>
<tr>
<td>GetComPortData</td>
<td>81</td>
</tr>
<tr>
<td>CloseComPort</td>
<td>82</td>
</tr>
<tr>
<td>Network Information</td>
<td>82</td>
</tr>
<tr>
<td>GetIPAddress</td>
<td>83</td>
</tr>
<tr>
<td>GetLastRemoteIP</td>
<td>83</td>
</tr>
<tr>
<td>LANIP</td>
<td>84</td>
</tr>
<tr>
<td>Ping</td>
<td>84</td>
</tr>
<tr>
<td>WANIP</td>
<td>85</td>
</tr>
<tr>
<td>GetOSVersion</td>
<td>85</td>
</tr>
<tr>
<td>RecurseFiles</td>
<td>86</td>
</tr>
<tr>
<td>RecurseFilesEx</td>
<td>86</td>
</tr>
<tr>
<td>RestartSystem</td>
<td>87</td>
</tr>
<tr>
<td>UnZip</td>
<td>87</td>
</tr>
<tr>
<td>Zip</td>
<td>88</td>
</tr>
<tr>
<td>Keys</td>
<td>89</td>
</tr>
<tr>
<td>Devices</td>
<td>91</td>
</tr>
<tr>
<td>The Device Class</td>
<td>91</td>
</tr>
<tr>
<td>dvMISC</td>
<td>95</td>
</tr>
<tr>
<td>eRelationship</td>
<td>95</td>
</tr>
<tr>
<td>DeviceScriptChange</td>
<td>96</td>
</tr>
<tr>
<td>Device Value Status Pairs</td>
<td>96</td>
</tr>
<tr>
<td>VSPair</td>
<td>97</td>
</tr>
<tr>
<td>VSVGPairType</td>
<td>99</td>
</tr>
<tr>
<td>ePairStatusControl</td>
<td>99</td>
</tr>
<tr>
<td>ePairControlUse</td>
<td>99</td>
</tr>
<tr>
<td>DeviceVSP Methods</td>
<td>100</td>
</tr>
<tr>
<td>DeviceVSP_AddPair</td>
<td>100</td>
</tr>
<tr>
<td>DeviceVSP_ChangePair</td>
<td>101</td>
</tr>
<tr>
<td>DeviceVSP_CountAll</td>
<td>101</td>
</tr>
<tr>
<td>DeviceVSP_CountStatus</td>
<td>102</td>
</tr>
<tr>
<td>DeviceVSP_CountControl</td>
<td>102</td>
</tr>
<tr>
<td>DeviceVSP_ClearAll</td>
<td>103</td>
</tr>
<tr>
<td>DeviceVSP_ClearAny</td>
<td>103</td>
</tr>
<tr>
<td>DeviceVSP_ClearStatus</td>
<td>104</td>
</tr>
</tbody>
</table>
DeviceVSP_ClearControl .......................................................... 104
DeviceVSP_ClearBoth .................................................................. 105
DeviceVSP_Get .......................................................................... 105
DeviceVSP_GetStatus .................................................................. 106
DeviceVSP_GetAllStatus ............................................................. 106
DeviceVSP_PairsProtected .......................................................... 107
Device Value Graphic Pairs .......................................................... 107
VGPair .................................................................................... 108
VSVGPairType .......................................................................... 109
DeviceVGP Methods .................................................................... 109
DeviceVGP_AddPair ................................................................... 109
DeviceVGP_Count ....................................................................... 110
DeviceVGP_ClearAll .................................................................... 110
DeviceVGP_Clear ........................................................................ 110
DeviceVGP_Get .......................................................................... 111
DeviceVGP_GetGraphic ............................................................... 111
DeviceVGP_PairsProtected .......................................................... 112
Device Type ................................................................................ 112
DeviceTypeInfo Object .................................................................. 113
Device_API ................................................................................ 113
eDeviceAPI ............................................................................... 114
Device_API_Description (Read Only) .......................................... 115
Device_Type ............................................................................... 115
eDeviceType_GenericRoot .......................................................... 115
eDeviceType_Media .................................................................... 116
eDeviceType_Plugin .................................................................... 116
eDeviceType_Script .................................................................... 117
eDeviceType_Security ................................................................. 117
eDeviceType_SourceSwitch ......................................................... 118
eDeviceType_Thermostat ............................................................. 118
Device_Type_Description (Read Only) .......................................... 119
Device_SubType ......................................................................... 119
eDeviceSubType_SecurityArea .................................................... 119
eDeviceSubType_Setpoint .......................................................... 120
Device_SubType_Description ...................................................... 120
Device_Type_String ..................................................................... 121
Device Exists, Reference, Address and/or Code ............................. 121
DeviceExistsRef ....................................................................... 122
DeviceExistsAddress .................................................................. 122
DeviceExistsAddressFull ............................................................ 123
DeviceExistsCode ...................................................................... 124
GetDeviceRef ............................................................................ 124
GetDeviceRefByName ................................................................... 125
GetDeviceParentRefByRef .......................................................... 126
GetDeviceCode ......................................................................... 126
Creating, Deleting, or Accessing Devices ..................................... 127
NewDeviceRef ........................................................................... 127
GetDeviceEnumerator .................................................................. 128
GetDeviceByRef ......................................................................... 129
DeleteDevice ................................................................. 129
DeviceCount ........................................................................ 130
DeviceButtonAdd .............................................................. 130
Device Value, String, or Last Change ...................................... 131
DeviceValue ......................................................................... 131
DeviceValueEx ...................................................................... 132
DeviceValueByName ............................................................. 132
DeviceValueByNameEx .......................................................... 133
SetDeviceValue ................................................................... 134
SetDeviceValueByRef ........................................................... 134
SetDeviceValueByName ......................................................... 134
DeviceString ........................................................................ 136
DeviceStringByName ............................................................. 137
SetDeviceString ................................................................... 137
SetDeviceStringByName ......................................................... 137
DeviceTime .......................................................................... 138
DeviceTimeByName ............................................................. 139
DeviceDateTime ..................................................................... 140
SetDeviceLastChange ........................................................... 141
DeviceLastChange ................................................................ 141
DeviceLastChangeRef ........................................................... 142
On - Off .............................................................................. 143
IsOn .................................................................................. 143
IsOnByName ....................................................................... 144
IsOff ................................................................................ 145
IsOffByName ...................................................................... 145
Device Script Buttons ........................................................... 146
DeviceScriptButton_AddButton ............................................. 146
DeviceScriptButton_DeleteButton ......................................... 147
DeviceScriptButton_DeleteAll ............................................... 148
DeviceScriptButton_Locate ................................................... 148
DeviceScriptButton_List ....................................................... 149
Device Energy Management .................................................. 150
Energy_AddData, Energy_AddDataArray ................................... 150
EnergyData Class ................................................................ 150
enumEnergyDevice .............................................................. 150
enumEnergyDirection .......................................................... 150
Energy_SetEnergyDevice ....................................................... 152
enumEnergyDevice .............................................................. 152
Energy_AddCalculator, Energy_AddCalculatorEvenDay ............... 153
Energy_CalcCount ................................................................ 153
Energy_GetCalcByName, Energy_GetCalcByIndex ..................... 154
EnergyCalcData Class .......................................................... 154
Energy.GetData, Energy_GetArchiveData .................................. 155
EnergyData Class ................................................................ 155
enumEnergyDevice .............................................................. 155
enumEnergyDirection .......................................................... 155
Energy_RemoveData ............................................................. 156
Device Control API (CAPI) ...................................................... 158
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DisableEventByRef</td>
<td>186</td>
</tr>
<tr>
<td>Triggering Events</td>
<td>186</td>
</tr>
<tr>
<td>TriggerEvent</td>
<td>186</td>
</tr>
<tr>
<td>TriggerEventEx</td>
<td>187</td>
</tr>
<tr>
<td>DelayTrigger</td>
<td>188</td>
</tr>
<tr>
<td>TriggerEventAndWait</td>
<td>188</td>
</tr>
<tr>
<td>RemoveDelayedEvent</td>
<td>189</td>
</tr>
<tr>
<td>Modifying Events</td>
<td>190</td>
</tr>
<tr>
<td>AddDeviceActionToEvent</td>
<td>190</td>
</tr>
<tr>
<td>EventSetRecurringTrigger</td>
<td>191</td>
</tr>
<tr>
<td>EventSetTimeTrigger</td>
<td>191</td>
</tr>
<tr>
<td>EventSetVRTrigger</td>
<td>192</td>
</tr>
<tr>
<td>NewEventEx</td>
<td>193</td>
</tr>
<tr>
<td>NewEventGetRef</td>
<td>193</td>
</tr>
<tr>
<td>SaveEventsDevices</td>
<td>194</td>
</tr>
<tr>
<td>DeleteEvent</td>
<td>195</td>
</tr>
<tr>
<td>SetSecurityMode</td>
<td>195</td>
</tr>
<tr>
<td>Internet</td>
<td>196</td>
</tr>
<tr>
<td>FTP</td>
<td>196</td>
</tr>
<tr>
<td>FTP</td>
<td>198</td>
</tr>
<tr>
<td>FTP_LastError</td>
<td>198</td>
</tr>
<tr>
<td>SetRemoteTimeout</td>
<td>198</td>
</tr>
<tr>
<td>GetURL</td>
<td>199</td>
</tr>
<tr>
<td>GetURL</td>
<td>199</td>
</tr>
<tr>
<td>GetURLEx</td>
<td>200</td>
</tr>
<tr>
<td>GetURLIE</td>
<td>201</td>
</tr>
<tr>
<td>GetURLImage</td>
<td>201</td>
</tr>
<tr>
<td>GetURLImageEx</td>
<td>202</td>
</tr>
<tr>
<td>URLAction</td>
<td>203</td>
</tr>
<tr>
<td>SetRemoteTimeout</td>
<td>204</td>
</tr>
<tr>
<td>GenCookieString</td>
<td>204</td>
</tr>
<tr>
<td>Phone</td>
<td>205</td>
</tr>
<tr>
<td>Scripting_Phone_LINEClearDTMF</td>
<td>207</td>
</tr>
<tr>
<td>Scripting_Phone_WaitMS</td>
<td>208</td>
</tr>
<tr>
<td>Scripting_Phone_StopListening</td>
<td>209</td>
</tr>
<tr>
<td>Scripting_Phone_StartListening</td>
<td>209</td>
</tr>
<tr>
<td>Scripting_Phone_Speak</td>
<td>211</td>
</tr>
<tr>
<td>Scripting_Phone_SetSpeaker</td>
<td>212</td>
</tr>
<tr>
<td>Scripting_Phone_RestoreSettings</td>
<td>214</td>
</tr>
<tr>
<td>Scripting_Phone_MBSort</td>
<td>215</td>
</tr>
<tr>
<td>Scripting_Phone_MBSave</td>
<td>217</td>
</tr>
<tr>
<td>Scripting_Phone_MBNextUnreadMessage</td>
<td>218</td>
</tr>
<tr>
<td>Scripting_Phone_MBNextReadMessage</td>
<td>219</td>
</tr>
<tr>
<td>Scripting_Phone_MBNew</td>
<td>221</td>
</tr>
<tr>
<td>Scripting_Phone_MBMessageTime</td>
<td>221</td>
</tr>
<tr>
<td>Scripting_Phone_MBMessageName</td>
<td>222</td>
</tr>
<tr>
<td>Scripting_Phone_MBMessageLength</td>
<td>225</td>
</tr>
<tr>
<td>Scripting_Phone_MBMessageFrom</td>
<td>226</td>
</tr>
<tr>
<td>Scripting_Phone_MBMessageDate</td>
<td>228</td>
</tr>
<tr>
<td>Scripting_Phone_MBNew29</td>
<td>229</td>
</tr>
</tbody>
</table>
Scripting_Phone_HIPCmd .......................... 303
Scripting_Phone_HandsetOnHook .......................... 304
Scripting_Phone_GetLastVoiceCommand .......................... 306
Scripting_Phone_CreateMessageFilename .......................... 307
Scripting_Phone_ContactClass .......................... 308
Scripting_Phone_ClearLastVoiceCommand .......................... 311
Scripting_Phone_CIDNumber .......................... 313
Scripting_Phone_CIDName .......................... 314
Scripting_Phone_ADRSave .......................... 316
Scripting_Phone_ADRNew .......................... 317
Scripting_Phone_ADRGet .......................... 318
Scripting_Phone_ADRDelete .......................... 320
Scripting_Phone_ADRCount .......................... 321
Scripts .................................................. 322
GetScriptPath ........................................... 323
IsScriptRunning ........................................... 324
RunScript ........................................... 324
RunScriptFunc ........................................... 325
ScriptsRunning ........................................... 326
WaitEvents ........................................... 326
WaitSecs ........................................... 327
Speech Recognition ........................................... 328
Modifying Voice Recognition Commands ....................... 328
AddVoiceCommand ........................................... 328
ClearAllVoiceCommands ........................................... 330
Getting Last Voice Command Information .......................... 330
GetLastVRCollection ........................................... 331
cLastVR ........................................... 331
GetLastVRInfo ........................................... 332
cLastVR ........................................... 333
LastCommandSelected ........................................... 334
LastVoiceCommand ........................................... 334
LastVoiceCommandHost ........................................... 335
LastVoiceCommandInstance ........................................... 336
LastVoiceCommandPhone ........................................... 337
LastVoiceCommandRaw ........................................... 337
Controlling Speaker Clients ........................................... 338
GetListenStatus ........................................... 338
ListenMode ........................................... 339
ListenForCommands ........................................... 339
SetSpeaker ........................................... 340
StartListen ........................................... 340
StopListen ........................................... 341
Strings, Global Variables, and Encryption ....................... 341
Global Variables ........................................... 342
CreateVar ........................................... 342
DeleteVar ........................................... 343
GetVar ........................................... 343
SaveVar ........................................... 344
<table>
<thead>
<tr>
<th>Encryption</th>
<th>344</th>
</tr>
</thead>
<tbody>
<tr>
<td>EncryptString</td>
<td>345</td>
</tr>
<tr>
<td>EncryptStringEx</td>
<td>346</td>
</tr>
<tr>
<td>DecryptString</td>
<td>346</td>
</tr>
<tr>
<td>Counters</td>
<td>347</td>
</tr>
<tr>
<td>CounterValue</td>
<td>348</td>
</tr>
<tr>
<td>CounterReset</td>
<td>348</td>
</tr>
<tr>
<td>CounterIncrement</td>
<td>349</td>
</tr>
<tr>
<td>CounterDecrement</td>
<td>349</td>
</tr>
<tr>
<td>Timers</td>
<td>350</td>
</tr>
<tr>
<td>TimerValue</td>
<td>350</td>
</tr>
<tr>
<td>TimerReset</td>
<td>351</td>
</tr>
<tr>
<td>Time and Calendar</td>
<td>351</td>
</tr>
<tr>
<td>Time Related</td>
<td>351</td>
</tr>
<tr>
<td>LocalTimeZone</td>
<td>352</td>
</tr>
<tr>
<td>SolarNoon</td>
<td>352</td>
</tr>
<tr>
<td>Sunrise</td>
<td>353</td>
</tr>
<tr>
<td>SunriseDt</td>
<td>354</td>
</tr>
<tr>
<td>Sunset</td>
<td>354</td>
</tr>
<tr>
<td>SunsetDt</td>
<td>355</td>
</tr>
<tr>
<td>TimeZoneName</td>
<td>355</td>
</tr>
<tr>
<td>Calendar Related</td>
<td>356</td>
</tr>
<tr>
<td>DaylightSavings</td>
<td>357</td>
</tr>
<tr>
<td>DaysInMonth</td>
<td>357</td>
</tr>
<tr>
<td>DaysLeftInMonth</td>
<td>358</td>
</tr>
<tr>
<td>DaysLeftInYear</td>
<td>359</td>
</tr>
<tr>
<td>EvenOddMonth</td>
<td>360</td>
</tr>
<tr>
<td>EvenOddDay</td>
<td>360</td>
</tr>
<tr>
<td>GetLastWeekday</td>
<td>361</td>
</tr>
<tr>
<td>GetSpecialDay</td>
<td>362</td>
</tr>
<tr>
<td>IsSpecialDay</td>
<td>363</td>
</tr>
<tr>
<td>IsWeekday</td>
<td>364</td>
</tr>
<tr>
<td>IsWeekend</td>
<td>365</td>
</tr>
<tr>
<td>Moon</td>
<td>366</td>
</tr>
<tr>
<td>Weekdays</td>
<td>367</td>
</tr>
<tr>
<td>WeekEndDays</td>
<td>368</td>
</tr>
<tr>
<td>WeekNumber</td>
<td>369</td>
</tr>
<tr>
<td>WeekNumberEx</td>
<td>369</td>
</tr>
<tr>
<td>WeeksLeftInYear</td>
<td>370</td>
</tr>
<tr>
<td>WeeksLeftInYearEx</td>
<td>371</td>
</tr>
<tr>
<td>Text-To-Speech and Media</td>
<td>372</td>
</tr>
<tr>
<td>GetInstanceList</td>
<td>373</td>
</tr>
<tr>
<td>IsSpeakerBusy</td>
<td>373</td>
</tr>
<tr>
<td>SpeakToFile</td>
<td>374</td>
</tr>
<tr>
<td>Speaker Client Global Audio</td>
<td>374</td>
</tr>
<tr>
<td>SetVolume</td>
<td>375</td>
</tr>
<tr>
<td>GetVolume</td>
<td>375</td>
</tr>
<tr>
<td>GetMuteStatus</td>
<td>376</td>
</tr>
<tr>
<td>GetPauseStatus</td>
<td>376</td>
</tr>
</tbody>
</table>
Welcome to the HS3 online help system. Browse through the help pages by clicking on the icons below or selecting pages in the table of contents to the left. To quickly find specific product information, enter search criteria in the search box above and click the search button.

If you're unable to find what you're looking for in this help system, try these alternative resources:
- Our Website
- Our Helpdesk
or contact our support team:
HomeSeer Support Options

Most popular pages
- Welcome HS3
- QuickStart
- DeviceScriptButton_AddButton
- The Device Class
- Device Script Buttons
- DeviceTypeInfo Object
- Common Scripting Questions
- VSPair
- Using Replacement Variables
- Executing Single Script Statements
First Things First

Overview
HS3 is a powerful cross-platform web-based home automation system that works with a wide variety of technologies to control virtually everything in the home! With HS3, home owners can monitor and control their homes from anywhere in the world with smart phones, tablets and other similar devices. HS3 is available as a stand alone software program or as an embedded firmware in our gateway controller products.

Basic Architecture
HS3 is a web application that's designed to be run on Windows or Linux compatible computers or similar devices (including our gateway controllers). The application includes a web interface that's used to configure the system. The web interface may be accessed with any web browser on any desktop or mobile device. HSTouch is the graphical user interface (GUI) system that integrates with HS3 to allow home control via smart phones, tablets, desktop PCs and other devices. HSTouch is compatible with Apple, Android, Windows and Linux devices.

Setup 1-2-3
Here are the essential steps for setting up any HS3 system:

1. Install HS3 software or gateway controller
2. Install hardware interfaces (Z-Wave, Insteon, UPB, X10, etc)
3. Install software interfaces (plug-ins) to communicate with hardware interfaces
4. Create or import product devices and status devices
5. Create automation events
6. Install HSTouch on desktop or mobile devices

Important Terms
Please take a few moments to review the important terms below, as they will be referred to throughout this guide.

Product Device: A software representation of a physical device, including light switches, thermostats, door locks, sensors and other similar physical devices. Controlling a product device in the software will control the corresponding physical device (product) in the home. In the example below, a HomeSeer device "Light Switch" is set to "Dim 59%" and the corresponding light switch in the home is also set to 59%.
Status Device: A software representation of something other than a physical device that's used to track changing status or value of something. Status devices may be used to track circumstances, states of being, status or other attributes and may be used to influence the behavior of the automation system. For example, a status device named "Vacation Mode" with 'on' and 'off' status may change lighting events in the home when home owners are on vacation. As with product devices, status device may be manually changed or may be changed with automation events.

Event: An event is any action or group of actions that are activated by any combination of triggers and conditions. Events are very powerful and form the basis for automation in HS3.

Action: An action is a command to do something in HS3. Actions can control devices, launch events, send Emails, speak announcements, reminders or alerts and do much more.

Triggers and Conditions: A trigger is used to launch a HomeSeer event. Common triggers may include specific or relative times and dates, and the changing status or value of a device. A condition is a circumstance or state of being that's typically used in concert with a trigger to influence the behavior of a HomeSeer event. For example, you may wish to turn on a light switch at sunset provided you're not on vacation. "Sunset" is the trigger and your 'vacation mode' is the condition. A complete list of triggers and conditions may be found the Events area of HS3.

See also
Installing Hardware Interfaces
Installing Software Interfaces
Creating Devices
Creating Events
Remote Access

Installing Hardware Interfaces

A Hardware interface is a device that allows HomeSeer to communicate with products in your home. Hardware interfaces are available for a wide variety of home automation technologies including (but not limited to) Z-Wave, X10 and Insteon. Hardware interfaces are usually connected to a HomeSeer system via USB, RS232 or IP (Ethernet / WiFi) protocols.

In the example below, a Z-Wave hardware interface is connected to a HomeTroller gateway controllers. The HomeTroller issues a command to the interface and it, in turn, broadcasts a Z-Wave command to control a product in the home. Likewise, status information from the products are broadcast back to the interface which reports to the HomeTroller.

Example system based on Z-Wave technology:
A complete listing of hardware interfaces is available at homeseer.com. Consult with the user guide or product manual for details regarding installation of any interface onto your HomeSeer system.

See also
First Things First
Installing Hardware Interfaces
Creating Devices
Creating Events
Remote Access

Installing Software Interfaces

Software interfaces are "plug-ins" or drivers that allow HomeSeer systems to communicate with home automation technologies and hardware interfaces. Once you've attached your hardware interface, you'll need to download and install a compatible HomeSeer plug-in to make it work. To do this, open the PLUG-INS menu and click 'Manage'. A list of installed plug-ins appears at the top of the page. Additional plug-ins are listed at the bottom of the page. Select the plug-ins you wish to install and click the "Download and Install" button. Downloaded plug-ins will be installed and added to the list at the top. Enable plug-ins to run them.

See also
First Things First
Installing Hardware Interfaces
Creating Devices
Creating Events
Remote Access
Creating Devices

To control real products in your home (light switches, thermostats, plug-in modules, etc) or to track the changing status of your home, you must create “devices” in your HomeSeer system. The example below is from the “device management” function and shows a list of

The process for creating product devices varies with each technology.

Z-Wave: A Z-Wave “primary controller” (like our Z-Troller) is used to add Z-Wave devices to the network. Once the devices are added to the Z-Troller, that unit is connected to the HomeSeer system and the devices are imported. HomeSeer automatically creates product devices during the import process.

X10: X10 devices are created manually in the device management area. Each device has a unique housecode and unit code (A1, for example) and each device may be assigned unique characteristics (dimming, non-dimming, etc).

UPB: Creating UPB product devices is a 2 step process. The UPB network is created with a software program called "Upstart". That program generates a data file which works with the UPB software plug-in to manage UPB product devices.

See also
First Things First
Installing Hardware Interfaces
Installing Software Interfaces
Creating Events
Remote Access

Creating Events

An “Event” is one or more actions that are set into motion by a trigger. For example, a light may be turned on (Action) at Sunset (Trigger) or an announcement may be issued (Action) when a door is opened (Trigger). Events can be simple (like these) or they can be complex, involving multiple actions, triggers and conditions. Example: If motion is sensed (Trigger) and the light level of a room is low (condition) and it's a weekday (condition), the room light is turned on (Action) and a text message is sent to you (Action).

In this example, the event “Turn lights on at sunset” is triggered at sunset and turns on a table lamp and a floor lamp:
Organizing Events
With HS3, there's no limit to the number of events that may be created. Before creating events, we recommend developing a strategy for organizing events using "Event Groups". This will make it easier to locate events and you'll be able to assign 'group conditions' and 'group actions' to any or all events within a group.

Event Group Examples:

See also
First Things First
Installing Hardware Interfaces
Installing Software Interfaces
Creating Devices
Remote Access
Remote Access

All HomeSeer systems include a built-in web server which is designed to allow remote (and local) access for configuration and control. You may use this service to access your system when you are away, or from within your home. The service makes it easy to find your system on your network.

Benefits of the MyHomeSeer service:

- No need to set a static IP address on your HomeSeer system, DHCP will always work and your system will always be found
- No need to mess with your router configuration, no ports to forward
- All connections are SSL secured so access is done using HTTPS connections
- Simple configuration on your phone, no need to deal with IP addresses

Follow the steps below to enable remote access for the web and mobile device interfaces:

**Setting Up Remote Access using the MyHomeSeer service (preferred)**

MyHomeSeer is a free service that will give you remote access to your system from any web browser or phone. Setup is simple, no technical knowledge is necessary.

1. Register for a free account at myhs.homeseer.com. You will need your license ID and password for your HomeSeer system. For controllers, this information is printed on the bottom of the unit. For software, it was provided in your confirmation email. You can also request the information be sent to you from our support page at homeseer.com.
2. On the registration page enter your license ID and password, then create your own login information for the site. Your login is your email address. You must use a valid email as your account will not be activated until you confirm the email. Your email will only be used to notify you about issues with your account or the service. Enter a password for your account.
3. An email will be sent to you for you to confirm your account. Once this is done, you are ready to access your system.
4. Go to myhs.homeseer.com and make sure you are on the "LogIn" page. Log in with your account information that you just added and you should be re-directed to your HomeSeer system web interface. From there you can manage your system.
5. If an error displays or the page does not display, check the following:
   1. Make sure HomeSeer or your HomeTroller is running and connected to the internet.
   2. Try restarting the HomeSeer software or the HomeTroller unit. Wait a few minutes then try to log in.
   3. If you still cannot log in, go to find.homeseer.com and see if your system is found there. If so, click the link and see if you can connect to your system. If so, go into Setup and make sure MyHomeSeer is enabled.
6. To access your system using your phone, download the HSTouch application from the app store on your phone. Run the app and at the log in prompt enter your MyHomeSeer login information. You should be connected to your system.

**Note that if you have multiple HomeSeer systems, or are going to use cameras, sign up for our premium MyHomeSeer service. More information is available at myhs.homeseer.com**

This next section is for advanced users and requires some knowledge in router configuration

**Setting Up Remote Access for the Web Interface manually and connecting through your router (not using MyHomeSeer)**

1. Use the pull down menus and navigate to TOOLS>Setup and click the "Network" Tab. On the Network page, locate "Enable Server for Remote Access" is checked.
2. Locate the "Server Port" setting. This will normally be set to 80 by default. In some cases, Internet service providers block incoming connections on port 80. If this true for you, or if you'd like to enhance the security of your system, change your port setting to some other number (almost anything will work: 77, 123, 8080, 9999, 1357, 4455, etc..)
3. Network routers are designed to dynamically assign local IP addresses to attached devices (using a function called DHCP). However, DHCP can prevent remote access to your HomeSeer system. For best results, we recommend setting a "static IP address" for your HomeSeer system. There are 2 methods for doing this:
   
   **Method #1:** Use your router's "address reservation" function (if available) to assign a specific IP address based on the MAC address of your HomeSeer System's network card.
   
   **Method #2:** Manually set the IP address to something you know is not already used by another device. For most systems, this can be done using the Windows Network Control Panel. For the HomeTroller Zee, this can be done within the HomeSeer web interface by navigating to TOOLS>Setup>Network Tab and unchecking "Enable DHCP" in the "Ethernet Settings" section.
4. You'll need to set up "Port Forwarding" in your network router to channel incoming connections to your HomeSeer system. For example, if you've set your HomeSeer web server port to "123", you'll need to "Forward" port 123 to the static IP address of your HomeSeer system. Consult your router's manual for instructions on how to set up port forwarding.

**Setting Up Remote Access for the Mobile Device Interface and connecting through your router (not using MyHomeSeer)**

These steps should be done after completing the steps above for the manual connection.

1. Download and install the "HSTouch" or "HSTouchPad" app from the Apple or Android stores to your mobile device.
2. The default web server port for HSTouch is 10200. You'll need to set up "Port Forwarding" in your network router to channel incoming connections from your mobile device to your HomeSeer system. Follow the procedure above "Forward" port 10200 to the static IP address of your HomeSeer system.

See also
First Things First
Installing Hardware Interfaces
Installing Software Interfaces
Creating Devices
Creating Events
Using Events

Events are the core to automation in HS3. An event is the combination of a trigger and an action. For example, a single event can be used to turn on a device at a specific time, for example:

IF the time is at sunset then turn on the outside light

Events use simple "IF" "THEN" "ELSE" logic. They may contain multiple conditions in order to trigger the event:

IF the time is sunset
AND IF the the device "away mode" is ON
Then Turn on the outside lights

To configure an event, select "Events" from the View menu. Events are organized in Groups. When you display the events page you first see all of your groups. Groups allow you group together events such all of your evening events. Click the "+" next to the group name to expand the group and view all of the events in that group.

To create a new event, click on the green arrow icon and build your event, or click the "+" on an existing event to expand it.

Event actions are the actions that are taken when an event triggers. For example at sunset you might want to turn on a few lights:

IF the time is sunset then
Then Set Device front porch lights to On
Then Set Device garage lights to ON

There is no limit to the number of actions you can add to an event.

About Event Conditions

While not obvious, some event triggers are actually conditions and not triggers. In HS2, triggers and conditions were separated. This caused confusion and complicated setting up events. In HS3 triggers and conditions are treated the same, although not all conditions can be triggers. If a condition cannot be a trigger, you will not be allowed to select it after the "IF" statement, it will only be enabled after an "AND" statement.

For example, and IF statement such as:

IF Kitchen Lights changes and becomes On ....

This trigger is a trigger since the light is changing from one state to another. Ideally all of your triggers should start with a trigger like this. Pick a trigger that is an instant change such a device changing state, or a time change.

A trigger such as:

IF Kitchen Lights has a value equal to On ....

This is a condition, since there is no state change, it is simply testing if the light is On. What would normally happen with a trigger like this is that it will keep triggering as long as the light is On. HS3 checks conditions every second so it would trigger once a second. This is obviously not what you would expect and would cause problems. HS3 will only trigger this event if the device changes to On from some other state. This allows the event to trigger once when the condition becomes true. Once it triggers, it will not trigger again until the light changes to another state then back to On.

Ideally, a statement like this should be used after the "AND" statement and not after the "IF".

See also
Welcome HS3
QuickStart
Event Triggers

The following event triggers are available:

**By time including:**
- Time is at
- Time is before this
- Time is after this
- Time is sunrise
- Time is this before sunrise
- Time is this after sunrise
- Time is sunset
- Time is this before sunset
- Time is this after sunset
- Time is daytime (after sunrise, before sunset)
- Time is nighttime (after sunset, before sunrise)
- Time is before sunrise but after this:
- Time is before sunset but after this:
- Time is after sunrise but after this:
- Time is after sunset but after this:

**Device value including:**
- This device changes and becomes...
- This device has been/for for at least...
- This device has been/for exactly...
- This device has a value equal to...
- This device has a value that is not equal to...
- This device has a value that is greater than...
- This device has a value that is less than...
- This device has a value that just changed:
- This device had its value set to:
- This device just had its value set or changed.

**Recurring trigger including:**
- The event will automatically trigger every...
- The event will automatically trigger from the top of the hour every...
- The event will automatically trigger once per hour from the top of the hour at...

**Counter Value including:**
- A counters value is this:
- A counters value is this or multiple of this:
- A counters value has exceeded this:
- A counters value is this or has exceeded this:
- A counters value is this absolute value:
- A counters value is this or a multiple of this absolute value:
- A counters value has exceeded this absolute value:
- A counters value has exceeded this or a multiple of this absolute value:

**Timer Value including:**
- A timers value is:
- A timers value is less than:
- A timers value is more than:
**Voice Recognition including:**
- A phrase was recognized:
- A phrase was recognized from the telephone: (Windows only)
- A phrase was recognized from a microphone
- A phrase was recognized from a specific microphone (speaker client)
- Any phrase was recognized coming from the telephone (Windows only)

**Received Email**
Specify the contents of the email to trigger the event

**Manually Triggered**
The event is triggered from the "Run" button on the event. Any other triggers or conditions set on the event are ignored and the actions are executed.

**Note:**
There may be more triggers list in this section as plug-ins can add their own triggers. See the documentation associated with any loaded plugins to see if they supply their own triggers.

---

See also
Event Actions

---

**Event Actions**

The following event actions are available:

**Control a Device**
Includes options:
- Delay action for some time period
Includes advanced options:
- Set a device string to specific text

**Control a Timer**
Set Timer
Start Timer
Stop Timer
Resume Timer

**Control a Counter**
Increment Counter
Decrement Counter
Increment Counter by ...
Decrement Counter by ...
Reset Counter
Set Counter to ...

**Speak Something**
Includes advanced options:
- Speak the contents of a file
- Wait for speaking to finish
Speak out a specific speaker client

**Play an Audio File**
Includes advanced options:
Play audio out a specific speaker client
Wait for audio to finish

**Send an Email**
Includes advanced options:
BCC and CC recipients

**Wait (hours/ minutes/ seconds)**

**Run another Event**
Option: delay hours/minutes/seconds
Option: run only if other event conditions are True

**Run a Script or Script Command**
(Script Commands are not available on Linux)
(Linux only supports vb.net scripts, Windows supports both vb.net and vbscript)
See the scripting help file for more information on running scripts as well as available script commands.

**Run Commands from Received Email**
Email reception must be enabled on the E-Mail tab in setup.

**Run another program or process**
Select a program to run.

**Run the Event Group actions**
Event group actions are configured at the bottom of the event list for the current event group. The event group actions may be applied to any/all events in the group using this action.

**Cancel a delayed Device Action**
A device action can delay the control of a specific device. Use this action to cancel those actions before they start. For example, an event can have a device action that turns on a light with a delay of 5 minutes. If within that 5 minutes you wish to cancel that action, you can use this action to do so.

**Cancel a delayed Event Action**
The run event action can run an event delayed. Use this action to cancel that pending event.

**Change the systems audio output**
This action controls the audio on specific speaker clients. Actions are Mute/UnMute/Pause and Resume.

Other actions may appear in this list for enabled plug-ins.

See also
Event Triggers
# Setup

Setup is used to modify program settings. See the individual sections for explanations of all available settings.

## General

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration File</td>
<td>Selects the configuration to use. A configuration file is a database file that holds information about your devices and events.</td>
</tr>
<tr>
<td>Time Zone*</td>
<td>Select the time zone for the unit. The time zone is used to calculate the system time.</td>
</tr>
<tr>
<td>Location</td>
<td>Select the graphical location by state/city or country/city. This sets the proper longitude and latitude for your location which is used to calculate sunrise and sunset times.</td>
</tr>
<tr>
<td>Latitude</td>
<td>Set the latitude for your location. If your location is not in the Location list, you may enter your location here.</td>
</tr>
<tr>
<td>Longitude</td>
<td>Set the longitude for your location. If your location is not in the Location list, you may enter your location here.</td>
</tr>
</tbody>
</table>

## Other Settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Launch Web Browser on Startup^</td>
<td>When started, the home screen is launched in a browser. (if accessing the system remotely, keep this feature disabled)</td>
</tr>
<tr>
<td>Scripts cannot timeout^</td>
<td>When enabled, vbscripts will not display a timeout dialog if they run too long.</td>
</tr>
<tr>
<td>New Events are Disabled by Default</td>
<td>If enabled, new events are created disabled. This is useful if you are creating many events and do not want them triggering until they are fully configured.</td>
</tr>
<tr>
<td>Securit Offset +/- (minutes)</td>
<td>When an event has the security option enabled, this setting specifies the amount of randomly selected minutes that are added/subtracted from the trigger time.</td>
</tr>
<tr>
<td>HSSentry (Software System Monitor) ^</td>
<td>Enables a small monitoring program the monitors the built in web server. If the server does not respond, the system is restarted.</td>
</tr>
<tr>
<td>Temperature Scale</td>
<td>Sets the temperature display on display pages to display Fahrenheit or Celcius.</td>
</tr>
</tbody>
</table>

## Power Failure Recovery Settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Event catch-up upon power restoration</td>
<td>If enabled, any event that has the &quot;Include in Powerfailure Recovery&quot; checked will be examined at startup to see if it would have triggered while the power was off. If so, its actions are run. This allows devices to be restored to the proper state in the case where the software was not running for a period of time.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>Number of hours to catch-up</td>
<td>If “Enable Event catch-up upon power restoration” is enabled, this setting sets the number of hours the system goes back to determine which events should be evaluated.</td>
</tr>
<tr>
<td>Do not allow scripts to run during recovery</td>
<td>If “Enable Event catch-up upon power restoration” is enabled and this is checked, script actions to events are not run.</td>
</tr>
<tr>
<td>At startup, delay recovery by # seconds</td>
<td>If “Enable Event catch-up upon power restoration” is enabled, this setting controls how long of delay there is after startup before the recovery is started.</td>
</tr>
<tr>
<td><strong>Log Settings</strong></td>
<td></td>
</tr>
<tr>
<td>Enable Log</td>
<td>If enabled, various informational messages are written to a database file. This also enables the Log page from the View menu. On lower power controllers, you can gain some performance and reduce resource usage by disabling the log. On flash based systems, disabling the log will avoid extra wear on the flash.</td>
</tr>
<tr>
<td>Most recent entries at bottom</td>
<td>If enabled, the log shows the most recent entries at the bottom of the page. Otherwise the most recent entry is at the top of the page.</td>
</tr>
<tr>
<td>Log text-to-speech phrases</td>
<td>If enabled, all speak phrases are logged. If your system speaks a lot, you may want to disable this to reduce the number of log entries.</td>
</tr>
<tr>
<td>Max log size in megabytes (MB)</td>
<td>Determines the maximum size of the log database. On systems with limited disk or flash space, this limits the amount of disk or flash used. When the log size goes over this limit, older log entries are deleted from the log automatically.</td>
</tr>
<tr>
<td>Max number of days of log entries to keep</td>
<td>When the log is enabled, this specifies how many days worth of entries are kept in the log. Entries older than this are deleted from the log. Note that if the size of the log goes over the &quot;Max Log Size&quot; limit, entries could be deleted from the log before the &quot;Max Days&quot; number is reached.</td>
</tr>
<tr>
<td><strong>Energy Database Settings</strong></td>
<td>The energy database logs energy information for devices that monitor power.</td>
</tr>
<tr>
<td>Max size in megabytes (MB) for the Energy Database</td>
<td>Determines the maximum size of the energy database. On systems with limited disk or flash space, this limits the amount of disk or flash used. When the size goes over this limit, entries are compressed into smaller time ranges.</td>
</tr>
<tr>
<td><strong>Log Colors</strong></td>
<td>The log can be customized by specifying colors for specific log entries. For example, if the word &quot;error&quot; is detected in a log entry, that entire entry can be displayed in RED. Enter the word you want to search for in the “keyword” box, and the color. Click on the color picker icon to select a color. Leave the “Style Name” box empty unless you have created a particular CSS style that you want to apply. Click the disk icon on the right to save your settings.</td>
</tr>
</tbody>
</table>

* ZEE HomeTroller only
^ Does not apply to ZEE HomeTroller

See also
Network
Email
Voice
Custom
Manage users of the system. Users listed here can access the web interface as well as HSTouch. For HSTouch set the user as "Normal". The admin user has full access to the system, including event creation and device control. If a user is set to "Local", that user is used when accessing the system from the local network and the "No Password Required for Local Login" setting is enabled.

<table>
<thead>
<tr>
<th>Allow Scripts and Plug-Ins Access to the User List</th>
<th>If checked, scripts can access the user list to obtain user names.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>WIFI Settings</strong></td>
<td>ZEE only, set wifi settings when a wifi adapter is installed.</td>
</tr>
<tr>
<td><strong>Ethernet Settings</strong></td>
<td>ZEE only, set the system Ethernet settings such as DHCP or Static IP.</td>
</tr>
<tr>
<td><strong>Web Server Settings</strong></td>
<td></td>
</tr>
<tr>
<td>Server Port (80=default)</td>
<td>Set the port the web server uses for connections. Port 80 is the default and this is the default port for a web server. If you wish to hide your server, you can set a different port such as 8945. If you change the port, you will need to add the port to the web address that you use to access the system. For example, if you access your system with <a href="http://jimhome.com">http://jimhome.com</a>, you would need to add the port: <a href="http://jimhome.com:8945">http://jimhome.com:8945</a>.</td>
</tr>
<tr>
<td>Bind Server to IP Address</td>
<td>For systems with multiple network adapters, this setting will allow you to assign the web server to listen for connections on a specific adapter.</td>
</tr>
<tr>
<td>Log Server Errors</td>
<td>Enable to log any server errors to the event log.</td>
</tr>
<tr>
<td>Inactivity Logout</td>
<td>If enabled, users are automatically logged out if there is no activity for 10 minutes. This will prevent someone from accessing the system if a web browser was left open to one of the web pages.</td>
</tr>
<tr>
<td>Log Remote Logins</td>
<td>When enabled, logins to the web server from the Internet are logged with the remote IP address.</td>
</tr>
<tr>
<td>Log Local Logins</td>
<td>When enabled, logins to the web server from the local network are logged with the remote IP address.</td>
</tr>
<tr>
<td>No Password Required for Local Login</td>
<td>When enabled, no password is required when the user is connecting from the same network that HomeSeer is running on. Passwords will always be required when accessing from a different network such as the Internet.</td>
</tr>
<tr>
<td>Additional Local Subnets, Comma Separated</td>
<td>In some cases there may be more than one network active at a site. If all networks are to be treated as local, add the other subnets here. For example, a site has a local network of 192.168 and 192.167. Enter 192.168,192.167 in this box.</td>
</tr>
<tr>
<td>HomeSeer is Discoverable Using UPNP</td>
<td>When enabled, HomeSeer will send out a UPNP announcement every 30 seconds. This announcement can be monitored with the HomeSeer HSDiscover application or some other UPNP application. This allows HomeSeer to be found easily. If HomeSeer has a static IP, you may wish to disable this feature.</td>
</tr>
<tr>
<td>Enable IP Hack Blocking</td>
<td>When enabled, the web server will block access when too many invalid attempts are made to access the web server. An invalid attempt is some request that the web server does not support. See the next 3 settings to configure this.</td>
</tr>
<tr>
<td>Time to block triggered IP addresses (minutes)</td>
<td>When &quot;IP Hack Blocking&quot; is enabled, and an IP address is blocked, this setting determines how long an IP address is blocked before it can re-try a connection.</td>
</tr>
<tr>
<td>Invalid access 'hits' before block imposed</td>
<td>When &quot;IP Hack Blocking&quot; is enabled this setting determines how many invalid attempts are allowed before the IP address is blocked.</td>
</tr>
<tr>
<td>Time between 'hits' to count toward block being imposed (seconds) recovery</td>
<td>If 2 invalid accesses are attempted within this timeframe, it is added to the &quot;hit&quot; counter.</td>
</tr>
<tr>
<td>Page Views</td>
<td>This counter counts the number of times a page on the server was accessed.</td>
</tr>
</tbody>
</table>

### MyHomeSeer-Connect Settings

MyHomeSeerConnect is a service offered by HomeSeer that will map your systems IP address to a domain name. The IP address of your home will constantly change (unless you have a static IP address). Since the IP address changes, it is impossible to reliably access your home from the Internet. This service detects IP address changes and keeps your domain name pointing to the correct IP. For example, if you have this service, you can create your own name such as "smithhome.myhomeconnect.com". To access your system from the internet, you only need to enter "smithhome.myhomeconnect.com" in your web browser. Note that you will need to enable port forwarding in your router also. See the port forwarding section in this help file.

| MyHomeSeer Connect LicenseID | Enter your MyHomeSeerConnect license ID as given on your service receipt. |
| MyHomeSeer Connect Password | Enter your MyHomeSeerConnect password as given on your service receipt. |
| MyHomeSeer Connect Domain Name | Enter your domain name that you used when you signed up for the service. If you picked "smithhome", just enter that without the "myhomeconnect.com" extension. |
### Notification Email address (when IP changes)
If you would like to be notified when your IP address changes, enter an email address here. Note that you must have email configured on the "E-Mail" tab.

### Speaker Clients
Speaker clients are audio clients that are connected to the system from either the local system or a remote system. Note that the ZEE does not have a local speaker client, it always speaks out the local audio jack. The ZEE does support remote speaker clients that are running on Windows systems. HSTouch clients on IOS devices, Android devices, and Windows PC's are also considered speaker clients. These clients are used for audio output such as text-to-speech for announcements and voice recognition for control.

### Speaker Client network port (default is 10401)
Set the port that the system uses for speaker client connections. This port is for the Windows Speaker client only and does not affect HSTouch clients. Normally this setting should not be changed. If it is changed, the port number will need to be changed in the Windows Speaker Client also.

### Default clients to speak to, comma separated list, blank=all clients
By default, the system will send audio to call speaker clients that are connected. Event actions that speak or play audio can specify a specific client. If all events are to use the same set of speaker clients, but not all, a default list can be specified here. For example, you have 3 speaker clients but the one in the bedroom is not to be treated as a default client. Just enter the other 2 clients in this box like: office:default,kitchen:default.

You can get the name of the speaker client by having the client connect and then checking the list at the bottom of this page.

### Connected Speaker and HSTouch Clients
When a speaker client or HSTouch client connects, it will be listed in this section. From here you can get the name of the client for use in events.

---

### E-Mail Settings General

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Gmail</td>
<td>Enable to use your gmail account for accessing email. This is easiest way to enable email support in HomeSeer.</td>
</tr>
<tr>
<td>Gmail Username</td>
<td>Your gmail username.</td>
</tr>
<tr>
<td>Gmail Password</td>
<td>Your gmail password.</td>
</tr>
</tbody>
</table>

### E-Mail Settings Sending

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMTP Server</td>
<td>If using SMTP to send email (gmail disabled), this is the ip address or name of your SMTP server. SMTP is used for sending email.</td>
</tr>
<tr>
<td>SMTP Username</td>
<td>SMTP server username.</td>
</tr>
<tr>
<td>SMTP Password</td>
<td>SMTP server password.</td>
</tr>
<tr>
<td>Mail Domain (enter if your username does not include @domain part)</td>
<td>If your username does include &quot;@host&quot;, then enter the hostname here, such as &quot;yahoo.com&quot;.</td>
</tr>
<tr>
<td>Default From address</td>
<td>The default FROM address that is used for sending email. Normally, this has to be the email address of your account. If you enter a different email, your messages may not be sent.</td>
</tr>
<tr>
<td>Use SSL ^</td>
<td>Enable if your SMTP server requires SSL for communications.</td>
</tr>
<tr>
<td>Default &quot;TO&quot; E-Mail address</td>
<td>Default TO address that email messages are sent to. In events, you can specify a different email address if desired.</td>
</tr>
<tr>
<td>Default E-Mail Subject</td>
<td>The default subject of the email messages. In events, you can specify a different subject if desired.</td>
</tr>
<tr>
<td>Default E-Mail Message</td>
<td>The default body of the message. In events, you can specify a different message body.</td>
</tr>
</tbody>
</table>
### E-Mail Settings Receiving

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check E-Mail</td>
<td>Check you want to receive messages. Receiving email is only used for triggering events, such as sending an email command to trigger an event. If you do not plan on using this feature, you can leave this disabled.</td>
</tr>
<tr>
<td>Check Frequency (minutes)</td>
<td>How often your server is checked for new email messages.</td>
</tr>
<tr>
<td>Receiving Server (POP)</td>
<td>The server used for receiving email, this is either a server name or IP address.</td>
</tr>
<tr>
<td>Use SSL</td>
<td>Check if your POP server requires SSL for communications.</td>
</tr>
<tr>
<td>Server Port (default=110)</td>
<td>The port your POP server is using, normally this is port 110.</td>
</tr>
<tr>
<td>POP Username</td>
<td>Your POP server username.</td>
</tr>
<tr>
<td>POP Password</td>
<td>Your POP server password.</td>
</tr>
</tbody>
</table>

### Voice

This page controls voice recognition. See the voice recognition section for more information on using voice recognition.

The following controls are available:

- **Speak Recognized Voice Commands**: If enabled, the system will speak any commands it recognizes before executing the command.
- **Confirm Voice Commands Containing "all"**: If enabled, any voice command that contains the word "all" will need to be confirmed before it is executed.
- **Confirm "In" and "On" voice commands even if the device confirmation is turned off**: If enabled, any command containing the words "in" or "On" will require confirmation. This is a command such as "In five minutes turn off the outside lights", or "On Sunday turn on the outside lights for 2 hours".

The remaining settings on this page allow you to change the format of commands.

### Custom

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Site Settings</td>
<td></td>
</tr>
<tr>
<td>Home Page Title</td>
<td>Set the title that appears at the top of the home page, for example: &quot;Smith Home&quot;</td>
</tr>
<tr>
<td>Custom Page # title</td>
<td>You can add up to 4 custom pages that appear on the View menu. Enter a name for each custom page.</td>
</tr>
<tr>
<td>Custom Page # URL</td>
<td>For each custom page you add, enter the URL to the page in the &quot;Custom Page URL&quot; field. For example, &quot;<a href="http://www.homeseer.com">http://www.homeseer.com</a>&quot;.</td>
</tr>
<tr>
<td>Use Custom Page 1 as Home Page</td>
<td>Normally the Home button on the menu bar takes the user to the device management page. Use this setting to have the Home button go to the first Custom Page listed above.</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Require a Password for Custom Page 1</td>
<td>Enable if you would your first custom page require a password.</td>
</tr>
<tr>
<td>Show Device Image Column on Device Management Page</td>
<td>Enable to show the device image on the Device Management page.</td>
</tr>
<tr>
<td>Show Last Change Column on Device Management Page</td>
<td>Enable to show the last change column on the Device Management page. This is the last time that a specific device changed.</td>
</tr>
<tr>
<td>Show Device Address/Code Column on Device Management Page</td>
<td>Enable to show the device address/code on the Device Management page. The address/code contains device specific information, for example, Z-Wave includes the node ID in this information.</td>
</tr>
<tr>
<td>Show Device Type Column on Device Management Page</td>
<td>Enable to show the device type on the Device Management page. The device type typically describes the device.</td>
</tr>
</tbody>
</table>

**Device Location Display Options**

<table>
<thead>
<tr>
<th>Location 1 Label</th>
<th>The label used for the first location. This location label cannot be disabled and is typically a &quot;Room&quot; label. The location is assigned to devices.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Location 2</td>
<td>Optionally, a second location can be used to describe a device. Enable this to use the location.</td>
</tr>
<tr>
<td>Location 2 label</td>
<td>The label to use for location 2, Typically this is set to &quot;Floor&quot;.</td>
</tr>
<tr>
<td>Display Location 1 First</td>
<td>Normally this is disabled since if you use &quot;Room&quot; for location 1 and &quot;Floor&quot; for location 2 you would want to display the &quot;Floor&quot; first. For example, you have a device that has location &quot;First Floor&quot; and location set to &quot;Kitchen&quot;, you would want the location to be describe as &quot;First Floor Kitchen&quot;. You can swap this if this option is enabled.</td>
</tr>
<tr>
<td>Filter Display by Location</td>
<td>When displaying devices, the filter is done based on the location. For example, you select &quot;First Floor&quot; for location 2, then only the devices that have location 2 set to &quot;First Floor&quot; will be displayed. If this option is set, the filter is then on location 1, then location 2...</td>
</tr>
</tbody>
</table>

**Startup**

<table>
<thead>
<tr>
<th>Startup Script</th>
<th>The script to run when the system starts, the default is &quot;Startup.vb&quot;. Note that on Linux systems only .vb scripts are supported. VBScripts (.txt&quot; scripts are not supported on Linux.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shutdown Script</td>
<td>The script to run when the system shuts down.</td>
</tr>
</tbody>
</table>

See also

General
Network
Email
Voice
Text-To-Speech

Text-to-Speech (TTS) is the ability to speak text. Text may be spoken in many ways. The most basic way is by creating an event and setting the action to "Speak Something". Depending on your configuration, the spoken text may be spoken out the soundcard of your system, or the soundcard of a remote system.

On the ZEE box, TTS is spoken out the audio out jack. It is handled by a script named "speak.sh" in the HomeSeer folder. As shipped, the unit uses the "flite" speech engine. Advanced users are welcome to modify this script to use other speech engines.

On Windows systems the speech is handled by a separate application called "Speaker". By default, this application is started when HomeSeer starts. This application uses a network connection to connect to HomeSeer and handle all speaking, as well as voice recognition. You can run this application on other Windows systems to speak remotely. The speaker client can even connect to ZEE boxes so those controllers can speak to remote systems.

When speaking, there are numerous replacement variables available that will allow the system to speak such things as the date and time. See the Using Replacement Variables section for more information.

See also
Welcome HS3
QuickStart
Using Events
Setup
Voice Recognition
Scripting

Using Replacement Variables

Replacement variables are a series of special characters that you can use in text being spoken or in the subject or body of an email. When HomeSeer encounters one of these variables, it substitutes the information indicated by the variable in place of the variable.

Example

```
hs.Speak "The time is $$time"
```

Results in (at 11AM): "The time is 11:00 AM"

**HomeSeer Replacement Variables**

(Replacement Variables are Case Insensitive)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$date</td>
<td>Replacement is the current date in long format, e.g.: April 1, 2006</td>
</tr>
<tr>
<td>$time</td>
<td>Replacement is the current time in 12 hour format, e.g. 2:00 PM</td>
</tr>
<tr>
<td>$$date</td>
<td>Replacement is the same as $date, but it is wrapped with the SAPI context tag for date so the text is a date being spoken. Use $$date when the output is going to be spoken.</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>$$time$$</td>
<td>Replacement is the same as $time, but it is wrapped with the SAPI context tag for time so the text will be spoken. Use $$time$$ when the output is going to be spoken.</td>
</tr>
<tr>
<td>$from$</td>
<td>Replacement is the email address of the last email received.</td>
</tr>
<tr>
<td>$$DVA:(address)$$</td>
<td>Replacement is the VALUE of the device indicated by (address). For example, if the device at address R40 has a value of 100, then using $$DVA:R40:$$ in the text will result in 100 after the substitution.</td>
</tr>
<tr>
<td>$$DVC:(code)$$</td>
<td>Same as $$DVA$$ but gets the device value using the device code.</td>
</tr>
<tr>
<td>$$DVR:(ref)$$</td>
<td>Same as $$DVA$$ but gets the device value using the device reference number.</td>
</tr>
<tr>
<td>$$DSA:(address)$$</td>
<td>Replacement is the STATUS of the device indicated by (address). For example, if the device at address S39 has a status of &quot;Disarmed&quot;, then using $$DSA:S39:$$ in the text will result in &quot;Disarmed&quot; after the substitution.</td>
</tr>
<tr>
<td>Note:</td>
<td>HTML used in the status may result in problems when the replaced text is spoken.</td>
</tr>
<tr>
<td>$$DSC:(code)$$</td>
<td>Same as $$DSA$$ but gets the status using a device code.</td>
</tr>
<tr>
<td>$$DSR:(ref)$$</td>
<td>Same as $$DSA$$ but gets the status using a device reference number.</td>
</tr>
<tr>
<td>$$DTA:(address)$$</td>
<td>Replacement is the STRING of the device indicated by (address). For example, if the device at address S39 has a string of &quot;Come listen to a story about a man named &lt;b&gt;Jed&lt;/b&gt;&quot;, then using $$DTA:S39:$$ in the text will result in &quot;Come listen to a story about a man named Jed&quot; after the substitution.</td>
</tr>
<tr>
<td>Note:</td>
<td>HTML used in the status may result in problems when the replaced text is spoken.</td>
</tr>
<tr>
<td>$$DTC:(code)$$</td>
<td>Same as $$DTA$$ but gets the string using a device code.</td>
</tr>
<tr>
<td>$$DTR:(ref)$$</td>
<td>Same as $$DTA$$ but gets the string using a device reference number.</td>
</tr>
<tr>
<td>$$LCI$$</td>
<td>(Windows Only) Replacement is information about the last phone caller, caller ID information.</td>
</tr>
<tr>
<td>$$CIN$$</td>
<td>(Windows Only) Replacement is the caller ID name of the last call.</td>
</tr>
<tr>
<td>$$CI#$$</td>
<td>(Windows Only) Replacement is the caller ID number of the last call.</td>
</tr>
<tr>
<td>$$LVM$$</td>
<td>(Windows Only) Replacement is the last voice message that was left (who left it, when it was left, and how long the message was)</td>
</tr>
<tr>
<td>$$COUNTER:(name)$$</td>
<td>Replacement is the value of a specific counter. If you have a counter named &quot;dryer_counter&quot; then you could speak the value of this counter with: $$COUNTER:dryer_counter:$$</td>
</tr>
<tr>
<td>$$TIMER:(name)$$</td>
<td>Replacement is the value of a specific timer. If you have a timer named &quot;dryer_timer&quot; then you could speak the value of this timer with: $$TIMER:dryer_timer:$$</td>
</tr>
</tbody>
</table>

See also
Voice Recognition

Voice recognition is handled by the speaker client application. While voice recognition is not supported on Linux, it is supported by running the speaker client on a Windows system and then connecting to your Linux system.

The speaker client uses the Microsoft voice recognition built into Windows. To start recognizing, click on the "Start Listening" button then speak into the microphone. When listening starts, the system is only listening for an attention phrase. By default this phrase is "computer". So before you can say any commands you need to get the computer's attention by saying "computer". By default, the computer will respond with "yes sire". After this phrase is heard, the computer is then listening for command.

The commands you can issue are dependent on your system configuration. If no events are created and no devices are added, you cannot issue any commands!

In the examples below, words surrounded with [] are optional and are not required to be spoken.

To control a device, create or add a device, then click on the device name to display its properties. Check the "Voice Command" checkbox to enable this device for voice commands. Once done, you can then issue commands to control the device such as: (assuming the device is named "lights" and its location 1 is "kitchen" and its location 2 is "first floor":

"turn on [the] lights"
"turn on [the] kitchen lights"
"turn on [the] first floor kitchen lights"
"turn off [the] kitchen lights"
"shut off [the] kitchen lights"
"kitchen lights off"
"kitchen lights on"
"kitchen lights 50%"
"set the kitchen lights to 50%"
"dim the kitchen lights to 20%"

The following 3 commands will control all the devices that have "light" or "lamp" in the device name and the device is marked for a voice command
"turn all [the] lights off"
"turn all [the] lights on"
"shut all [the] lights off"
"turn [all] [the] lights off"
"shut [all] [the] lights off"
"turn [all] [the] lights on"
"turn on [all] [the] lights"
"turn off [all] [the] lights"
"shut off [all] [the] lights"

To control events, add a "phrase was recognized" trigger to your event. For example, you can add a phrase such as:

"all outside lights on"

You can then say this phrase to trigger the event.

Creating events with voice

It is also possible to create new events using your voice. For example, you want to turn on the outside lights tonight for 4 hours. You can say:

"at seven oclock turn on the porch lights for 4 hours"
"January [the] fifth at six oh clock pm turn on the kitchen lights"

"in five minutes turn [the] office light off"

"in six hours turn [the] office light on"

"in 2 hours turn [the] office light on for two hours"

The format of the event creation command is as follows, "|" separates optional words where only one word may be spoken. "[ ]" indicates word is optional. "<R>" indicates that the next word may be repeated, such as "forty five". 

```plaintext
in
< R >(one|two|three|four|five|six|seven|eight|nine|ten|eleven|twelve|thirteen|fourteen|fifteen|sixteen|seventeen|eighteen|nineteen|twenty|twenty five|thirty|thirty five|forty|forty five|fifty|fifty five|sixty|sixty five|seventy|seventy five|eighty|eighty five|ninety)
(seconds|minutes|hours|days|minute|hour|day)
(turn|shut)
(on|off)
[the]
[device1|device2|etc.]
[for]
< R >(one|two|three|four|five|six|seven|eight|nine|ten|eleven|twelve|thirteen|fourteen|fifteen|sixteen|seventeen|eighteen|nineteen|twenty|twenty five|thirty|thirty five|forty|forty five|fifty|fifty five|sixty|sixty five|seventy|seventy five|eighty|eighty five|ninety)
(seconds|minutes|hours|days|minute|hour|day)
(turn|shut)
(on|off)
[the]
[device1|device2|etc.]
[for]
< R >(one|two|three|four|five|six|seven|eight|nine|ten|eleven|twelve|thirteen|fourteen|fifteen|sixteen|seventeen|eighteen|nineteen|twenty|twenty five|thirty|thirty five|forty|forty five|fifty|fifty five|sixty|sixty five|seventy|seventy five|eighty|eighty five|ninety)
(seconds|minutes|hours|days|minute|hour|day)
(turn|shut)
(on|off)
[the]
[device1|device2|etc.]
[for]
< R >(one|two|three|four|five|six|seven|eight|nine|ten|eleven|twelve|thirteen|fourteen|fifteen|sixteen|seventeen|eighteen|nineteen|twenty|twenty five|thirty|thirty five|forty|forty five|fifty|fifty five|sixty|sixty five|seventy|seventy five|eighty|eighty five|ninety)
(seconds|minutes|hours|days|minute|hour|day)

[next]
(todays|tomorrow|sunday|monday|tuesday|wednesday|thursday|friday|saturday)
at
(one|two|three|four|five|six|seven|eight|nine|ten|eleven|twelve)
< R >(zero|oh one|oh two|oh three|oh four|oh five|oh six|oh seven|oh eight|oh nine|ten|eleven|twelve|thirteen|fourteen|fifteen|sixteen|seventeen|eighteen|nineteen|twenty|twenty one|twenty two|twenty three|twenty four|twenty five|twenty six|twenty seven|twenty eight|twenty nine|thirty|thirty one|thirty two|thirty
```
Voice Recognition

HomeSeer HS3 - End User Documentation
An event will be created to perform this action. See the voice recognition tab in setup for the acknowledge phrases. These are the phrases the system replies with to verify the event creation commands

**Canceling Event Actions**

When future event is created, it may be cancelled. This is done by specifying the device that is to be controlled by the event. For example, you issued the command:

in 2 hours turn on the outside lights

But you decide you do not want this to happen. You can cancel with the command:

"cancel the delayed action for the outside lights"

The event will be deleted.

The format of this command is:

cancel [the] [delayed] [action] [for] [the] (device1|device2|etc.)

Most words are optional for this command, so the shortest phrase could be:

"cancel outside lights"

**Running event actions**

Any event can be run by name and so setup on the event is required. To run the event named "turn on all the lights", you can say:

"run [the] event turn on all the lights"

"in ten minutes run [the] event turn on all the lights"

"at nine oh clock pm run [the] event turn on all the lights"

If an event has a voice phrase set as a trigger, that can be used instead. For example, an event has a voice trigger set as "all lights off", you can say:

"all lights off"

"in ten minutes all lights off"

Reminders
Reminders may be created by voice. For example, you want the system to announce when you have to leave the house. You could say:

"in one hour remind me to pick up the kids"

By default, this command will simply announce "pick up the kids" in one hour over the system's audio system using text to speech.

The system can also send reminders via email. This first needs to be set up on the reminders page which is linked off the Tools menu. For example, on the reminders page add an email reminder for Bill like:

Name the reminder "Bill via email", set the "Remind Method" to "Email", and the "Destination" to Bill's email address.

Now by voice you can create an email reminder by saying:

"in one hour remind bill via email to pick up the kids"

In one hour an email will be sent to Bill's email address.

Other forms of this command:

"at two oh clock P M tomorrow remind bill via email to pick up the kids"
"march sixth at eight oh clock A M remind bill to go to the dentist"
"next friday at six P M remind bill via email that he has a haircut appointment"

To check that future event commands are created, you view them on the events page under the group "Delayed Voice Actions".

See also
Welcome HS3
QuickStart
Using Events
Setup
Text-To-Speech
Scripting
Scripting

In This Section

About Scripts
Applications and Plugins
Computer
Devices
Email
Events
Internet
Phone
Scripts
Speech Recognition
Strings, Global Variables, and Encryption
Time and Calendar
Text-To-Speech and Media

See also
Welcome HS3
QuickStart
Using Events
Setup
Text-To-Speech
Voice Recognition

About Scripts

In This Section

Creating A Script
Debugging Scripts
Executing Single Script Statements
VB.NET Scripts and NameSpaces
CSharp Scripting
Common Scripting Questions

See also
Applications and Plugins
Computer
Devices
Email
Events
Internet
Phone
Scripts
Speech Recognition
Strings, Global Variables, and Encryption
Time and Calendar
Text-To-Speech and Media
Creating A Script

All scripts must reside in the scripts folder in the HomeSeer application directory. Scripts commonly have the extension ".txt" or ".vbs" or ".vb" (for vb.net scripts) or .cs for C# scripts, and are simple text files. You can edit your scripts in this directory. Here are the steps to create a simple script that turns a light off if it's on:

- In the Events View, expand a group, then click the + button to create a new event, the event opens.
- A box will display to name the event, give the event a name like "light on test".
- Under the "IF" drop list, select "manually triggered" so the event is only triggered when we run it manually.
- Click the drop list next to the "THEN" to select an action and select "Run a script or script command".
- Click the red airplane icon to go into advanced mode so we can input a script. Click the Edit button to select a script. In the window that pops up, add "light_test.txt" to end of the path, then click "Submit" to close the dialog.
- Now expand the actions again to view the script edit box. Paste in the code below and click "Save Script Edits" to save the script.
- The script already has the main subroutine defined for you. Modify the script so it looks like this:

```vbs
Sub Main()
    Dim DevRef
    DevRef = hs.GetDeviceRefByName("Living Room Light")
    If hs.ison(DevRef) Then
        hs.Speak "The living room light is currently on."
    End If
End Sub
```

- To run the script, click the blue run button, check the event log for errors.
- VB.NET scripts use a slightly different format. The script is always passed a "parms" object which will be an array of objects that are the parameters. The object will be nothing if no parameters are supplied. The above script would be formatted as follows, and the name of the script needs to have a .vb extension.

```vbnet
Sub Main(parms As Object)
    Dim DevRef As Integer
    DevRef = hs.GetDeviceRefByName("Living Room Light")
    If hs.IsOn(DevRef) Then
        hs.Speak("The living room light is currently on.")
    End If
End Sub
```

- You can test script statements using the Control screen. Select the Control Panel link from the Tools menu. In the "Immediate Script Command" box, enter your script command. For example, to speak a phrase enter:

  ```vbs
  hs.speak "hello"
  ```

  The system should speak. If you want to get the value of a HomeSeer device (for example, the current light value from a HSM100 sensor), first get the name of the device. If your light device was named "Family Room Light", then enter this command in the Script Command box:

  ```vbs
  hs.writelog "msg",hs.devicevaluebyname("Family Room Light")
  ```

  Check your event log, you should see an entry like:

  9/19/2009 12:45:18 PM - msg - 98

See also
- Debugging Scripts
- Executing Single Script Statements
- VB.NET Scripts and NameSpaces
- CSharp Scripting
- Common Scripting Questions
Debugging Scripts

If your script has errors, the scripting engine will detect the error whenever the script is run. The error will appear in the event log. If your script is not working, check the log for errors. The log will contain the line number where the error occurred.

Do not allow your script to run for more than a few seconds. Scripts are to be used to perform a quick task that does not take a lot of time. The script engine will prompt you with a dialog box warning you that the script is taking a long time to run if the script is running for longer than 30 seconds. You can work around this, however. You can call the script function `hs.WaitEvents()` or `hs.WaitSecs()`. If you call this function within 30 seconds, the script will not time out. This will also let HomeSeer do other tasks while your script is executing.

See also
Creating A Script
Executing Single Script Statements
VB.NET Scripts and NameSpaces
CSharp Scripting
Common Scripting Questions

Executing Single Script Statements

In the Advanced section of the Run Script action, you can add a single script statement. This allows you to execute script commands without creating a file. Statements are preceded with an ampersand (&) so HomeSeer knows to treat it as a statement. For example, the following if then else logic could be typed into the “OR Script Statement” field of the Run Script action:

For VBScripts and VB.Net scripting:

```
&if hs.ison(1234) then hs.SetDeviceValue 1234,0 Else hs.SetDeviceValue 1234, 100
or
&hs.SetDeviceString 5678,"Garage Open",false
```

- Multiple statements may be added to the “OR Script Statement” field. Separate each statement with a colon (;

Note: On Windows, only VBScript formats are supported for immediate script commands. On Linux, immediate script commands use the VB.Net format. On Linux, VB.Net statements are compiled as a full script and can be slow on some systems. For better performance, use C# script statements on Linux

Note: C# scripts statements are supported on both Windows and Linux

An example C# script statement to speak would be:

```
&shs.Speak("hello",true,""");
```

See the CSharp scripting section for more information formatting C# scripts.

See also
Creating A Script
Debugging Scripts
VB.NET Scripts and NameSpaces
CSharp Scripting
Common Scripting Questions

VB.NET Scripts and NameSpaces

.NET uses "Namespaces" to refer to large libraries of code, which are not included in your VB.NET program or script unless you tell it that you wish to have it included.
By default, VB.NET scripts executed by HomeSeer will have the System.dll referenced, which means that to use a namespace within that library, add the IMPORTS statement to the top of your VB.NET script like this:

```
IMPORTS System.IO
IMPORTS System.Net
```

- Note that all namespaces under System are not necessarily included in System.dll - some System namespaces are in additional dll files.

If you wish to use other namespaces referenced in other libraries, you must first tell HomeSeer to include the library reference when it initializes the script engine. To do this, add your reference to the INI entry “ScriptingReferences”, which is under the [Settings] section in your \Config\Settings.INI file.

You can add references by including the namespace, a semicolon, and the dll file name in this entry. Multiple references can be added by separating them with a comma. Here is an example that adds a reference to 2 namespaces:

```
ScriptingReferences =

System.Management
;System.Management.dll

System.Drawing
;System.Drawing.dll
```

When broken down into individual parts:

```
ScriptingReferences =

System.Management
;System.Management.dll

System.Drawing
;System.Drawing.dll
```

See also
- Creating A Script
- Debugging Scripts
- Executing Single Script Statements
- CSharp Scripting
- Common Scripting Questions

---

CSharp Scripting

Scripts can also be written in C#. This is supported on both Windows and Linux and offers much more performance over vb.net scripting on Linux.

C# scripts use the extension .cs. An example script:

```csharp
public object Main(object[] Parms)
{
    hs.WriteLog("CSHARP","From C# script");
    String s = DateTime.Now.ToString("ddd");
    hs.WriteLog("C#","Day: " + s);
    Console.WriteLine("hello");
    return 0;
}
```

C# may also be used as script statements. Script statements start with an &, and C# scripts also need one other character to tell the script engine if it is a sub or a function. If the statement returns a value, then use "f", else if it does not return a value, use "s". For example, this sub simply writes to the log:

```
&shs.WriteLog("script","hello");
```

This statement returns a value:

```
&fDateTime.Now.ToString("ddd");
```

The scripting system knows when a script statement is C# by looking for a terminating semicolon ";". If one does not exist, it assumes either VBScript (for Windows), or VB.Net (for Linux)

C# is considerably faster on Linux than VB.Net for script statements.
Common Scripting Questions

How do you know when to use parenthesis when calling the scripting functions?

For Older VBScript Scripts, you do not need parenthesis if the procedure is not returning a value:

If the function is returning a value, you need to surround the parameters with parentheses. Otherwise, you need to omit them. Here is a function call that does not return a value:

```vbnet
hs.Speak "This is a test"
```

The following function returns a status value:

```vbnet
Dim MyVal
MyVal = hs.DeviceValue(1234)
```

For VB.NET Scripts, parenthesis are always used.

```vbnet
hs.Speak("This is a test")
Dim MyVal As Double
MyVal = hs.DeviceValue(1234)
```

For C# scripts, the format is the same as C#.

Are the function names case-sensitive?

Not for VBScript or VB.Net scripts. The function `hs.Speak("Test")` and `hs.speAK("Test")` are identical and work the same way. For C# scripts, they are case sensitive.
See also
About Scripts
Computer
Devices
Email
Events
Internet
Phone
Scripts
Speech Recognition
Strings, Global Variables, and Encryption
Time and Calendar
Text-To-Speech and Media

System Information

In This Section

AppStarting
DebugMode
GetAppPath
InterfaceVersion
ISLicensed
ShuttingDown
SystemUptime
SystemUpTimeTS
Version

See also
System Functions
INI File Editing
Plug-Ins
Logging
Web Pages
Callbacks
Launch
SendMessage
ReplaceVariables

AppStarting

Purpose

Indicates if the HomeSeer application is currently starting or doing startup processing.
Parameters

None.

Returns

Return value: status
Type: boolean
Description: Returns True if the application is starting or doing startup processing.

See also
DebugMode
GetAppPath
InterfaceVersion
IsLicensed
ShuttingDown
SystemUpTime
SystemUpTimeTS
Version

DebugMode

Purpose

This command sets or disables HomeSeer's built-in debugging trace functions. When enabled, debug information is written to the "Debug Logs" directory into a separate file for each debug type enabled. A "Debug_Composite.log" file in the debug logs directory contains all debug information written in all logs, and the "Debug_Other.log" contains debugging information not categorized into one of the debug types listed below as well as the HomeSeer log entries produced during the time when debug logging is enabled. Writing debug information can add to the overall system load, so please use this under advisement from HomeSeer Technologies support personnel.

Parameters

Parameter: mode
Type: long (.NET Integer)
Description: The debug mode for HomeSeer to be set to (see below). Multiple modes can be logically OR'd together to enable several debug modes at once.

Returns

None.

Debug Mode Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Function Trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disable debugging (normal)</td>
</tr>
<tr>
<td>1</td>
<td>Script activity</td>
</tr>
<tr>
<td>2</td>
<td>Event activity</td>
</tr>
<tr>
<td>4</td>
<td>Condition checks</td>
</tr>
<tr>
<td>8</td>
<td>Device value changes</td>
</tr>
<tr>
<td>16</td>
<td>HomeSeer procedure calls</td>
</tr>
<tr>
<td>128</td>
<td>Plug-In Procedure calls</td>
</tr>
<tr>
<td>256</td>
<td>Database Procedure Debugging</td>
</tr>
</tbody>
</table>
Example

Turn on debugging for script, event, and plug-ins: \(1 + 2 + 128 = 131\)

```
Sub Main()
    hs.DebugMode = 131
End Sub
```

See also

- AppStarting
- GetAppPath
- InterfaceVersion
- IsLicensed
- ShuttingDown
- SystemUpTime
- SystemUpTimeTS
- Version

GetAppPath

Purpose

Returns the path to the HomeSeer installation directory. This is useful for finding HomeSeer-specific files like the event log and script-created files.

Parameters

None.

Returns

Return value: **application path**
Type: **string**
Description: The path returned is not terminated by a directory path separator. Use of GetAppPath to construct a valid path will require the addition of "/" before additional path names or files are added.

Example

```
sub main()
    dim s
    s = hs.GetAppPath
    msgbox "The HomeSeer path is: " & s
end sub
```

See also

- AppStarting
- DebugMode
- InterfaceVersion
- IsLicensed
- ShuttingDown
- SystemUpTime
- SystemUpTimeTS
- Version
InterfaceVersion

Purpose
Returns the current version of HomeSeer Plug-In API Interface. This procedure is called by plug-ins to determine the capability level of the interface it is working with.

Parameters
None.

Returns
Return value: version
Type: integer (.NET Short)
Description: For HS2, the returned value is 3, and for HS3, the returned value is 4.

Example
Sub Main(ByVal Parm As Object)
    hs.WriteLog("Info","The API interface version of HomeSeer is " & hs.InterfaceVersion.ToString)
End Sub

See also
AppStarting
DebugMode
GetAppPath
IsLicensed
ShuttingDown
SystemUptime
SystemUpTimeTS
Version

IsLicensed

Purpose
Returns True/False indicating whether HomeSeer has been fully licensed.

Parameters
None.

Returns
Return value: license status
Type: boolean

Example
sub main()
    if hs.IsLicensed then
        hs.WriteLog "Info","HomeSeer is licensed, thank you."
    else
        hs.WriteLog "Info","This copy of HomeSeer is not currently licensed."
    end if
end sub
See also
AppStarting
DebugMode
GetAppPath
InterfaceVersion
ShuttingDown
SystemUptime
SystemUpTimeTS
Version

ShuttingDown

**Purpose**

This allows applications and plug-ins to determine if HomeSeer has started the shutdown process.

**Parameters**

None.

**Returns**

Return value: **Shut Down Status**
Type: Boolean
Description: If TRUE, the system is in the process of shutting down.

See also
AppStarting
DebugMode
GetAppPath
InterfaceVersion
IsLicensed
SystemUptime
SystemUpTimeTS
Version

SystemUptime

**Purpose**

Returns the amount of time HomeSeer has been running. Time is displayed in the format days hours:minutes:seconds.

**Parameters**

None.

**Returns**

Return value: **time**
Type: string

**Example**

' Set a virtual device to display the system uptime
Sub Main(ByVal Parms As Object)
    hs.SetDeviceString(1234, "Uptime: " & hs.SystemUpTime, True)
End Sub

' the display might be: Uptime: 1 Days 12:23:07

See also
   AppStarting  
   DebugMode  
   GetAppPath  
   InterfaceVersion  
   IsLicensed  
   ShuttingDown  
   SystemUpTimeTS  
   Version

SystemUpTimeTS

Purpose
   Returns the amount of time HomeSeer has been running in a TimeSpan structure.

Parameters
   None.

Returns
   Return value: time
   Type: TimeSpan

Example
Sub Main(ByVal Parms As Object)
    Dim TS As TimeSpan
    TS = hs.SystemUpTimeTS
    hs.WriteLog("Up Time", "HomeSeer has been running for " & TS.Days.ToString & " days, " & _
                 TS.Hours.ToString & " hours, and " & TS.Minutes.ToString & " minutes.")
End Sub

See also
   AppStarting  
   DebugMode  
   GetAppPath  
   InterfaceVersion  
   IsLicensed  
   ShuttingDown  
   SystemUpTime  
   Version

Version
**Purpose**

Returns the current version of HomeSeer.

**Parameters**

None.

**Returns**

Return value: version string
Type: string

**Example**

Sub Main(ByValParms As Object)
    hs.WriteLog("Version Info", "HomeSeer HS3 is currently version " & hs.version)
End Sub

See also
AppStarting
DebugMode
GetAppPath
InterfaceVersion
IsLicensed
ShuttingDown
SystemUptime
SystemUpTimeTS

---

**System Functions**

**Purpose**

System functions are used when a function must interface with either the HomeSeer application or the HomeSeer Phone application. Since voice recognition and text-to-speech will use different output and input devices, these functions handle routing the sound information to and from the devices.

When a script calls hs.Speak, the sound is normally sent directly to the sound card on the PC. However, if the script is run in response to a voice command over the phone, it is normally desired to send the sound over the phone's handset. By calling system.Speak, HomeSeer knows where the request originated, and therefore routes the sound out through the proper device.

The system functions listed here are merely wrappers for the real functions in either the hs (HomeSeer object) or hsp (HomeSeer Phone object).

- When using the system functions, note that they only work from within the script that was initially launched. If you call a second script (using hs.Run or hs.RunEx), any system functions used in these scripts will not work on the desired audio channel. This is due to the way HomeSeer creates the system object when the initial script is launched. It has no information about sub scripts as to where the script was launched from. It therefore creates a default system object that assumes the script is dealing with the default audio channel.

- Due to .NET using "system" as a NameSpace, VB.NET scripts must use "hssystem" in place of "system".

**Example**

Speak a phrase

    system.speak text as string, optional wait as boolean

Add a voice command

    system.AddVoiceCommand command as string

Get the last recognized voice command

    system.LastVoiceCommand as string

Clear all voice commands that were set using AddVoiceCommand

    If the script is used over the phone and the "AddVoiceCommand" was used, then this must be called when the script exists so that the standard voice
BackupDB

**Purpose**

This command will close the currently active HomeSeer configuration database and will make a backup copy of it in the Backup directory under the Config directory of HomeSeer.

**Parameters**

None.

**Returns**

Parameter: **result**
Type: **string**
Description: The result of the backup operation. If it is an empty string ("") then the operation was successful. If the result is not empty, then it will contain information regarding the error or problem encountered during the backup procedure.

**Notes / Additional Information**

The default number of backup copies of the database varies with the HomeSeer edition, but the typical amount is 10 for the standard edition of HomeSeer. The backup copies are numbered from 1 to the highest number of backups that are retained. The system overwrites the oldest file when this command is issued. Because the rotation is based upon file date/time, the 10th numbered file (in the case of a 10 limit rotation) is not always the oldest backup. You must examine the file modification dates/times to determine which backup is the most recent or oldest.

You can control the number of copies that are retained by adding the following setting to your settings.ini file in the CONFIG directory:

- **Database**
  - Backup_Copies=15

  In the above example, 15 copies will be retained. The number of copies to be retained must be greater than 0 and less than or equal to 50 or the value will be reset to its default value.

See also
PowerFailRecover

Purpose

This command will trigger a power failure recovery operation similar to the one that is started automatically if it is enabled in the system configuration. The recovery takes place using the number of hours set in the power fail options, and it starts from the time an event was last run successfully on the system.

Parameters

None.

Returns

None.

See also
BackupDB
ScheduleFile
Shutdown
System

ScheduleFile

Purpose

This property can be set and read. Setting this property configures HomeSeer to use a new configuration file. Reading this property reports the currently configured configuration file. Configuration files hold all configured devices and events.

Parameters

Parameter: Filename
Type: String
Description: The complete path to a new configuration file to use, or the filename only if the configuration file is already stored in the HomeSeer\Config directory.

Returns

Return value: Filename
Type: String
Description: Returns the full path and name of the current configuration file.

Example

' Set a new configuration file
hs.ScheduleFile = "c:\NewConfig.hsd"

' Read the current configuration file
Dim Config_File As String
Config_File = hs.ScheduleFile
See also
BackupDB
PowerFailRecover
Shutdown
System

Shutdown

**Purpose**

Causes HomeSeer to shut down immediately. This has the same affect as selecting **File > Exit** from the file menu within HomeSeer. If HomeSeer Phone is running, that will be shut down also.

**Parameters**

None.

**Returns**

None.

See also
BackupDB
PowerFailRecover
ScheduleFile
System

System

**Purpose**

Scripts can access the system object directly without using this function. However, external programs that wish to access the system object need to call this function to get access to it. The system object is an independent interface that allows for access to either the HS (HomeSeer) or HSP (HomeSeer Phone) object.

**Parameters**

None.

**Returns**

Return value: **system object**
Type: **object**
Description: Returns a reference to the system object.

**Example**

```vbscript
dim system
set system = hs.system
```

See also
BackupDB
INI File Editing

In This Section

- ClearINISection
- GetINISection
- GetINISectionEx
- GetINISetting
- SaveINISetting

See also
- System Information
- System Functions
- Plug-Ins
- Logging
- Web Pages
- Callbacks
- Launch
- SendMessage
- ReplaceVariables

---

ClearINISection

**Purpose**

Clears an entire section in an INI file.

- HomeSeer will be reset to its default settings if the “Settings” section is cleared in the `settings.ini` file.

**Parameters**

- **Parameter:** `section`
  - **Type:** string
  - **Description:** Name of the section in the INI to be cleared, such as “Settings” for the HomeSeer settings section.

- **Parameter:** `filename`
  - **Type:** string
  - **Description:** Name of the INI file to be accessed. For instance, the INI file for HomeSeer is `settings.ini`.
    - The file name is relative to the “config” folder in the HomeSeer program directory (`C:\Program Files\HomeSeer 2\Config` by default).

**Returns**

None.

See also
- GetINISection
- GetINISectionEx
- GetINISetting
- SaveINISetting
GetINISection

Purpose

Returns all values in the given INI section. Each entry in the section is separated with a NULL character.

Parameters

Parameter: section
Type: string
Description: Name of the section in the INI file to get, like "Settings" for the HomeSeer settings section.

Parameter: filename
Type: string
Description: File name of the INI file to access, such as "settings.ini". The file name is relative to the "config" folder in the HomeSeer program directory (C:\Program Files\HomeSeer 2\Config by default).

Returns

Return value: ini section
Type: string

Example

Sub Main(ByVal Parms As Object)
  Dim Items() As String
  Dim Section As String

  Section = hs.GetINISection("Settings", "settings.ini")
  Items = Section.Split(Chr(0))
  If Items IsNot Nothing AndAlso Items.Count > 0 Then
    For Each s As String In Items
      If String.IsNullOrEmpty(s) Then Continue For
      hs.WriteLog("Items", s)
    Next
  End If
End Sub

See also
ClearINISection
GetINISectionEx
GetINISetting
SaveINISetting

GetINISectionEx

Purpose

Returns all values in the given INI section. Each entry in the section is an element in a string array.

Parameters

Parameter: section
Type: string
Description: Name of the section in the INI file to get, like "Settings" for the HomeSeer settings section.

Parameter: filename
Type: **string**  
Description: File name of the INI file to access, such as "settings.ini". The file name is relative to the "config" folder in the HomeSeer program directory (C:\Program Files\HomeSeer 2\Config by default).

**Returns**

Return value: **ini section**  
Type: **string array**

**Example**

```vbscript
Sub Main(ByVal Parms As Object)
    Dim Items() As String
    Items = hs.GetINISection("Settings", "settings.ini")
    If Items IsNot Nothing AndAlso Items.Count > 0 Then
        For Each s As String In Items
            If String.IsNullOrEmpty(s) Then Continue For
            hs.WriteLog("Items", s)
        Next
    End If
End Sub
```

**See also**  
ClearINISection  
GetINISection  
GetINISetting  
SaveINISetting

---

**GetINISetting**

**Purpose**

Returns the value associated with the requested key from an INI file.

**Parameters**

Parameter: **section**  
Type: **string**  
Description: Name of section in INI file to get, such as "Settings" for the HomeSeer settings section.

Parameter: **key**  
Type: **string**  
Description: Name of the key in the INI file to access.

Parameter: **default**  
Type: **string**  
Description: The default value to return if the key is not found.

Parameter: **filename** (optional)  
Type: **string**  
Description: The file name of the INI file to access. The file name is relative to the "config" folder in the HomeSeer program directory (C:\Program Files\HomeSeer 2\Config by default). If this parameter is omitted, the HomeSeer settings.ini file is used.

**Returns**

Return value: **ini key value**  
Type: **string**

**Example**

```vbscript
Sub Main(ByVal Parms As Object)
    Dim ConfigFileName As String = ""
```

---

HomeSeer HS3 - End User Documentation
* Get the name of the current HomeSeer configuration file.
ConfigFileName = hs.GetINISetting("Settings", "configfile", "")
hs.WriteLog("Config", "The current HomeSeer configuration file is " & ConfigFileName)

End Sub

See also
ClearINISection
GetINISection
GetINISectionEx
SaveINISetting

SavelNISetting

Purpose
Saves a key/value pair in an INI file.

Parameters
Parameter: section
Type: string
Description: Name of the section in the INI file to save to, like "Settings" for the HomeSeer settings section.

Parameter: key
Type: string
Description: Name of the key in the INI file to access.

Parameter: value
Type: variant
Description: The value to save in the given key.

Parameter: filename
Type: string (optional)
Description: This is the file name of the INI file to access. The file name is relative to the "config" folder in the HomeSeer program directory (C:\Program Files\HomeSeer 2\Config by default). If this parameter is omitted, the HomeSeer settings.ini file is used.

Returns
None.

Example
hs.SaveINISetting("My Settings", "Zip Code", "49601", "My_App_Settings.ini")

See also
ClearINISection
GetINISection
GetINISectionEx
GetINISetting

Plug-Ins
GetHSPRef

Purpose

This returns a reference to the HomeSeer Phone interface. This function is used primarily for plug-ins to communicate with the HomeSeer Phone directly without having to build a reference to it that might cause HomeSeer Phone to run when the user did not want it to run.

Not available on systems running Linux such as the Zee controller or the Linux version of HomeSeer.

Parameters

None.

Returns

Return value: object reference
Type: object

PluginFunction

Purpose

Call a function in a plugin. This can only be used to call functions and subs. If you need to access a property, use PluginPropertyGet.

Parameters
Parameter: **Plugin Name**  
Type: **String**  
Description: The name of the plugin to access. The plugin must be enabled and running.

Parameter: **Instance Name**  
Type: **String**  
Description: The instance name of the plugin to access if this is a multi-instance plugin. For single instance plugins, pass an empty string.

Parameter: **Function Name**  
Type: **String**  
Description: The name of the function to call. The name is case sensitive.

Parameter: **Parameters**  
Type: **Array**  
Description: An array of parameters to pass to the function, if the function accepts parameters. If the function does not take any parameters, pass "Nothing".

**Returns**

Return value: **Result**  
Type: **varies**  
Description: The return value from the function call.

**Example**

The following .vb script will access the AccessLevel function in the Z-Wave plugin.

```vbnet
Sub Main(parm as object)
    Dim level as integer
    level = hs.PluginFunction("Z-Wave", "", "AccessLevel", nothing)
    hs.writelog("Plugin name", level.ToString)
End Sub
```

See also
- GetHSPRef
- PluginPropertyGet
- PluginPropertySet
- GetPluginsList
- RegisterLinkEx

---

**PluginPropertyGet**

**Purpose**

Access a property in a plugin. The plugin must be enabled and running. If you need to access a function or sub, use `PluginFunction`.

**Parameters**

Parameter: **Plugin Name**  
Type: **String**  
Description: The name of the plugin to access. The plugin must be enabled and running.

Parameter: **Instance Name**  
Type: **String**  
Description: The instance name of the plugin to access if this is a multi-instance plugin. For single instance plugins, pass an empty string.

Parameter: **Property Name**  
Type: **String**  
Description: The name of the function to call. The name is case sensitive.

Parameter: **Parameters**  
Type: **Array**  
Description: An array of parameters to pass to the function, if the function accepts parameters. If the function does not take any parameters, pass "Nothing".

**Returns**
Return value: **Result**  
Type: **varies**  
Description: The return value from the function call.

**Example**

The following .vb script will access the name property in the Z-Wave plugin.

```vbnet
Sub Main(parm as object)
    Dim name as String
    name = hs.PluginPropertyGet("Z-Wave", ",", "Name", nothing)
    hs.writelog("Plugin name", name)
End Sub
```

See also
- GetHSPRef
- PluginFunction
- PluginPropertySet
- GetPluginsList
- RegisterLinkEx

---

**PluginPropertySet**

**Purpose**

Set a property in a plugin. The plugin must be enabled and running. If you need to access a function or sub, use [PluginFunction](#).

**Parameters**

Parameter: **Plugin Name**  
Type: **String**  
Description: The name of the plugin to access. The plugin must be enabled and running.

Parameter: **Instance Name**  
Type: **String**  
Description: The instance name of the plugin to access if this is a multi-instance plugin. For single instance plugins, pass an empty string.

Parameter: **Property Name**  
Type: **String**  
Description: The name of the function to call. The name is case sensitive.

Parameter: **Value**  
Type: **Object**  
Description: The value to set to the property.

**Returns**

Return value: **Does not return a value, this is a Sub**

**Example**

The following .vb script will set a property named "ComPort" in the plugin "Acme Widget".

```vbnet
Sub Main(parm as object)
    hs.PluginPropertySet("Acme Widget", ",", "ComPort", 2)
End Sub
```

See also
GetPluginsList

**Purpose**

Returns an array of all plug-in names for plug-ins that are enabled.

**Parameters**

None.

**Returns**

Return value: *plug-in array*

Type: *Array of String*

**Note**

If a plug-in has instance information, then the instance name is appended to the plug-in name, with a colon (:) separating them. It should therefore be noted that this procedure may return the same plug-in name multiple times if there are multiple instances of that plug-in loaded.

**Example**

```vba
Sub Main(ByValParms As Object)
    Dim List() As String

    List = hs.GetPluginsList
    If List IsNot Nothing AndAlso List.Count > 0 Then
        For Each P As String In List
            If String.IsNullOrEmpty(P) Then Continue For
            hs.WriteLog("Plug-In List", P & " is currently enabled.")
        Next
    End If

End Sub
```

**See also**

GetHSPRef
PluginFunction
PluginPropertyGet
PluginPropertySet
RegisterLinkEx

RegisterLinkEx

**Purpose**

This procedure registers a web page link with HomeSeer that is handled by a plug-in.
This procedure only works with a static object reference such as a plug-in - it cannot be used by scripts.

Parameters

Parameter: object ref
Type: code object
Description: The object reference provided here is a code object such as a form or class, that contains the BuildPage, PagePut, etc. procedures that HomeSeer will call to provide the web page functionality.

Parameter: plug-in name
Type: string
Description: Name of the plug-in or program that this link is associated with. HomeSeer uses this to keep the links updated in the event that the plug-in or program is removed or otherwise goes away.

Returns

None.

Example

hs.RegisterLinkEx FrmWebPage, IFACE_NAME

See also

GetHSPRef
PluginFunction
PluginPropertyGet
PluginPropertySet
GetPluginsList

ClearLog

Purpose

See also

System Information
System Functions
INI File Editing
Plug-Ins
Web Pages
Callbacks
Launch
SendMessage
ReplaceVariables
Clears the event log in memory. The event log file specified in the General Setup screen will not be touched.

**Parameters**

None.

**Returns**

None.

**Example**

`hs.ClearLog`

See also

GetLog  
LogGet  
NoLog  
WriteLog  
WriteLogEx  
WriteLogDetail

---

**In This Section**

LogEntry Structure  
GetLog_FullFilter  
GetLog_Date  
GetLog_Date_Text  
GetLog_Date_Priority  
GetLog_Date_ErrorCode

---

See also

ClearLog  
LogGet  
NoLog  
WriteLog  
WriteLogEx  
WriteLogDetail

---

**LogEntry Structure**

This structure is used by the GetLog_ functions listed in this section, and is as follows:

```plaintext
Public Structure LogEntry
    Public LogTime As Date ' The date and time the log entry was recorded.
    Public LogType As String ' The 'type' string as logged.
    Public LogText As String ' The main message text of the log entry.
    Public LogStyleColor As String ' The color code string associated with this log entry.
    Public LogPriority As Integer ' The priority (0 = None Specified).
    Public LogFrom As String ' Information on the software that generated the log entry.
    Public LogErrorCode As Integer ' An error code as provided by the originator of the log entry.
    Public LogLength As Integer ' The length of the full text of the log entry.
End Structure
```
GetLog_FullFilter

Function GetLog_FullFilter(StartDate As Date, EndDate As Date, mType As String, mEntry As String, mEntry_RegEx As Boolean, Pri_Start As Integer, Pri_End As Integer, Show_NoPri As Boolean, ErrorCode As Integer, ShowAllErrorCode As Boolean) As LogEntry

Purpose

This function retrieves entries from the system log database and returns them as an array of LogEntry structures. This function provides the most filter parameters possible for selection of log entries. Use this function when you need a very precise set of log entries.

Parameters

Parameter: StartDate
Type: Date/Time
Description: This is the starting date for the log entries that you want retrieved. If you do not care about a starting date, use Date.MinValue

Parameter: EndDate
Type: Date/Time
Description: This is the ending (latest) date for the log entries that you want retrieved. If you do not care about an ending date, use Date.MaxValue

Parameter: mType
Type: String
Description: This is the log entry type, such as "Info" or "Error" - these must EXACTLY match what you are searching for - leave empty to not use this.

Parameter: mEntry
Type: String
Description: This is the entry text to find. This is an exact match field unless wildcards or RegEx (next parameter) is used. To use a wildcard, use the percent (%) character. For example, to match everything that starts with "Super", use "Super%" which will match SuperDuper, Super Cool, and Super Delicious.

Parameter: mEntry_RegEx
Type: Boolean
Description: When this parameter is TRUE, the previous parameter (mEntry) contains a Regular Expression to be run on the log message field retrieved from the database. For help with Regular Expressions, see Regular-Expressions.info

Parameter: Pri_Start
Type: Integer
Description: This is the starting priority for log entries to be retrieved. If you wish to retrieve log entries which are priority 1 through 5, provide a value of 1 here and a value of 5 in the Pri_End parameter. If you do not need to filter by priority, use the value -1.

Parameter: Pri_End
Type: Integer
Description: This is the ending priority value for the log entries to be retrieved. If you wish to retrieve log entries which are priority 1 through 5, provide a value of 1 in the Pri_Start parameter and a value of 5 in this parameter. If you do not need to filter by priority, use the value -1.

Parameter: Show_NoPri
Type: Boolean
Description: When set to True, unprioritized entries (Priority = 0) are included in the selection in addition to the priorities selected with Pri_Start and Pri_End.

Parameter: ErrorCode
Type: Integer
Description: This is the error code to select records with. Use a value of -1 (Or use ShowAllErrorCode) if you do not care to filter log entries using the Error Code value.

Parameter: ShowAllErrorCode
Type: Boolean
Description: When set to True, the ErrorCode parameter is ignored and all log entries that match the other filters are returned.
Returns

Return value: LogEntry
Type: Array of Structure LogEntry
Description: See LogEntry Structure.

Example

This example retrieves all log entries from a week ago to today:

```vbs
Dim Logs() As HomeSeerAPI.LogEntry
Logs = hs.GetLog_FullFilter(Now.AddDays(-7), Now, "Error", "", False, 1, 3, True, -1, True)
If Logs IsNot Nothing AndAlso Logs.Count > 0 Then
    hs.WriteLog("Info", Logs.Count.ToString & " log entries retrieved, and this just added another!")
End If
```

See also

LogEntry Structure
GetLog_Date
GetLog_Date_Text
GetLog_Date_Priority
GetLog_Date_ErrorCode

GetLog_Date

Function GetLog_Date(ByVal StartDate As Date, ByVal EndDate As Date) As LogEntry()

Purpose

This function retrieves entries from the system log database and returns them as an array of LogEntry structures. Use this function when you need to only filter log entries by date.

Parameters

Parameter: StartDate
Type: Date/Time
Description: This is the starting date for the log entries that you want retrieved. If you do not care about a starting date, use Date.MinValue

Parameter: EndDate
Type: Date/Time
Description: This is the ending (latest) date for the log entries that you want retrieved. If you do not care about an ending date, use Date.MaxValue

Returns

Return value: LogEntry
Type: Array of Structure LogEntry
Description: See LogEntry Structure.

Example

This example retrieves all log entries from a week ago to today:

```vbs
Dim Logs() As HomeSeerAPI.LogEntry
Logs = hs.GetLog_Date(Now.AddDays(-7), Now)
If Logs IsNot Nothing AndAlso Logs.Count > 0 Then
    hs.WriteLog("Info", Logs.Count.ToString & " log entries retrieved, and this just added another!")
End If
```
GetLog_Date_Text

Function GetLog_Date_Text(StartDate As Date, EndDate As Date, _
ByVal mType As String, ByVal mEntry As String, ByVal mEntry_RegEx As Boolean) _
As LogEntry()

Purpose

This function retrieves entries from the system log database and returns them as an array of LogEntry structures. This function provides the ability to filter on date and the log entry text fields of type and message. Use this function when you are looking for specific log entries without knowing the priority or error codes.

Parameters

Parameter: StartDate
Type: Date/Time
Description: This is the starting date for the log entries that you want retrieved. If you do not care about a starting date, use Date.MinValue

Parameter: EndDate
Type: Date/Time
Description: This is the ending (latest) date for the log entries that you want retrieved. If you do not care about an ending date, use Date.MaxValue

Parameter: mType
Type: String
Description: This is the log entry type, such as "Info" or "Error" - these must EXACTLY match what you are searching for - leave empty to not use this.

Parameter: mEntry
Type: String
Description: This is the entry text to find. This is an exact match field unless wildcards or RegEx (next parameter) is used. To use a wildcard, use the percent (%) character. For example, to match everything that starts with "Super", use "Super%" which will match SuperDuper, Super Cool, and Super Delicious.

Parameter: mEntry_RegEx
Type: Boolean
Description: When this parameter is TRUE, the previous parameter (mEntry) contains a Regular Expression to be run on the log message field retrieved from the database. For help with Regular Expressions, see Regular-Expressions.info

Returns

Return value: LogEntry
Type: Array of Structure LogEntry
Description: See LogEntry Structure.

Example

This example retrieves all log entries from a week ago to today, with a type of "Error":

Dim Logs() As HomeSeerAPI.LogEntry
Logs = hs.GetLog_Date_Text(Now.AddDays(-7), Now, "Error", "", False)
If Logs IsNot Nothing AndAlso Logs.Count > 0 Then
    hs.WriteLog("Info", Logs.Count.ToString & " log entries retrieved, and this just added another!")
End If

See also
LogEntry Structure
GetLog_Date_Priority

Function GetLog_Date_Priority(ByVal StartDate As Date, ByVal EndDate As Date, _
ByVal Pri_Start As Integer, ByVal Pri_End As Integer, ByVal Show_NoPri As Boolean) As LogEntry()

Purpose

This function retrieves entries from the system log database and returns them as an array of LogEntry structures. This function provides filtering the entries using the date and priority.

Parameters

Parameter: StartDate  
Type: Date/Time  
Description: This is the starting date for the log entries that you want retrieved. If you do not care about a starting date, use Date.MinValue

Parameter: EndDate  
Type: Date/Time  
Description: This is the ending (latest) date for the log entries that you want retrieved. If you do not care about an ending date, use Date.MaxValue

Parameter: Pri_Start  
Type: Integer  
Description: This is the starting priority for log entries to be retrieved. If you wish to retrieve log entries which are priority 1 through 5, provide a value of 1 here and a value of 5 in the Pri_End parameter. If you do not need to filter by priority, use the value -1.

Parameter: Pri_End  
Type: Integer  
Description: This is the ending priority value for the log entries to be retrieved. If you wish to retrieve log entries which are priority 1 through 5, provide a value of 1 in the Pri_Start parameter and a value of 5 in this parameter. If you do not need to filter by priority, use the value -1.

Parameter: Show_NoPri  
Type: Boolean  
Description: When set to True, unprioritized entries (Priority = 0) are included in the selection in addition to the priorities selected with Pri_Start and Pri_End.

Returns

Return value: LogEntry  
Type: Array of Structure LogEntry  
Description: See LogEntry Structure.

Example

This example retrieves all log entries from a week ago to today, with a priority between 1 and 3 inclusive or unprioritized (Priority = 0):

Dim Logs() As HomeSeerAPI.LogEntry
Logs = hs.GetLog_Date_Priority(Now.AddDays(-7), Now, 1, 3, True)
If Logs IsNot Nothing AndAlso Logs.Count > 0 Then  
    hs.WriteLog("Info", Logs.Count.ToString & " log entries retrieved, and this just added another!")
End If

See also
LogEntry Structure  
GetLog_FullFilter  
GetLog_Date  
GetLog_Date_Text  
GetLog_Date_ErrorCode
GetLog_Date_ErrorCode

**Function**

```vba
Function GetLog_Date_ErrorCode(ByVal StartDate As Date, ByVal EndDate As Date, ByVal ErrorCode As Integer, ByVal ShowAllErrorCode As Boolean) As LogEntry()
```

**Purpose**

This function retrieves entries from the system log database and returns them as an array of LogEntry structures. This function provides filtering of the log entries by date, and the error code associated with the log entry.

**Parameters**

- **Parameter:** `StartDate`  
  **Type:** Date/Time  
  **Description:** This is the starting date for the log entries that you want retrieved. If you do not care about a starting date, use Date.MinValue

- **Parameter:** `EndDate`  
  **Type:** Date/Time  
  **Description:** This is the ending (latest) date for the log entries that you want retrieved. If you do not care about an ending date, use Date.MaxValue

- **Parameter:** `ErrorCode`  
  **Type:** Integer  
  **Description:** This is the error code to select records with. Use a value of -1 (Or use ShowAllErrorCode) if you do not care to filter log entries using the Error Code value. Error codes are provided by the procedure that added the log entry - they are not standardized. Consult the author of 3rd party provided scripts and plug-ins to obtain a list of error codes that they may have used.

- **Parameter:** `ShowAllErrorCode`  
  **Type:** Boolean  
  **Description:** When set to True, the ErrorCode parameter is ignored and all log entries that match the other filters are returned.

**Returns**

- **Return value:** LogEntry  
  **Type:** Array of Structure LogEntry  
  **Description:** See LogEntry Structure.

**Example**

This example retrieves all log entries from a week ago to today, with an error code of 4166:

```vba
Dim Logs() As HomeSeerAPI.LogEntry
Logs = hs.GetLog_Date_ErrorCode(Now.AddDays(-7), Now, 4166, False)
If Logs IsNot Nothing AndAlso Logs.Count > 0 Then
    hs.WriteLog("Info", Logs.Count.ToString & " log entries retrieved, and this just added another!")
End If
```

**See also**

- LogEntry Structure
- GetLog_FullFilter
- GetLog_Date
- GetLog_Date_Text
- GetLog_Date_Priority
Purpose

Retrieves the current HomeSeer log buffer contents.

Parameters

Parameters: none

Returns

Return value: buffer
Type: string
Description: The contents of the HomeSeer log buffer.

The HomeSeer log is written to the HomeSeer log file (typically HomeSeer.log) and is stored in memory, up to the limit the user has set on the "General" tab of the HomeSeer configuration pages. The buffer returned here contains the log entries up to that limit, or since the log buffer in memory was last cleared by the user or a script action.

See also
ClearLog
GetLog
NoLog
WriteLog
WriteLogEx
WriteLogDetail

NoLog

Property NoLog() As Boolean

Purpose

This property allows you to get or set whether logging is to take place on the process this property is set from. If you set this property in a script, then all logging entries from procedures run from the script are stopped. If this property is set from a plug-in, then logging from that plug-in is prevented until NoLog is reset or the plug-in is shut-down and restarted.

Note: Log entries (e.g. WriteLogDetail) which include a Priority value of 1 (highest) are always written regardless of the NoLog setting.

Parameters

Parameter: NoLog
Type: Boolean
Description: When set to True, logging for the process thread is turned off.

Returns

Return value: NoLog
Type: Boolean
Description: Provides the current NoLog setting for the process the property is retrieved from.

Example

To disable logging for the current process thread (script, plug-in, event):

hs.NoLog = True

See also
ClearLog
GetLog
LogGet
WriteLog
WriteLogEx
WriteLogDetail
WriteLog

Purpose

Writes a message to the event log.

Parameters

Parameter: type
Type: string
Description: This is a string that defines the type of event like "Error" or "Info". It can be anything you like. Common message types are "Info", "Warning", and "Error".

Parameter: message
Type: string
Description: This is the text to be displayed in the log, like a descriptive error message.

Returns

None.

Example

Sub Main()
    hs.WriteLog "Error", "An error has occurred in my script!"
End Sub

See also
ClearLog
GetLog
LogGet
NoLog
WriteLogEx
WriteLogDetail

WriteLogEx

Purpose

Writes a message to the event log, with an optional COLOR specified.

Parameters

Parameter: type
Type: string
Description: This is a string that defines the type of event like "Error" or "Info". It can be anything you like. Common message types are "Info", "Warning", and "Error".

Parameter: message
Type: string
Description: This is the text to be displayed in the log, like a descriptive error message.

Optional Parameter: color
Type: string
Description: This is the color code that you want associated with the log entry when you view it in the web browser. The color code must be in the #XXXXXX format, which is the # symbol followed by hexadecimal values for Red, Green, and Blue. Here are some examples of colors and their color values:
NOTE: At this time, the HomeSeer colors for Error, Warning, and Updater log entries of Red, Orange, and Green respectively are automatic and are still set by HomeSeer.

Returns
None.

Example

Sub Main()
    hs.WriteLogEx "Error", "An error has occurred in my script!"
    hs.WriteLogEx "Hello", "I much prefer to see this in Navy Blue!", "#000080"
End Sub

See also
ClearLog
GetLog
LogGet
NoLog
WriteLog
WriteLogDetail

WriteLogDetail

Purpose
This procedure writes an entry to the HomeSeer log with additional detailed information which can be used by plug-ins and on the log screen to highlight specific log events.

Parameters

Parameter: **mType**
Type: **String**
Description: This is the log entry "type", which is first to appear in the log. Usually this is used to indicate a severity or what the log entry pertains to, such as "Error" or "My Script".

Parameter: **Message**
Type: **String**
Description: This is the main log message.

Parameter: **Color**
Type: **String**
Description: This is the color code that you want associated with the log entry when you view it in the web browser. The color code must be in the #XXXXXX format, which is the # symbol followed by hexadecimal values for Red, Green, and Blue. Here are some examples of colors and their color values:

WHITE = "#FFFFFF"
RED = "#FF0000"
BLACK = "#000000"
NAVY = "#000080"
LIGHT BLUE = "#D9F2FF"
LIGHT GRAY = "#E1E1E1"
PINK = "#FFB6C1"
ORANGE = "#D58000"
GREEN = "#008000"
NOTE: At this time, the HomeSeer colors for Error, Warning, and Updater log entries of Red, Orange, and Green respectively are automatic and are still set by HomeSeer.

Parameter: **Priority**
Type: **Integer**
Description: This is an indicator of the priority of the log entry, with the value 0 being unspecified, and the value 1 being the highest priority. Even if a process (e.g. Event) has logging turned off, priority 1 log entries are still written to the log.

Parameter: **mFrom**
Type: **String**
Description: This indicates the source of the message. For example: "Cool_Plugin, Main Procedure, Update Section"

Parameter: **ErrorCode**
Type: **Integer**
Description: This is an error code number which is meaningful only to the script or plug-in that generated this log entry.

**Returns**

Return value: None.

**Example**

```javascript
hs.WriteLogDetail("Error", "Oh No, Mr. Bill!", COLOR_RED, 1, "SaturdayNight Plugin", 911)
```

**See also**

ClearLog
GetLog
LogGet
NoLog
WriteLog
WriteLogEx
GetPageFooter

Purpose

Returns the HomeSeer generated page footer for use in creating your own web pages.

Parameters

Optional Parameter: **NoEndTags**
- Type: **Boolean**
- Description: If set to TRUE, the html ending tags /BODY and /HTML will be omitted from the output. The default value if this parameter is not provided is FALSE.

Returns

The output is a string of HTML that comprises the HomeSeer generated page ending (footer) for web pages. The output consists of these elements:

- The navigation links, if the configuration is set to display them at the bottom of the web page, enclosed in a "navbottom" SPAN tag.
- The contents of the tail.htm file, enclosed in a "tailfile" SPAN tag.
- The </body> and </html> closing tags, unless the "NoEndTags" parameter is TRUE.

See also
- GetPageHeader
- WebValidateUser
- WebStatsPageViews
- WebServerSSLPort
- WebServerPort
- WebLoggedInUser
- GetUsers
- GetPlugLinks
- RegisterHelpLink
- RegisterLinkEx
- UnRegisterHelpLinks

GetPageHeader

Purpose

This procedure, useful when creating your own Active Server web Pages (ASPs) allows you to use HomeSeer's page header generating function with parameters that allow you to get all or part of the header information.

- As this procedure has options for returning the header with the HTML, HEAD, and BODY tags, you should make sure your use of this procedure does not generate duplicates of the above tags in your resulting page.

Parameters

Parameter: **title**
- Type: **string**
- Description: The title for the web page to be displayed at the top of the page before the logo bar.

Parameter: **extra_meta**
- Type: **string**
- Description: This parameter allows you to specify additional HTML to be included in the HEAD section of the page, and it should be formatted as a complete <meta ...> tag.

Parameter: **HSOnload**
- Type: **string**
- Description: This parameter is used by HomeSeer to specify Body_OnLoad procedures in the resulting page. They are prepended to the "Web Site" configuration item of the same name.

Parameter: **ExcludeNavLinks**
- Type: **Boolean**
- Description: If TRUE is passed to this parameter, the navigation links will not be included in the output.
**Parameter:** NoHeader  
**Type:** Boolean  
**Description:** If TRUE is passed to this parameter, the HTML tag and all contents of the HEAD section will be excluded from the output. The HEAD section includes the META tags, the page title, and the BODY tag (and thus Body_OnLoad is excluded).

**Optional Parameter:** HeadContentOnly  
**Type:** Boolean  
**Description:** If set to TRUE, only the contents of the HEAD html tag will be returned - use this if you are generating the other page elements yourself. See the "NoHeader" parameter for information on what is included in the HEAD tag section. The default value if this parameter is not specified is FALSE.

**Optional Parameter:** BodyContentOnly  
**Type:** Boolean  
**Description:** If set to TRUE, only the contents of the BODY html tag will be returned - this includes any Body_OnLoad specifications that are passed with the "HSOnload" parameter or the user's Body_OnLoad configuration value. This is useful when generating your own web pages but wish to maintain the user's Body_OnLoad options which may be used with other plug-ins in the system. The BODY tag is included in the output. The default value if this parameter is not specified is FALSE.

**Optional Parameter:** BodyOnLoadOnly  
**Type:** Boolean  
**Description:** If set to TRUE, only the contents of the "HSOnload" parameter and the user's Body_OnLoad configuration value. See "BodyContentOnly" for a usage scenario. The BODY html tag is not included in the output. The default value if this parameter is not specified is FALSE.

**Returns**

A string value containing the HTML content specified through the parameter choices.

A summary of a complete HomeSeer generated page header are as follows:

- HTML Tag
- HEAD Tag
- HomeSeer expiration and cache META tags, and any user specified HTML from the file "Web Site" configuration or META.HTM file.
- The TITLE tag and the title of the page.
- The BODY tag and any additional Body_OnLoad procedure specifications from the "Web Site" configuration.
- The contents of the HEAD.HTM file if it exists.
- The HomeSeer logo table area, enclosed in a "logotable" SPAN tag, and including these elements:
  - The page title portion of the logo table area, enclosed in a "pgtitle" SPAN tag.
  - The clock portion of the logo table area, enclosed in a "clock" SPAN tag, and an empty "userclock" SPAN tag for use in replacing the HomeSeer clock with a user generated version.
  - The sunrise portion of the logo table area, enclosed in a "lbsunrise" SPAN tag.
  - The sunset portion of the logo table area, enclosed in a "lbsunset" SPAN tag.
  - The logged on user portion of the logo table area enclosed in a "lbuser" SPAN tag.
- The navigation links, if specified to be included in the top of the page, enclosed in a "navtop" SPAN tag.

See also  
GetPageFooter  
WebValidateUser  
WebStatsPageViews  
WebServerSSLPort  
WebServerPort  
WebLoggedInUser  
GetUsers  
GetPlugLinks  
RegisterHelpLink  
RegisterLinkEx  
UnRegisterHelpLinks
Purpose

Returns TRUE if the given username/password pair is valid for the web server. Useful if you create your own login ASP web page.

Parameters

- Parameter: **username**
  - Type: string
  - Description: Name of the user to validate.

- Parameter: **password**
  - Type: string
  - Description: Password of the user to validate.

Returns

- Return value: **user authorization**
  - Type: boolean

See also

- GetPageFooter
- GetPageHeader
- WebStatsPageViews
- WebServerSSLPort
- WebServerPort
- WebLoggedInUser
- GetUsers
- GetPlugLinks
- RegisterHelpLink
- RegisterLinkEx
- UnRegisterHelpLinks

WebStatsPageViews

Purpose

This is a read/write property. It will return the number of times your web site has displayed a complete page. To reset the statistics, set this property to 0.

Parameters

- Parameter: **value**
  - Type: string
  - Note: Set to 0 when clearing the stats.

Returns

- Return value: **page statistics**
  - Type: integer
  - Description: The number of page views from the HomeSeer web site as an integer.

Example

```vbscript
' get the page view stats and set to a virtual device for display
sub main()
    dim s
    s = hs.WEBStatsPageViews
    hs.SetDeviceString "z1","Page Views: ",cstr(s)
end sub

' reset the stats
```
sub main()
    hs.WEBStatsPageViews = 0
end sub

See also
GetPageFooter
GetPageHeader
WebValidateUser
WebServerSSLPort
WebServerPort
WebLoggedinUser
GetUsers
GetPlugLinks
RegisterHelpLink
RegisterLinkEx
UnRegisterHelpLinks

WebServerSSLPort

Purpose

Returns the port number of the HomeSeer SSL (Secure Socket Layer) Web Server if enabled.

Parameters

None.

Returns

Return value: port
Type: integer
Description: The port number if the SSL server is enabled, otherwise 0.

See also
GetPageFooter
GetPageHeader
WebValidateUser
WebServerSSLPort
WebServerPort
WebLoggedinUser
GetUsers
GetPlugLinks
RegisterHelpLink
RegisterLinkEx
UnRegisterHelpLinks

WebServerPort

Purpose
Provides the port number of the HomeSeer web server.

Parameters

None.

Returns

Return value: **port**
Type: integer
Description: The port number.

See also
GetPageFooter
GetPageHeader
WebValidateUser
WebStatsPageViews
WebServerSSLPort
WebLoggedInUser
GetUsers
GetPlugLinks
RegisterHelpLink
RegisterLinkEx
UnRegisterHelpLinks

WebLoggedInUser

Purpose

Returns the user ID of the person who is logged into the web server. This is useful for scripts that you may not want to run if a guest is logged in.

Parameters

None.

Returns

Return value: **current user**
Type: string

See also
GetPageFooter
GetPageHeader
WebValidateUser
WebStatsPageViews
WebServerSSLPort
WebServerPort
GetUsers
GetPlugLinks
RegisterHelpLink
RegisterLinkEx
UnRegisterHelpLinks

GetUsers

Purpose
Returns a list of web users and their access rights (but not passwords) from the system. The list returned is in the form:

```
username|rights,username2|rights2, etc.
```

User rights are values that are sometimes OR'd together and are as follows:

```
USER_GUEST = 1
USER_ADMIN = 2
USER_LOCAL = 4
USER_NORMAL = 8
```

A user is either GUEST, NORMAL, or ADMIN, but any of the user IDs with that access level can also be the LOCAL user, which is the username used when the system is accessed via the local network. For example, a user with rights that are equal to 10 (decimal) is the local user, and that user has admin rights.

- This function returns an empty string if the command has not been permitted in the web server configuration options, Security setting.

**Parameters**

None.

**Returns**

Return value: user list
Type: string

**Example**

This script will produce a list of the users and their rights to the system log.

```vbs
Sub main()
    Dim sAllUsers
    Dim sUserPairs
    Dim i
    Dim iTemp
    Dim sUser
    Dim iRights
    Dim sRights
    Dim bNoRights
    CONST USER_GUEST = 1
    CONST USER_ADMIN = 2
    CONST USER_LOCAL = 4

    sAllUsers = hs.GetUsers
    sUserPairs = Split(sAllUsers, ",")
    ' Now sUserPairs is an array of username.rights pairs.

    For i = 0 To UBound(sUserPairs)
        iTemp = sUserPairs(i)
        sUser = Left(iTemp.Instr(sTemp, ",") + 1)
        iRights = CInt(trim(iTemp.Instr(sTemp, ",") + 1))
        sRights = ""
        bNoRights = False
        If (iRights And USER_GUEST) = USER_GUEST Then
            sRights = sRights & "Guest"
        End If
        If (iRights And USER_ADMIN) = USER_ADMIN Then
            sRights = sRights & "Admin"
        End If
        If Len(sRights) = 0 Then
            bNoRights = True
        End If
        If (iRights And USER_LOCAL) = USER_LOCAL Then
            If Len(sRights) > 0 Then
                sRights = sRights & ", and this is the Local Login ID"
            Else
                sRights = "Local Login ID"
            End If
        End If
        If bNoRights Then
            hs.WriteLog "User Info","Name is:" & sUser & ", and has no user rights." & sRights
        Else
            hs.WriteLog "User Info","Name is:" & sUser & ", and the rights are:" & sRights
        End If
    Next
End Sub
```
Example output from this script:
4/1/2004 12:00:00 AM~!~Event Trigger~!~Trigger from menu (GetUsers Test)
4/1/2004 12:00:00 AM~!~User Info~!~Name is: guest and has no user rights.
4/1/2004 12:00:00 AM~!~User Info~!~Name is: Mary and has no user rights. Local Login ID
4/1/2004 12:00:00 AM~!~User Info~!~Name is: Charlie and the rights are: Admin

See also
GetPageFooter
GetPageHeader
WebValidateUser
WebStatsPageViews
WebServerSSLPort
WebServerPort
WebLoggedInUser
GetPlugLinks
RegisterHelpLink
RegisterLinkEx
UnRegisterHelpLinks

GetPlugLinks

Purpose
Returns a list of plug-in web page titles and link locations, separated by chr(1) and chr(2) characters.

Parameters
None.

Returns
Return value: plug-in link pages information
Type: string

Example
sub main()
    dim pname
    dim plink
    dim xTemp
    sTemp = hs.GetPlugLinks
    plink = split(sTemp.chr(2))
    for x = 0 to ubound(plink)
        pname = split(plink(x).chr(1))
        hs.writelog "GetPlugLinks"."Link titled " & pname(0) & " is at " & pname(1)
    next
end sub

See also
GetPageFooter
GetPageHeader
WebValidateUser
WebStatsPageViews
WebServerSSLPort
WebServerPort
WebLoggedInUser
GetUsers
RegisterHelpLink
RegisterLinkEx
UnRegisterHelpLinks
RegisterHelpLink

**Sub** RegisterHelpLink(**ByVal** cbo **As** WebPageDesc)

**Purpose**
This call registers a help link resource with HomeSeer so that it will appear on the help (/help) page of HomeSeer. This function can be used by both plug-ins and scripts.

**Parameters**
- **Parameter:** cbo
  - **Type:** Call-Back Object
  - **Description:** This class object contains several parameters used by web page link definitions. See WebPageDesc for more information on the properties within this object.

**Returns**
None.

**Example**

```vbscript
Sub Main(**ByVal** Parms **As** Object)
  Dim cbo **As** New WebPageDesc
  cbo.plugInName = "UserScript1"
  cbo.plugInInstance = "Help Page Link"
  cbo.link = "MyScript\Help\MyHelpFile.pdf"
  cbo.linktext = "Utility Script System Help"
  cbo.page_title = "Utility System Help Page"
  hs.RegisterLinkEx(cbo)
End Sub
```

See also
GetPageFooter
GetPageHeader
WebValidateUser
WebStatsPageViews
WebServerSSLPort
WebServerPort
WebLoggedInUser
GetUsers
GetPlugLinks
RegisterLinkEx
UnRegisterHelpLinks

WebPageDesc Object

This class object is used by several functions which register web pages or links to web pages in the HomeSeer web server or web page menus.

```vbscript
Public Class WebPageDesc
  Public plugInName **As** String = "" ' When used by a script call to register a non-plugin link,
    ' this is a name to be associated with the link so that
    ' links can be grouped and for the removal of links done
    ' via other calls.
  Public plugInInstance **As** String = "" ' When used by a script call to register a non-plugin link,
```

HomeSeer HS3 - End User Documentation
RegisterLinkEx

Sub RegisterLinkEx(ByVal cbo As WebPageDesc)

Purpose

This call registers a link resource with HomeSeer so that it will appear on the menu bar of HomeSeer. This function can be used by both plug-ins and scripts.

Parameters

Parameter: cbo
Type: Call-Back Object
Description: This class object contains several parameters used by web page link definitions. See WebPageDesc for more information on the properties within this object.

Returns

None.

Example

Sub Main(ByVal Parms As Object)

Dim cbo As New WebPageDesc
    cbo.plugInName = "UserScript1"
    cbo.plugInInstance = "Configuration Page Link"
    cbo.link = "UtilityConfig"
   .cbo.linktext = "Utility Script System Configuration"
    cbo.page_title = "Utility System Configuration Page"
    hs.RegisterLinkEx(cbo)

End Sub

See also
GetPageFooter
GetPageHeader
WebValidateUser
WebStatsPageViews
WebServerSSLPort
WebServerPort
WebLoggedinUser
GetUsers
GetPlugLinks
RegisterHelpLink
UnRegisterHelpLinks
WebPageDesc Object

This class object is used by several functions which register web pages or links to web pages in the HomeSeer web server or web page menus.

```vbp
Public Class WebPageDesc
  Public plugInName As String = "" ' When used by a script call to register a non-plugin link,
  ' this is a name to be associated with the link so that
  ' links can be grouped and for the removal of links done
  ' via other calls.
  Public plugInInstance As String = "" ' When used by a script call to register a non-plugin link,
  ' this is a unique string that can be used to unregister
  ' an individual link from a group of links registered under
  ' the same plugInName.
  Public link As String = "" ' The link to be registered. For example, "MyAboutPage",
  ' once registered, could be accessed using:
  ' http://(HomeSeer:Port)/MyAboutPage
  Public linktext As String = "" ' The text to appear in the HomeSeer menu system for the link.
  Public page_title As String = "" ' The title to be displayed for the web page.
  Public order As Integer
  ' Used by RegisterHelpLink only to determine the display order of help links.
End Class
```

See also

UnRegisterHelpLinks

Purpose

This call removes all of the registered help resource links for the plug-in or script/ASPX registered with the provided name. See RegisterHelpLink for more information on registering a help resource.

- This procedure is only valid in HomeSeer HS2 versions after 2.2.0.0.
- Help resources that exist on the hard drive such as a static html document do not need to be explicitly unregistered. However, when a help resource is provided by a plug-in or when the help resource requires the use of a plug-in, then this procedure should be used to unregister the resource when the plug-in shuts down (e.g. ShutdownIO) so the user does not have a link displayed that will not work properly.

Parameters

Parameter: plug-in name
Type: String
Description: This is the plug-in name or some sort of unique identifier for a script or ASPX based system. This identifier is used to group multiple links from the same plug-in or script/ASPX together, and it is displayed as a heading on the help page. It is not required that a plug-in use its IFACE_NAME value, but it is necessary to use the same text here as when you used RegisterHelpLink to register the link in the first place.

Returns

None.

Example

To unregister all help resources for the Acme_Widgets plug-in, which were registered using a plug-in name of "Acme Application"

```vbp
hs.UnRegisterHelpLinks("Acme Application")
```
Callbacks

In This Section

RegisterStatusChangeCB
UnRegisterStatusChangeCB

See also
System Information
System Functions
INI File Editing
Plug-Ins
Logging
Web Pages
Launch
SendMessage
ReplaceVariables

RegisterStatusChangeCB

Purpose

HomeSeer has the ability to trigger events based on a device changing. It may be useful to run a script when a device changes. The RegisterStatusChangeCB function can be used to register your script with HomeSeer. When a device changes, your script will be called. The script is passed the code of the device that changed, the address of the device that changed, as well as the value the device changed to and the reference ID of the device.

To remove the callback script, call hs.UnRegisterStatusChangeCB. There are no parameters with this call.

Remarks

When a device changes status, the given script is called as follows:

script_name( parm )

The parms parameter is an array of parameters. The following parameters are available:

parm(0) = Code part of the Address of the device that changed status
parm(1) = Full Address of the device that changed status (including Code if present)
parm(2) = The New value of the device that changed status.
parm(3) = The Old value of the device that changed status.
parm(4) = The device reference ID number. Can be used with GetDeviceByRef to find the DeviceClass of the specific device.

Note that since a function can be called in the callback script, the registration and actual callback can all reside in the same script file.

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 script is assumed to be in the scripts directory (C:\Program Files\HomeSeer HS3\Scripts by default).

Parameter: Function
Type: String
Description: This is the function in the script to run, such as Main.

Returns
Return: Result
Type: Boolean
Description: If True, the registration of the script succeeded.

Example
' register a callback script

Sub Main(ByVal Parms As Object)
    hs.RegisterStatusChangeCB("Stat_Change.vb", "StatusChangeCB")
End Sub

' the actual status change script that is called when a device changes status given the above register call

Sub StatusChangeCB(ByVal Parm As Object())
    If Parm Is Nothing Then Exit Sub
    If Parm.Length < 5 Then Exit Sub
    Dim Code As String ="
    Dim Address As String ="
    Dim OldVal As Double
    Dim NewVal As Double
    Dim Ref As Integer
    Try
        Code = Parm(0).ToString
        Address = Parm(1).ToString
        NewVal = Parm(2)
        OldVal = Parm(3)
        Ref = Parm(4)
    Catch ex As Exception
    Exit Sub
    End Try
    hs.WriteLog("Change", hs.DeviceName(Ref) & " changed from " & OldVal.ToString & " to " & NewVal.ToString & " (" & Address & ")")
End Sub

See also
UnRegisterStatusChangeCB
Purpose

This function removes a script associated with a status change as set with RegisterStatusChangeCB.

Parameters

Parameter: Script
Type: String
Description: This is the name of the script file provided when RegisterStatusChangeCB was called.

Returns

None.

See also

RegisterStatusChangeCB

Launch

Function Launch(ByName Name As String, ByVal Params As String, ByVal Directory As String, ByVal LaunchPri As Integer) As Integer

Purpose

Launches a given application. The function will return before the application finishes launching.

Parameters

Parameter: Name
Type: String
Description: This is the name of the EXE file to launch. It can be a simple application name (the path to the application would have to be in your system path) or it can be a full path name to the file. Application files can also be launched and the application that owns the file will be executed.

Parameter: Params
Type: String
Description: Any parameters or command line switches that are to be passed to the application.

Parameter: Directory
Type: String
Description: The working directory the application is launched from. Leave an empty string for most applications.

Parameter: LaunchPri
Type: Integer
Description: The running priority of the process. Use -1 for Below Normal, 0 for Normal, 1 for Above Normal.

Returns

Return value: Process Instance
Type: Integer
Description: The instance number of the process (not very useful).

See also

System Information
System Functions
INI File Editing
Plug-Ins
Logging
Web Pages
Callbacks
SendMessage
ReplaceVariables
SendMessage

Purpose

This command transmits a text message to speaker clients and controls how it is to be displayed.

- This command is only valid with the Professional edition of HomeSeer.

Parameters

Parameter: message
Type: string
Description: The text of the message you wish to send.

Parameter: host
Type: string
Description: The host name, comma separated hosts list, or comma separated list of host:instance names that you wish to send the message to.

Parameter: showballoon
Type: boolean
Description: If set to TRUE, the text will be displayed in a balloon popup window in the system tray.

Returns

Return value: error
Type: integer (.NET Enum, Short)
Description: 1 = No Error, 2 = There are no speaker clients to send to, 3 = An error occurred during the sending.

Example

Sub Main()
    Dim i
    i = hs.SendMessage("Hello, World.","*:Default", True)
    if i > 1 then
        hs.WriteLog "Error","SendMessage failed with code " & CStr(i)
    end if
End Sub

See also

System Information
System Functions
INI File Editing
Plug-Ins
Logging
Web Pages
Callbacks
Launch
ReplaceVariables

ReplaceVariables

Purpose

Does string variable replacement in a script. The variables that can be replaced are the same as those listed here.
Using Replacement Variables

Replacement variables are a series of special characters that you can use in text being spoken or in the subject or body of an email. When HomeSeer encounters one of these variables, it substitutes the information indicated by the variable in place of the variable.

Example

hs.Speak "The time is $$time"

Results in (at 11AM):  "The time is 11:00 AM"

HomeSeer Replacement Variables

(Replacement Variables are Case Insensitive)

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$date</td>
<td>Replacement is the current date in long format, e.g.: April 1, 2006</td>
</tr>
<tr>
<td>$time</td>
<td>Replacement is the current time in 12 hour format, e.g. 2:00 PM</td>
</tr>
<tr>
<td>$$date</td>
<td>Replacement is the same as $date, but it is wrapped with the SAPI context tag for date so the text to speech engine knows it is a date being spoken. Use $$date when the output is going to be spoken.</td>
</tr>
<tr>
<td>$$time</td>
<td>Replacement is the same as $time, but it is wrapped with the SAPI context tag for time so the text to speech engine knows it is a time being spoken. Use $$time when the output is going to be spoken.</td>
</tr>
<tr>
<td>$from</td>
<td>Replacement is the email address of the last email received.</td>
</tr>
<tr>
<td>$$DVA:(address):</td>
<td>Replacement is the VALUE of the device indicated by (address). For example, if the device at address R40 has a value of 100, then using $$DVA:R40: in the text will result in 100 after the substitution.</td>
</tr>
<tr>
<td>$$DVC:(code):</td>
<td>Same as $$DVA but gets the device value using the device code.</td>
</tr>
<tr>
<td>$$DVR:(ref):</td>
<td>Same as $$DVA but gets the device value using the device reference number.</td>
</tr>
<tr>
<td><strong>$$DSA:(address):</strong></td>
<td>Replacement is the STATUS of the device indicated by (address). For example, if the device at address S39 has a status of &quot;Disarmed&quot;, then using $$DSA:S39: in the text will result in &quot;Disarmed&quot; after the substitution.</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>HTML used in the status may result in problems when the replaced text is spoken.</td>
</tr>
<tr>
<td><strong>$$DSC:(code):</strong></td>
<td>Same as $$DSA but gets the status using a device code.</td>
</tr>
<tr>
<td><strong>$$DSR:(ref):</strong></td>
<td>Same as $$DSA but gets the status using a device reference number.</td>
</tr>
<tr>
<td><strong>$$DTA:(address):</strong></td>
<td>Replacement is the STRING of the device indicated by (address). For example, if the device at address S39 has a string of &quot;Come listen to a story about a man named &lt;b&gt;Jed&lt;/b&gt;&quot;, then using $$DTA:S39: in the text will result in &quot;Come listen to a story about a man named Jed&quot; after the substitution.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>HTML used in the status may result in problems when the replaced text is spoken.</td>
</tr>
<tr>
<td><strong>$$DTC:(code):</strong></td>
<td>Same as $$DTA but gets the string using a device code.</td>
</tr>
<tr>
<td><strong>$$DTR:(ref):</strong></td>
<td>Same as $$DTA but gets the string using a device reference number.</td>
</tr>
<tr>
<td><strong>$$LCI:</strong></td>
<td>(Windows Only) Replacement is information about the last phone caller, caller ID information.</td>
</tr>
<tr>
<td><strong>$$CIN:</strong></td>
<td>(Windows Only) Replacement is the caller ID name of the last call.</td>
</tr>
<tr>
<td><strong>$$CIN:</strong></td>
<td>(Windows Only) Replacement is the caller ID number of the last call.</td>
</tr>
<tr>
<td><strong>$$LVM:</strong></td>
<td>(Windows Only) Replacement is the last voice message that was left (who left it, when it was left, and how long the message was)</td>
</tr>
<tr>
<td><strong>$$COUNTER:(name):</strong></td>
<td>Replacement is the value of a specific counter. If you have a counter named &quot;dryer_counter&quot; then you could speak the value of this counter with: $$COUNTER:dryer_counter:</td>
</tr>
<tr>
<td><strong>$$TIMER:(name):</strong></td>
<td>Replacement is the value of a specific timer. If you have a timernamed &quot;dryer_timer&quot; then you could speak the value of this timer with: $$TIMER:dryer_timer:</td>
</tr>
</tbody>
</table>

See also

Home > Scripting > Computer

**Computer**

**In This Section**

- Serial Port Communication
- Network Information
- GetOSVersion
- RecurseFiles
- RecurseFilesEx
- RestartSystem
- UnZip
- Zip
- Keys

See also

About Scripts
Applications and Plugins
Devices
Email
Events
Internet
Phone
Scripts
Speech Recognition
Strings, Global Variables, and Encryption
Time and Calendar
Text-To-Speech and Media
OpenComPort

Purpose

This function opens a communication port. If the port is already open by another application, an error occurs. Once a port is opened, it remains open until the CloseComPort function is called. The port is not closed when the script terminates. However, the port will close when the application terminates.

- In previous versions of HomeSeer, the OpenComPort function was limited to COM ports 1 to 8 and OpenComPortEx was for ports above 8. Beginning with HomeSeer Version 2.0, there is no limit on the number of COM ports, so these functions can be used interchangeably. The "Resource" parameter in the OpenComPortEx function is not used in HomeSeer 2.0, but the parameter is included in order to support scripts created with older versions of HomeSeer.
- If you need to control the hardware handshaking signals Request To Send (RTS) or Data Terminal Ready (DTR), please refer to the SetComPortRTSDTR command.

Parameters

- Parameter: **port**
  Type: integer
  Description: This is the port number to open. An error is returned if the port is already open or is not installed on the system. To use port numbers above 8, see the resource description below.

- Parameter: **config**
  Type: string
  Description: Port Configuration (see below).

- Parameter: **mode**
  Type: integer
  Description: Operating Mode (see below).

- Parameter: **cb_script**
  Type: string
  Description: Port Data Handling Script (see below).

- Parameter: **cb_func**
  Type: string
  Description: Function in Port Handling Script (see below).

Returns

The function returns an empty string if it was successful, otherwise it returns a text string describing the error.

Port Configuration

The **config** parameter is composed of four settings and has the format **BBBB, P, D, S**.

**BBBB** is the baud rate, **P** is the parity, **D** is the number of data bits, and **S** is the number of stop bits. For example, to set the port to 9600 baud, no parity, 8 bit and no stop bits, the **config** string would be **9600, N, 8, 1**.
The valid baud rates are listed below.

```
110  2400  19200  57600
300  4800  28800  115200
600  9600  38400  128000
1200 14400  57600  256000
```

The parity values are:
- E = Even
- M = Mark
- N = None
- O = Odd
- S = Space

The data bit values are:
- 5
- 6
- 7
- 8

The stop bit values are:
- 1
- 1.5
- 2

**Operating Mode**

This parameter affects the way data is received on the COM port. Two modes are available:

- **0 = raw mode**
  - In this mode, each character that is received on the COM port causes the specified script and function to be called. It is up to the called function to call GetComPortData to actually get the characters.

- **1 = strings mode**
  - This mode buffers up characters until a terminator is received. At this point the specified script and function are called with the data. This mode makes it easy to deal with devices that send text data terminated with known characters. To specify the terminator characters, see the term parameter description below. If you do not specify a terminator, the default terminator of carriage return and line-feed (CrLf) are used. To set the terminator character, use OpenComPortTerm.

**Port Data Handling Script**

This parameter is the name of the script that will be called when COM port data arrives. The script will be called with a single parameter, which is the received text string. If you do not wish to be called back when data is received, leave this parameter as an empty string. You can still use the GetComPortData function to poll for data yourself. The following example shows what your called script should look like.

```
sub callback(data)
    ' handle the data
end sub
```

**Function in Port Data Handling Script**

This is the function that will be called in the specified script. If your script was defined as above, the cb_func parameter would be set to callback.

If this parameter is omitted, the main function will be called by default.

**Termination String**

This is the termination string for mode 1 operation. Characters will be received into the COM port buffer until this termination string is found in the buffer. If this parameter is not provided, then the default value is the character pair of carriage return and line-feed (CrLf). See OpenComPortTerm for assigning a termination string.

See also
- OpenComPortTerm
- OpenComPortEx
- SetComPortRTS/DSR
- SendToComPort
- GetComPortCount
- GetComPortData
- CloseComPort
OpenComPortTerm

Purpose
This procedure is identical to OpenComPort but accepts one extra parameter which is the terminator character. This character is checked when data on the COM port is received and if detected will then call the callback script.

See OpenComPort for more information.

See also
OpenComPort
OpenComPortEx
SetComPortRTSDTR
SendToComPort
GetComPortCount
GetComPortData
CloseComPort

OpenComPortEx

Purpose
This function opens a communication port. If the port is already open by another application, an error occurs. Once a port is opened, it remains open until the CloseComPort function is called. The port is not closed when the script terminates. However, the port will close when the application terminates.

- In previous versions of HomeSeer, the OpenComPort function was limited to COM ports 1 to 8 and OpenComPortEx was for ports above 8. Beginning with HomeSeer Version 2.0, there is no limit on the number of COM ports, so these functions can be used interchangeably. The "Resource" parameter is not used in HomeSeer 2.0, but the parameter is included in order to support scripts created with older versions of HomeSeer.
- If you need to control the hardware handshaking signals Request To Send (RTS) or Data Terminal Ready (DTR), please refer to the SetComPortRTSDTR command.

Parameters
Parameter: port
Type: integer
Description: This is the port number to open. An error is returned if the port is already open or is not installed on the system. To use port numbers above 8, see the resource description below.

Parameter: config
Type: string
Description: Port Configuration (see below).

Parameter: mode
Type: integer
Description: Operating Mode (see below).

Parameter: cb_script
Type: string
Description: Port Data Handling Script (see below).

Parameter: cb_func
Type: string
Description: Function in Port Handling Script (see below).

Parameter: term (optional)
Type: string
Description: Termination String (see below). This parameter is not used in HomeSeer 2.0 but is included for backward-compatibility with scripts created in older versions of HomeSeer.

Parameter: resource (optional)
Type: integer
Description: Resource Number (see below). This parameter is not used in HomeSeer 2.0 but is included for backward-compatibility with scripts created in older versions of HomeSeer.
Returns

The function returns an empty string if it was successful, otherwise it returns a text string describing the error.

Port Configuration

The *config* parameter is composed of four settings and has the format `BBBB,P,D,S`.

`BBBB` is the baud rate, `P` is the parity, `D` is the number of data bits, and `S` is the number of stop bits. For example, to set the port to 9600 baud, no parity, 8 bit and no stop bits, the *config* string would be `9600,N,8,1`.

The valid baud rates are listed below.

<table>
<thead>
<tr>
<th>Baud Rate</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>2400</td>
</tr>
<tr>
<td>300</td>
<td>4800</td>
</tr>
<tr>
<td>600</td>
<td>9600</td>
</tr>
<tr>
<td>1200</td>
<td>14400</td>
</tr>
<tr>
<td>2400</td>
<td>28800</td>
</tr>
<tr>
<td>4800</td>
<td>38400</td>
</tr>
<tr>
<td>9600</td>
<td>57600</td>
</tr>
<tr>
<td>115200</td>
<td>115200</td>
</tr>
<tr>
<td>19200</td>
<td>28800</td>
</tr>
<tr>
<td>230400</td>
<td>38400</td>
</tr>
<tr>
<td>256000</td>
<td>57600</td>
</tr>
</tbody>
</table>

The parity values are:

- `E` = Even
- `M` = Mark
- `N` = None
- `O` = Odd
- `S` = Space

The data bit values are:

- 5
- 6
- 7
- 8

The stop bit values are:

- 1
- 1.5
- 2

Operating Mode

This parameter affects the way data is received on the COM port. Two modes are available:

- **0 = raw mode**
  
  In this mode, each character that is received on the COM port causes the specified script and function to be called. It is up to the called function to call `GetComPortData` to actually get the characters.

- **1 = strings mode**
  
  This mode buffers up characters until a terminator is received. At this point the specified script and function are called with the data. This mode makes it easy to deal with devices that send text data terminated with known characters. To specify the terminator characters, see the *term* parameter description below. If you do not specify a terminator, the default terminator of carriage return and line-feed (CrLf) are used.

Port Data Handling Script

This parameter is the name of the script that will be called when COM port data arrives. The script will be called with a single parameter, which is the received text string. If you do not wish to be called back when data is received, leave this parameter as an empty string. You can still use the `GetComPortData` function to poll for data yourself. The following example shows what your called script should look like.

```vbs
sub callback(data)
  ' handle the data
end sub
```

Function in Port Data Handling Script

This is the function that will be called in the specified script. If your script was defined as above, the `cb_func` parameter would be set to `callback`. If this parameter is omitted, the `main` function will be called by default.

Termination String

This is the terminator string for mode 1 operation. Characters will be received into the COM port buffer until this termination string is found in the buffer. If this parameter is not provided, then the default value is the character pair of carriage return and line-feed (CrLf).

Resource Number

This parameter is no longer necessary in HomeSeer 2.0 but is included for backward-compatibility with scripts created in older versions of HomeSeer.

This is a resource number to allocate `OpenComPortEx` resources so that `OpenComPortEx` can be used with COM ports above 8. There are 8 resources available between `OpenComPort` and `OpenComPortEx`. When you wish to use COM ports above 8 you can specify the higher COM port.
number for the port parameter, but then you must specify a resource number with this parameter. HomeSeer does NOT keep track of used resource numbers. If COM3 is opened with OpenComPort, which means resource 3 was assigned to it, you must remember not to use resource 3 with OpenComPortEx.

See also
OpenComPort
OpenComPortTerm
SetComPortRTSDTR
SendToComPort
GetComPortCount
GetComPortData
CloseComPort

SetComPortRTSDTR

**Purpose**

Sets the levels of the RTS and DTR signals on the given COM port.

**Parameters**

Parameter: port
Type: integer
Description: This is the COM port to access or the resource number of the port to access if OpenComPortEx was used to open it.

Parameter: rts_val
Type: boolean
Description: Set to TRUE to raise the RTS line or set to FALSE to lower the line.

Parameter: dtr_val
Type: boolean
Description: Set to TRUE to raise the DTR line or set to FALSE to lower the line.

**Returns**

None.

See also
OpenComPort
OpenComPortTerm
OpenComPortEx
SendToComPort
GetComPortCount
GetComPortData
CloseComPort

SendToComPort

**Purpose**

Send a string of characters out a communications port. The port must have been previously opened with the OpenComPort or OpenComPortEx call.

- Some devices that you are communicating with require a special character to terminate the string of characters you are sending to it. For example, a modem needs a carriage-return (CR) at the end of the string you send to it before it will be recognized. Some devices may require a carriage-return and a line-feed character, others perhaps something entirely different. Please be aware of the requirements of the device you are communicating with. If you require hardware handshaking on the communications port, please see the SetComPortRTSDTR command.

**Parameters**
Home > Scripting > Computer > Serial Port Communication > GetComPortCount

GetComPortCount

Purpose

Returns the number of received characters available on a communications port. This function can be used to poll the COM port for data. The best way to receive characters on a COM port is to use the callback function that is set with OpenComPort or OpenComPortEx.

Parameters

Parameter: port
Type: integer
Description: The port number of the port to check or the resource number of the port to be checked if OpenComPortEx was used to open it.

Returns

Return value: number
Type: integer
Description: The number of characters available at the COM port.

See also
OpenComPort
OpenComPortTerm
OpenComPortEx
SetComPortRTSDTR
GetComPortCount
GetComPortData
CloseComPort

GetComPortData

Purpose

Returns the data available at a COM port. The data is a variant and could be a text string or an array of bytes, depending on the type of data available. This function is not used if the COM port is opened in mode 1. If the port is opened as mode 0, this function should be used in your callback function to get the data.

See also
OpenComPort
OpenComPortTerm
OpenComPortEx
SetComPortRTSDTR
SendToComPort
GetComPortData
CloseComPort
**Parameters**

Parameter: port  
Type: integer  
Description: The COM port to read or the resource number of the port to be read if OpenComPortEx was used to open it.

**Returns**

Return value: data  
Type: variant  
Description: The data available is a string of characters.

See also

OpenComPort  
OpenComPortTerm  
OpenComPortEx  
SetComPortRTS/DSR  
SendToComPort  
GetComPortCount  
CloseComPort

---

**CloseComPort**

**Purpose**

Closes a communications port previously opened with OpenComPort, or the communications port associated with a resource that was opened with the OpenComPortEx command.

**Parameters**

Parameter: port  
Type: integer  
Description: The number of the port to be closed or the resource number of the port to be closed if OpenComPortEx was used to open it.

**Returns**

None.

See also

OpenComPort  
OpenComPortTerm  
OpenComPortEx  
SetComPortRTS/DSR  
SendToComPort  
GetComPortCount  
GetComPortData

---

**Network Information**

See also

Serial Port Communication  
GetOSVersion  
RecurseFiles  
RecurseFilesEx
GetIPAddress

**Purpose**

Returns the IP address of your computer as a string like 192.168.1.1. Note that if your computer has multiple network interfaces, this will return the IP address of each interface separated by a "space" character: 192.168.1.1 192.168.1.2.

If you wish to get the IP address and hostname of the machine, please see LANIP.

**Parameters**

None.

**Returns**

Return value: IP address  
Type: string

**Example**

```vbs
sub main()
    dim ipaddress
    ipaddress = hs.GetIPAddress
    hs.WriteLog "Info","The IP Address is " & ipaddress
end sub
```

**See Also**

GetLastRemoteIP  
LANIP  
WANIP

GetLastRemoteIP

**Purpose**

Returns the IP address of the last client computer to access the HomeSeer web server, as a string like 192.168.1.1.

**Parameters**

None.

**Returns**

None.
Purpose

Provides the IP address and hostname of the HomeSeer computer's primary network interface, as seen from the local (in house) network. If you want the IP address only, please see GetIPAddress.

Parameters

None.

Returns

Return value: IP
Type: string
Description: The IP address and hostname is returned in the format: xxx.xxx.xxx.xxx (hostname)

See Also

GetIPAddress
GetLastRemoteIP
WANIP

See also
GetIPAddress
GetLastRemoteIP
Ping
WANIP
Ping

**Purpose**
Indicates of a host is available.

**Parameters**
- **Parameter:** host name
  - **Type:** string
  - **Description:** Name or IP address of the host to ping.

**Returns**
- **Return value:** host status
  - **Type:** integer
  - **Description:** Returns 0 if host is alive and 26118 if host is not available.

See also
- GetIPAddress
- GetLastRemoteIP
- LANIP
- WANIP

WANIP

**Purpose**
Provides the IP address and hostname of the HomeSeer computer's primary network interface as seen from the Internet.

**Parameters**
- None.

**Returns**
- **Return value:** IP
  - **Type:** string
  - **Description:** The IP address and hostname is returned in the format: `xxx.xxx.xxx.xxx (hostname)`

See Also
- GetIPAddress
- GetLastRemoteIP
- LANIP

See also
- GetIPAddress
- GetLastRemoteIP
- LANIP
- Ping

GetOSVersion
**Purpose**

Returns the version of the operating system running HomeSeer.

**Parameters**

None.

**Returns**

Return value: **OS Version**

- **Type:** string
- **Example:** 5.1.0.2600

**Example**

```vba
sub main()
    hs.WriteLog "Info","The Operating System version is " & hs.GetOSVersion
end sub
```

See also

- Serial Port Communication
- Network Information
- GetOSVersion
- RecurseFilesEx
- RestartSystem
- UnZip
- Zip
- Keys

---

**RecurseFiles**

**Purpose**

This command returns a comma separated string of files that are in the starting directory and all sub-directories within it.

**Parameters**

- **Parameter:** Starting Directory
  - **Type:** string
  - **Description:** The full path to the starting directory to be recursed.

**Returns**

- **Parameter:** file list
  - **Type:** string
  - **Description:** The list of files in the starting directory and sub-directories, separated by a comma.

See also

- Serial Port Communication
- Network Information
- GetOSVersion
- RecurseFilesEx
- RestartSystem
- UnZip
- Zip
- Keys
RecurseFilesEx

Function RecurseFilesEx(ByVal SourceDir As String) As String

Purpose
This command returns an array of strings that are in the starting directory and all sub-directories within it.

Parameters
Parameter: Starting Directory
Type: string
Description: The full path to the starting directory to be recursed.

Returns
Parameter: file list
Type: string array
Description: The list of files in the starting directory and sub-directories, one entry per array element.

See also
Serial Port Communication
Network Information
GetOSVersion
RecurseFiles
RestartSystem
UnZip
Zip
Keys

RestartSystem

Purpose
This command will shut down HomeSeer and restart your computer system. Use with caution!

The shut down command for HomeSeer is coming from the computer system. Thus, it is possible for the system to restart before HomeSeer has completed its shut down processing.

Parameters
None.

Returns
None.

See also
Serial Port Communication
Network Information
GetOSVersion
RecurseFiles
RecurseFilesEx
UnZip
Zip
Keys
UnZip

Purpose
This command will unzip a Zip archive file to the destination you provide.

Parameters
- **filename**
  - Type: string
  - Description: Path and name of the source Zip archive file to be unzipped.
- **destination** (optional)
  - Type: string
  - Description: Path to the destination starting directory for the files in the zip archive. If this parameter is not provided, the files in the Zip archive will be unzipped to the same directory as the source zip file.
- **IgnoreZipDirs** (optional)
  - Type: boolean
  - Description: If True, the zip directories within the Zip archive will be ignored and all of the files will be unzipped into the destination directory only. (Note: Two files of the same name in different Zip archive directories will result in only one of them existing at the end of the UnZip operation if this parameter is set to True.) (Default is False)
- **OverWrite** (optional)
  - Type: boolean
  - Description: If set to True, existing destination files will be overwritten. (Default is False)
- **password** (optional)
  - Type: string
  - Description: If the source Zip archive was created with a password, provide it in this parameter.

Returns
- **Status**
  - Type: String
  - Description: Returns the status of the UnZip function. An empty string indicates success, otherwise an error message is returned.

Example

See also
- Serial Port Communication
- Network Information
- GetOSVersion
- RecurseFiles
- RecurseFilesEx
- RestartSystem
- Zip
- Keys

Zip

Purpose
This command will zip up files and create a zip archive file.

Parameters
- **ZipWhat**
  - Type: string
  - Description: Path to a directory, or path and filename of the directory or file to be added to a zip archive file.
- **ZipFileName**
  - Type: string
  - Description: Path and filename of the Zip archive file to be created or to have files added to.
Parameter: **compression** (optional)
Type: integer (.NET Short)
Description: The zip file compression level to use - the higher the level, the longer the zip operation will take. If this parameter is not provided, a default value of 6 is used. The valid values are from 0 to 9.

Parameter: **password** (optional)
Type: string
Description: If you wish the file(s) to be password protected in the archive, provide the password here. (Case sensitive)

Parameter: **RemoveBase** (optional)
Type: boolean
Description: If False, the entire directory structure up to and including the source file or directory will be included in the zip archive. If this parameter is not provided, then by default it is True and the directory structure before the starting point of the source files will be removed.

Example: ZipWhat is C:\Program Files\HomeSeer\HTML\MyStuff (a directory) MyStuff has sub-directories Stuff1 and Stuff2.

With this parameter True, the resulting archive will have Stuff1\*.* and Stuff2\*.* in it.

With this parameter False, the resulting archive will have Program Files\HomeSeer\HTML\MyStuff\Stuff1\*.* and Program Files\HomeSeer\HTML\MyStuff\Stuff2\*.* in it.

Parameter: **Flatten** (optional)
Type: boolean
Description: If True, the files will be put in the zip archive without any path information. Files that have the same filename but are in different subdirectories will result in only one of the files being left in the archive at the end of the Zip function.

**Returns**

Parameter: **Status**
Type: String
Description: Returns the status of the zip function. An empty string indicates success, otherwise an error message is returned.

**Example**

```vbnet
Sub Main(parm as object)
  Dim result as String
  result=hs.Zip("c:\file.txt","c:\file.zip")
  hs.WriteLog("ZIP","Result: " & result)
End Sub
```

**See also**

Serial Port Communication
Network Information
GetOSVersion
RecurseFiles
RecurseFilesEx
RestartSystem
UnZip
Keys

**Keys**

**Sub** Keys(ByVal KeyCode As String, ByVal Title As String, ByVal Wait As Boolean)

**Purpose**

This function allows you send keyboard commands to a running application. This is merely an interface into the .NET SendKeys.Send function.

Each key is represented by one or more characters. To specify a single keyboard character, use the character itself. For example, to represent the letter A use "A" for the string. To represent more than one character, append each additional character to the one preceding it. To represent the letters A, B, and C, use "ABC" for the string.

The plus sign (+), caret (^), percent sign (%), tilde (~), and parentheses ( ) have special meanings to the SendKeys function. To specify one of these characters, enclose it within braces ({}). For example, to specify the plus sign, use {+}. Brackets [[]] have no special meaning to SendKeys, but you must enclose them in braces. In other applications, brackets do have a special meaning that may be significant when dynamic data exchange (DDE) occurs. To specify brace characters, use {{}} and {}}.

To specify characters that aren't displayed when you press a key, such as ENTER or TAB, and keys that represent actions rather than characters, use
the codes shown below:

<table>
<thead>
<tr>
<th>Key</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKSPACE</td>
<td>{BACKSPACE}, {BS}, or {BKSP}</td>
</tr>
<tr>
<td>BREAK</td>
<td>{BREAK}</td>
</tr>
<tr>
<td>CAPS LOCK</td>
<td>{CAPSLOCK}</td>
</tr>
<tr>
<td>DEL or DELETE</td>
<td>{DELETE} or {DEL}</td>
</tr>
<tr>
<td>DOWN ARROW</td>
<td>{DOWN}</td>
</tr>
<tr>
<td>END</td>
<td>{END}</td>
</tr>
<tr>
<td>ENTER</td>
<td>{ENTER} or ~</td>
</tr>
<tr>
<td>ESC</td>
<td>{ESC}</td>
</tr>
<tr>
<td>HELP</td>
<td>{HELP}</td>
</tr>
<tr>
<td>HOME</td>
<td>{HOME}</td>
</tr>
<tr>
<td>INS or INSERT</td>
<td>{INSERT} or {INS}</td>
</tr>
<tr>
<td>LEFT ARROW</td>
<td>{LEFT}</td>
</tr>
<tr>
<td>NUM LOCK</td>
<td>{NUMLOCK}</td>
</tr>
<tr>
<td>PAGE DOWN</td>
<td>{PGDN}</td>
</tr>
<tr>
<td>PAGE UP</td>
<td>{PGUP}</td>
</tr>
<tr>
<td>PRINT SCREEN</td>
<td>{PRTSC}</td>
</tr>
<tr>
<td>RIGHT ARROW</td>
<td>{RIGHT}</td>
</tr>
<tr>
<td>SCROLL LOCK</td>
<td>{SCROLLLOCK}</td>
</tr>
<tr>
<td>TAB</td>
<td>{TAB}</td>
</tr>
<tr>
<td>UP ARROW</td>
<td>{UP}</td>
</tr>
<tr>
<td>F1</td>
<td>{F1}</td>
</tr>
<tr>
<td>F2</td>
<td>{F2}</td>
</tr>
<tr>
<td>F3</td>
<td>{F3}</td>
</tr>
<tr>
<td>F4</td>
<td>{F4}</td>
</tr>
<tr>
<td>F5</td>
<td>{F5}</td>
</tr>
<tr>
<td>F6</td>
<td>{F6}</td>
</tr>
<tr>
<td>F7</td>
<td>{F7}</td>
</tr>
<tr>
<td>F8</td>
<td>{F8}</td>
</tr>
<tr>
<td>F9</td>
<td>{F9}</td>
</tr>
<tr>
<td>F10</td>
<td>{F10}</td>
</tr>
<tr>
<td>F11</td>
<td>{F11}</td>
</tr>
<tr>
<td>F12</td>
<td>{F12}</td>
</tr>
<tr>
<td>F13</td>
<td>{F13}</td>
</tr>
<tr>
<td>F14</td>
<td>{F14}</td>
</tr>
<tr>
<td>F15</td>
<td>{F15}</td>
</tr>
<tr>
<td>F16</td>
<td>{F16}</td>
</tr>
</tbody>
</table>

To specify keys combined with any combination of the SHIFT, CTRL, and ALT keys, precede the key code with one or more of the following codes:

<table>
<thead>
<tr>
<th>Key</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHIFT</td>
<td>+</td>
</tr>
<tr>
<td>CTRL</td>
<td>^</td>
</tr>
<tr>
<td>ALT</td>
<td>%</td>
</tr>
</tbody>
</table>

To specify that any combination of SHIFT, CTRL, and ALT should be held down while several other keys are pressed, enclose the code for those keys in parentheses. For example, to specify to hold down SHIFT while E and C are pressed, use "+(EC)". To specify to hold down SHIFT while E is pressed, followed by C without SHIFT, use "+{EC}".

To specify repeating keys, use the form {key number}. You must put a space between key and number. For example, {LEFT 42} means press the LEFT ARROW key 42 times; {h 10} means press H 10 times.

Note that you can't use SendKeys to send keystrokes to an application that is not designed to run in Microsoft Windows. Sendkeys also can't send the PRINT SCREEN key {PRTSC} to any application.

Parameters

Parameter: KeyCode
Type: String
Description: Is the key code to send (see below for special codes).

Parameter: Title
Type: String
Description: This is the title string that appears in the main window of the target application you wish to control.

Parameter: Wait
Type: Boolean (optional)
Description: This parameter is true to slow down the sending of the keys. Normally you want this to be TRUE, or (1).

Returns

None.
Example

This script will launch the calculator program:

```vbscript
sub main()
    dim I
    i=hs.launch("calc.exe","")
end sub
```

This script will use the calculator to add some numbers:

```vbscript
sub main()
    hs.speak "I will add some numbers"
    hs.keys "1{+}2{+}3{ENTER}","calc",1
end sub
```

See also
- Serial Port Communication
- Network Information
- GetOSVersion
- RecurseFiles
- RecurseFilesEx
- RestartSystem
- UnZip
- Zip

Devices

In This Section

- The Device Class
- Device Exists, Reference, Address and/or Code
- Creating, Deleting, or Accessing Devices
- Device Value, String, or Last Change
- Device Script Buttons
- Device Energy Management
- Device Control API (CAPI)
- Images

See also
- About Scripts
- Applications and Plugins
- Computer
- Email
- Events
- Internet
- Phone
- Scripts
- Speech Recognition
- Strings, Global Variables, and Encryption
- Time and Calendar
- Text-To-Speech and Media
DeviceClass Properties and Procedures

Use caution when working with the DeviceClass properties directly. Internally, HomeSeer will compare, for example, an address from the device class to the address provided in a script command by making both lowercase or both uppercase. When you access the DeviceClass directly, you are getting the address exactly as it was entered by the user, so one device could have an address of "Hello" while another has an address of "HeLLo".

In MOST cases of accessing a property or procedure, there is a parameter of "hs" which is the type IHSApplication. This is the hs object itself. The reason for this is for data continuity. When you access the DeviceClass from a plug-in, a COPY of the DeviceClass object is what traverses the interface to the plug-in, and the plug-in is not accessing the real object. By including the hs object, you are indicating to HomeSeer that you want the latest information (GET) or are making a change (SET) and HomeSeer uses this reference to work with the actual DeviceClass object.

Example:

When you retrieve the location without providing a valid HomeSeer Interface Object (hs):

In your script or plug-in, you get a reference to the device that you want to work with (hs.GetDeviceByRef) and store the object in the variable dv. At the time you got that object reference, the location was "Family Room"

Time passes, and through the HomeSeer User Interface, somebody has changed the location of that device to "Den".

Now, you retrieve the location name, but you do not provide a valid HomeSeer Application Interface (hs) reference:

```vbnet
Dim Loc As String = dv.Location(hs)
```

If you look at the Loc variable, it will still be "Family Room". However, if you get the location and include the hs object:

```vbnet
Dim Loc As String = dv.Location(hs)
```

Now the Loc variable contains "Den", and this is because THROUGH the hs object, HomeSeer retrieved the information from the "Live" version of the object.

Reference

**Public Property** Ref(ByVal hs As IHSApplication) As Integer

The Ref property holds the device's unique device reference number. The Ref should never be changed except by a plug-in or script which has first used a procedure to generate a Ref that is guaranteed to be unique in the system.

**Public Property** Address(ByVal hs As IHSApplication) As String

The Address is a user or plug-in assigned string that identifies the device within a logical grouping. When you GET the value of this property, it always returns the Address AND the Code separated by a hyphen. For example, if the Address were set to HELLO, and the Code were set to WORLD, retrieving the Address would result in the string "HELLO-WORLD". The Code is always set separate from the Address. This field might be used to identify the module in a machine for which there are several sub-points, and each sub-point is a different Code - as such, all of the members of the module would be given the same Address and unique Code values.

**Public Property** Code(ByVal hs As IHSApplication) As String

The Code is treated both separately and in combination with the Address property. Both GET and SET may be done on the Code, but when a GET is done on the Address, the string returned is in the format Address-Code, with the value in this property being the Code. For example, a Z-Wave device that is a part of the network 00AABBCC and is Node 6 might have 5 child devices, so each device would have an Address of 00AABBCC-6, but a unique Code such as Q01, Q02, Q03 such that any one of the devices may have a full address of 00AABBCC-6-Q02

Identity

**Public Property** Name(ByVal hs As IHSApplication) As String

The Name property holds the name of the device, such as "Light", "Lamp", or "Heater"

**Public Property** Location(ByVal hs As IHSApplication) As String

This is the location name of the device, such as "Family Room".

**Public Property** Location2(ByVal hs As IHSApplication) As String

This is a second location modifier, which may be disabled in the HomeSeer settings, or if used can be used to further qualify the location of a device such as "First Floor".

**Public Property** UserNote(ByVal hs As IHSApplication) As String

This property stores any information the user so chooses, and is also editable on the device management page by clicking on the note icon.

Status

**Public ReadOnly Property** devString(ByVal hs As IHSApplication) As String
This is the device string for the device. When this property contains a value, it can override the display of the device's normal status display which is based upon the device's value. This property may contain HTML if HTML features are desired to be used when the device is viewed on the device utility page or the status views. This property is Read Only, so script commands must be used to modify the string value such as hs.SetDeviceString.

Public ReadOnly Property devValue(ByVal hs As IHSApplication) As Double
This is the device's numerical value, which can be positive or negative and may contain a decimal point. Setting the device's value can cause changes to occur or change the status of the device. This property is Read Only, as the device value needs to be changed using commands such as hs.SetDeviceValue.

Public Property Last_Change(ByVal hs As IHSApplication) As Date
This is the date and time that the device's value or string was last updated. Some ways of updating the value or string may explicitly block this from being updated, but in most cases it reflects the date and time of the last change.

Public Property Attention(ByVal hs As IHSApplication) As String
This property is used to enable/disable the attention icon on a device when viewed in HomeSeer, and the text that is displayed when you hover your mouse over the attention icon. When set to a non-null value, HomeSeer will display an attention icon in the device management page and status views to alert the user that there is a message. When the mouse is over the alert icon, the contents of the message (this property) will be displayed.

Configuration

Public Property Device_Type_String(ByVal hs As IHSApplication) As String
The actual device type of a device is determined by information in the DeviceTypeInfo object (See DeviceType_Get and DeviceType_Set). This property may be used to hold a more "user friendly" device type string which is displayed on the device utility page. For example, if the device is owned by the Z-Wave plug-in, the DeviceTypeInfo object may identify it as a plug-in device type, but this property might display "Z-Wave Switch Multilevel".

Public ReadOnly Property DeviceType_Get(ByVal hs As IHSApplication) As DeviceTypeInfo
Public WriteOnly Property DeviceType_Set(ByVal hs As IHSApplication) As DeviceTypeInfo
The DeviceTypeInfo object holds several pieces of information describing the device type of the device. If the device is used with a technology API such as a Thermostat, Media, or Security, then the DeviceTypeInfo specifically identifies which part of the API the device fulfills.

Public Property Status_Support(ByVal hs As IHSApplication) As Boolean
This property indicates (when True) that the device supports the retrieval of its status on-demand through the "Poll" feature on the device utility page. The plug-in which owns the device is responsible for returning the status when the poll command is issued.

Public Property Can_Dim(ByVal hs As IHSApplication) As Boolean
This property is largely unused in HS3. When set to True and no other device value/status pairs have been assigned to the device, the default value/status pairs assigned will allow for levels/values from 1 to 99 in addition to 0 (Off) and 100 (On).

Public Property Image(ByVal hs As IHSApplication) As String
The image property holds a path string to an image file to represent the device on the status views pages. The image path should be referenced from the root of the HTML folder under the main HomeSeer folder.

Public Property Interface(ByVal hs As IHSApplication) As String
This property holds the name of the plug-in that owns/manages this device. If the property is null or an empty string, the device is not managed by a plug-in.

Public Property InterfaceInstance(ByVal hs As IHSApplication) As String
This property holds the instance name of the plug-in that owns/manages this device. If the property is null or an empty string, either the plug-in does not support multiple instances (if the Interface property is not blank) or the device is not managed by a plug-in.

Public Property ScriptName(ByVal hs As IHSApplication) As String
Public Property ScriptFunc(ByVal hs As IHSApplication) As String
These properties are used ONLY when the Device_Type's API is set to Script, and the Device_Type's Device_Type is set to one of the script action values (See eDeviceType_Script) This provides functionality that will cause a script to be run when the device's value, string, or either are changed. ScriptName is the name of the script file to be run, and ScriptFunc is the name of a procedure in the script file to be called - if no ScriptFunc is provided, then Sub Main will be called.

When the script is run, it will be passed parameters as an object array, and those parameters are:

- Parm(0) - Integer - The device reference ID.
- Parm(1) - DeviceScriptChange (Integer) - Indicates what changed to cause the script to be run.
- Parm(2) - Double - The device's new value.
Parm(3) - String - The device's new string.

Display

Public Property ScaleText(ByVal hs As IHSApplication) As String
A device that is used to display (for example) a temperature, the scale (Fahrenheit or Celsius) may not be known at the time the device is created or may be set/changed by an external device such that the device value/status pairs cannot be configured to display the proper scale symbol. To address this, plug-ins may update this property with the correct scale text just prior to adjusting the device's value. This property may be retrieved by other systems displaying this device's status and used in a similar manner to how it is used with the HomeSeer user interfaces.

Public Property AdditionalDisplayData(ByVal hs As IHSApplication) As String()
Similar to ScaleText, this property is used to enhance the device status display when variable elements of data are a part of the device status. For example, a Z-Wave enabled Smoke Detector may report an alarm, as well as location information. Since the variable location information cannot be assigned to value/status pairs in advance, this array of string values may be used.

Device Association

Public Sub AssociatedDevice_Add(ByVal hs As IHSApplication, ByVal dvRef As Integer)
Public ReadOnly Property AssociatedDevices_Count(ByVal hs As IHSApplication) As Integer
Public Sub AssociatedDevice_ClearAll(ByVal hs As IHSApplication)
Public ReadOnly Property AssociatedDevices(ByVal hs As IHSApplication) As Integer()
Public ReadOnly Property AssociatedDevices_List(ByVal hs As IHSApplication) As String
Public Sub AssociatedDevice_Remove(ByVal hs As IHSApplication, ByVal dvRef As Integer)

Public ReadOnly Property Parent As Enums.eRootChildStatus
Public ReadOnly Property Child As Enums.eRootChildStatus

These procedures and properties allow for getting information or making changes regarding the association of devices to one another. The typical usage is to associate one device (for example a Z-Wave Root Device) with several devices (Z-Wave Child Devices). For devices owned by plug-ins which represent technology API devices, it is strongly recommended for enumeration purposes that the single parent/multiple child relationship is used.

Associating devices should also be accompanied by the setting of the Root (Parent) device type on the parent device in a cluster of related devices. Each defined Device Type API contains a Device Type which indicates a Root device for that API, and a Device Type constant also exists to indicate a root device in the situation where there is a parent/child relationship between devices that do NOT belong to a specific technology API.

To add an association of another device to a device, use AssociatedDevice_Add. Example: dv.AssociatedDevice_Add(hs, 1234) - associates the device referenced by the device ID 1234 to the device class object dv.

To determine how many devices are associated to a device (in the device class object ‘dv’), use AssociatedDevices_Count.

AssociatedDevice_ClearAll will remove all associated devices from the one which the AssociatedDevice_ClearAll procedure is called from.

AssociatedDevices and AssociatedDevices_List both return the device reference ID numbers for any devices associated with the device in which the property/procedure is called. AssociatedDevices returns an array of integers, and AssociatedDevices_List returns a comma separated string list.

AssociatedDevice_Remove will remove a single associated device reference number from the list of associated devices, which removes the association.

Relationship Status

Public Property Relationship(ByVal hs As IHSApplication) As Enums.eRelationship
The Relationship property can be used to determine or set the parent (root) or child status of a device. The return value is an Enum (eRelationship) which can indicate whether the device is a Parent/Root (it has child devices associated with it), a Child device (it is associated with a parent device), or Standalone (it is not associated with any other device). Additionally, the return value will indicate Not Set if the device has never had one of the values set to it, or Indeterminate, which may be used by a device in the process of being created or that could be in transition from one state to another. (Indeterminate is rarely used.)

Misc Bits - Check, Clear, Set

Public Function MISC_Check(ByVal hs As IHSApplication, ByVal Misc As Enums.dvMISC) As Boolean
Public Sub MISC_Clear(ByVal hs As IHSApplication, ByVal MISC As Enums.dvMISC)
Public Sub MISC_Set(ByVal hs As IHSApplication, ByVal MISC As Enums.dvMISC)

These procedures allow you to determine if various bits in the device's MISC settings are set or not, or to make changes to those bit settings. All of these procedures are aided by the use of an Enum called dvMISC so that more friendly names may be used instead of odd numerical values. The list of dvMISC values may be viewed here.

MISC_Check is used to determine if the selected MISC bit is Set (Returns True) or not set/cleared (Returns False).
Example: If dv.MISC_Check(hs, Enums.dvMISC.NO_LOG) Then can be used to determine if the NO_LOG option was set.

MISC_Set and MISC_Clear are used to Set/Enable or Reset/Clear the indicated bit respectively.
dvMISC

This Enum holds values referencing individual bits in an integer which indicate different characteristics of a device.

```
Enum dvMISC As UInteger
    NO_LOG = 8
    STATUS_ONLY = &H10
    HIDDEN = &H20
    INCLUDE_POWERFAIL = &H80
    SHOW_VALUES = &H100
    AUTO_VOICE_COMMAND = &H200
    VOICE_COMMAND_CONFIRM = &H400
    NO_STATUS_TRIGGER = &H20000
    CONTROL_POPUP = &H100000
End Enum
```

See also
eRelationship
DeviceScriptChange
Device Value, String, or Last Change
Device Script Buttons
Device Energy Management
Device Control API (CAPI)
Images

See also
Device Exists, Reference, Address and/or Code
Creating, Deleting, or Accessing Devices
Device Value, String, or Last Change
Device Script Buttons
Device Energy Management
Device Control API (CAPI)
Images

eRelationship

This eEnum is used as the return for the Parent and Child properties of the DeviceClass object, and are as follows:

```
Enum eRelationship As Integer
    Not_Set = 0
    Indeterminate = 1 ' Could not be determined
    Parent_Root = 2
    Standalone = 3
```

See also
eRelationship
DeviceScriptChange
Device Value Status Pairs
Device Value Graphic Pairs
Device Type
Device_Type_String
DeviceScriptChange

This Enum is used when the Device_Type API is set to Script, and the Device_Type type is set to one of the script run values (See `eDeviceType_Script`). This Enum is one of the parameters passed to the script that is run when the device changes, and it indicates what changed to cause the script to be run. The values are:

```
Enum DeviceScriptChange As Integer
  DevValue = 1 ' The device value changed.
  DevString = 2 ' The device string changed.
  Both = 3 ' Both the device value and string changed.
End Enum
```

See also
dvMISC
DeviceScriptChange
Device Value Status Pairs
Device Value Graphic Pairs
Device Type
Device_Type_String

Device Value Status Pairs

Devices hold a value property (double) that represents the status in the device, and a string which can be displayed regardless of the device value. It's possible to assign name->value pairs to a device. When this is done, the list of names is presented to the user in a drop list or some other UI form. Also, all trigger and actions dialogs will present the user with the value options rather than prompting them to enter a number. Strings are not as powerful as value/status pairs for control options on a device, but they are useful when strings are not known at device creation time or are dynamic during the runtime of HomeSeer.

Value/Status pairs can represent a status-only value, a control-only value, or both. An example would be the desire to have the value 100 represent "On" as a status, but a different value such as 200 with the status "Turn It On" for control. This arrangement allows a script or plug-in to trigger on the change of the device value to 200 which indicates a change needs to be made, and then set the value to 100 to indicate that the change is complete.

The sections of this help file under this topic will inform you about the VSPair object type, as well as the HomeSeer script interface commands which allow you to make changes to the value/status pairs.

See also
dvMISC
eRelationship
DeviceScriptChange
Device Value Graphic Pairs
This is the VSPair object, which is used to describe a single value/status relationship or a range of values and associated status relationship. Multiples of these objects can be associated with a device to handle different types of control or status operations. Most of the modification of these pairs is done using the HomeSeer scripting/application interface commands that start with DeviceVSP_.

```vbnet
Public Class VSPair
    Public PairType As VSVGPairType
    Public Render_Location As Enums.CAPIControlLocation
    Public RangeStart As Double
    Public RangeEnd As Double
    Public RangeStatusPrefix As String = ""
    Public RangeStatusSuffix As String = ""
    Public RangeStatusDecimals As Integer = 0
    Public RangeStatusDivisor As Double = 0
    Public IncludeValues As Boolean = True
    Public ValueOffset As Double = 0
    Public HasAdditionalData As Boolean = False
    Public HasScale As Boolean = False
    Public ZeroPadding As Boolean = False
    Public Const ScaleReplace As String = "@S@"
    Public Shared Function AddDataReplace(ByVal Index As Integer) As String
    Public ReadOnly Property ControlStatus As ePairStatusControl
    Public Property Render As Enums.CAPIControlType
    Public WriteOnly Property Value As Double
    Public Property StringList As String
    Public WriteOnly Property StringListAdd As String
    Public Property ControlUse As ePairControlUse
End Class
```

The definition for each member is as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>Structure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PairType</td>
<td></td>
<td>This enum indicates whether the pair represents a single value or a range of values.</td>
</tr>
<tr>
<td>Render_Location</td>
<td>Row</td>
<td>If this is a control pair that is set to be rendered as a button, then set this to the row number to position the button at. Row or Column of 0 results in the button not being drawn, but the control option still exists.</td>
</tr>
<tr>
<td></td>
<td>Column</td>
<td>If this is a control pair that is set to be rendered as a button, then set this to the column number to position the button at. Row or Column of 0 results in the button not being drawn, but the control option still exists.</td>
</tr>
<tr>
<td></td>
<td>ColumnSpan</td>
<td>For more exact positioning of rendered controls on a device, you may indicate that a rendered control is to use multiple columns, thus allowing for alignment options. Note that not all 3rd party User Interfaces will be able to honor Row, Column, and ColumnSpan settings.</td>
</tr>
<tr>
<td>RangeStart</td>
<td></td>
<td>If this VSPair is a range, this contains the lowest value of the range being specified.</td>
</tr>
<tr>
<td>RangeEnd</td>
<td></td>
<td>If this VSPair is a range, this contains the highest value of the range being specified.</td>
</tr>
<tr>
<td>Value</td>
<td></td>
<td>If this VSPair is a single value pair and not a range, then this holds the value that this pair represents.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>RangeStatusPrefix</td>
<td>These contain strings of text to be prepended (prefix) or appended (suffix) to the status string value as it is generated for a range value/status pair. These are not used for single-value pairs. An example of their use is in the range 1 to 99 to represent dim values, the prefix would be set to &quot;Dim &quot; and the suffix to &quot;%&quot;, for a net status string when the value is 49 of &quot;Dim 49%&quot;.</td>
<td></td>
</tr>
<tr>
<td>RangeStatusSuffix</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>When the value/status pair is not a range, this holds the status string to be displayed when the device is at the value set by the Value property.</td>
<td></td>
</tr>
<tr>
<td>RangeStatusDecimals</td>
<td>For range type value/status pairs, you may set this to a value &gt; 0 to have that many decimal places displayed in the value. For example, if the range is 1 to 10, and the RangeStatusDecimals is set to 1, then the full range would encompass values such as: 1.0, 1.1, 1.2, 1.3,... 9.8, 9.9, 10.</td>
<td></td>
</tr>
<tr>
<td>RangeStatusDivider</td>
<td>For range type value/status pairs, it may be inconvenient to modify the value so that it fits a more user-friendly display without messing up what the user has to enter for device value triggers - in that scenario, if you force the value to be 100 to represent 100K, the user may think they can enter 100 for a trigger when they need to enter 100000. To deal with this, set this property to 1000 and HomeSeer will divide the value by 1000 prior to formatting the display status string - the actual value will not be changed.</td>
<td></td>
</tr>
<tr>
<td>IncludeValues</td>
<td>For range value/status pairs, it is sometimes inconvenient having the value as part of the status when it is not indicative of anything meaningful. If IncludeValues is set to False, the status string generated will not include the values. Example: For a device which has an invalid state on values in the range 101 to 254, turn IncludeValues off (set it False) and set your RangeStatusPrefix to &quot;INVALID VALUE&quot;, and that will cause HomeSeer to display &quot;INVALID VALUE&quot; for each of those values without having to use single value, value-status pairs.</td>
<td></td>
</tr>
<tr>
<td>ValueOffset</td>
<td>When it is desirable to have separate status and control value/status pair ranges, this property can be used to facilitate that since two separate status and control pairs cannot be for the same value. To use this, establish one range to use the true values of 1 to 100. Now, establish a second range to use the &quot;fake&quot; values of 101 to 200, but set the ValueOffset to 100, which causes HomeSeer to use a display status string with the value having 100 subtracted from it. For example, if you have a status range pair which creates a status of &quot;Setting Is Currently 50&quot; with a value of 50, you can have a control pair that creates a control option of &quot;Set to 50 Degrees&quot;, which corresponds to the value 150. When a script or plug-in receives notification of the device changing to 150, the appropriate command can be sent to invoke the change to 50, and then the device may be set to 50 to indicate that the change has been made.</td>
<td></td>
</tr>
<tr>
<td>HasScale</td>
<td>At the time a device is created, it may not be known whether its scale is meters or Miles, Fahrenheit or Celsius, or some other set of multiple scales. To help with those situations, set HasScale to True, use the constant ScaleReplace in your range prefix or suffix, and then at runtime when the device is being updated to a new value, set the device ScaleText to your scale (e.g. &quot;degF&quot; or &quot;degC&quot;), set the value, and then when the status is requested, HomeSeer will replace ScaleReplace (@S@) with your ScaleText.</td>
<td></td>
</tr>
<tr>
<td>ScaleReplace</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZeroPadding</td>
<td>When the status is rendered and a RangeStatusDecimals (decimal places) are specified, setting this to True will cause trailing zeros to be padded to the end of the value so that it always displays the number of decimal places desired. For example, with RangeStatusDecimals set to 3, the value 1.5 displays as 1.5 when ZeroPadding is False, and it displays as 1.500 when ZeroPadding is True.</td>
<td></td>
</tr>
<tr>
<td>HasAdditionalData</td>
<td>The usage of these two properties is identical to that of HasScale/ScaleReplace except that you can use a virtually limitless number of replacements in the status string. To use, set HasAdditionalData to True, and then when the device is updated, set the device's AdditionalDisplayData (array of string) to the values that you want replaced. AddDataReplace is a shared (constant) function that can be used to generate a replacement variable for any number - for example, AddDataReplace with an argument value of 4 will return a replacement variable of &quot;@%4%@&quot; (without quotes). If your range prefix or suffix contains this string (which you can use by calling AddDataReplace(4) when you are setting up the prefix or suffix), then that string will be replaced with the 4th value from the array of strings set on the device's AdditionalDisplayData property. Example: pair.HasAdditionalData = True pair.RangeStatusSuffix = AddDataReplace(0) &amp; &quot; of &quot; &amp; AddDataReplace(1) (Device value changes and becomes 5555...)</td>
<td></td>
</tr>
<tr>
<td>AddDataReplace</td>
<td>Dim AddData(1) As String AddData(0) = &quot;Miles&quot; AddData(1) = &quot;Asphalt&quot; dv.AdditionalDisplayData(hs) = AddData hs.SetDeviceValue(MyDevice, 5555) Result: 5555 Miles of Asphalt</td>
<td></td>
</tr>
</tbody>
</table>
ControlStatus

This read-only property allows you to retrieve an Enum value indicating whether the pair is designated as being for status only, control only, or both status and control. Use the hs.DeviceVSP_ChangePair (and other functions) to set or change the ControlStatus.

ControlUse

This property is valid only if the ControlStatus is set to Control or Both, and is used to indicate, most appropriately for lighting control devices, which control pairs are for On, Off, or Dim control of a light. An option exists for On_Alternate so that devices supporting multiple “On” methods may indicate secondary On control values. The “Dim” ControlUse value should be used on any/all pairs which are levels between fully on and completely off.

Render

This property is used to get or set how the control status/value pair is to be rendered when control options are offered by a user interface. See the values under the CAPIControl/CAPIControlType topic.

StringList

StringListAdd

When the UI Render type is set to a drop-down list of strings, set StringList to the array of string values to be displayed. You may add the items one at a time using StringListAdd, or may set them all at once using StringList.

See also
DeviceVSP Methods

VSVGPairType

The VSVGPairType, used with both value/status and value/graphic pairs, is an Enum as follows:

Public Enum VSVGPairType
    SingleValue = 1
    Range = 2
End Enum

See also
ePairStatusControl
ePairControlUse

ePairStatusControl

ePairStatusControl is an Enum used in value/status pairs and has the following values:

Public Enum ePairStatusControl
    Status = 1
    Control = 2
    Both = 3
End Enum

See also
VSVGPairType
ePairControlUse
ePairControlUse

ePairControlUse is an Enum used in value/status pairs and has the following values:

```vbnet
Public Enum ePairControlUse
    Not_Specified = 0
    _On = 1
    _Off = 2
    _Dim = 3
    _On_Alt = 4
End Enum
```

See also
- VSVGPairType
- ePairStatusControl

DeviceVSP Methods

Body of text here

See also
- VSPair

DeviceVSP_AddPair

To add the name->value pair to a device use this function:

```vbnet
Public Function DeviceVSP_AddPair(ByVal dvRef As Integer, ByVal Pair As VSPair) As Boolean
hs.DeviceVSP_AddPair(ref, Pair)
Where:
    ref= device reference #
    Pair = The VSPair object that you want added

You can check the return value (Boolean) to determine if it was successful or not.
```

See also
- DeviceVSP_ChangePair
- DeviceVSP_CountAll
- DeviceVSP_CountStatus
- DeviceVSP_CountControl
- DeviceVSP_ClearAll
- DeviceVSP_ClearAny
- DeviceVSP_ClearStatus
- DeviceVSP_ClearControl
- DeviceVSP_ClearBoth
- DeviceVSP_Get
- DeviceVSP_GetStatus
- DeviceVSP_GetAllStatus
- DeviceVSP_PairsProtected
DeviceVSP_ChangePair

This will change the pair type of an existing value/status pair.

**Public Function** DeviceVSP_ChangePair(ParamName dvRef As Integer, _
ParamName Existing As VSPair, _
ParamName NewType As ePairStatusControl) As Boolean

hs.DeviceVSP_ChangePair(ref, Existing, NewType)

Where:
- **ref** = device reference #
- **Existing** = The current VSPair (value/status pair) object that is set on the device.
- **NewType** = The new ePairStatusControl type (Status, Control, Both) that you want the pair type changed to.

For example, to change the pair type from Status to Both:

hs.DeviceVSP_ChangePair(ref, Pair, ePairStatusControl.Both)

You can check the return value (Boolean) to determine if it was successful or not.

See also
- DeviceVSP_AddPair
- DeviceVSP_CountAll
- DeviceVSP_CountStatus
- DeviceVSP_CountControl
- DeviceVSP_ClearAll
- DeviceVSP_ClearAny
- DeviceVSP_ClearStatus
- DeviceVSP_ClearControl
- DeviceVSP_ClearBoth
- DeviceVSP_Get
- DeviceVSP_GetStatus
- DeviceVSP_GetAllStatus
- DeviceVSP_PairsProtected

DeviceVSP_CountAll

Use this function to get a count of all value/status pairs on a device.

**Public Function** DeviceVSP_CountAll(ParamName dvRef As Integer) As Integer

Count = hs.DeviceVSP_CountAll(ref)

Where:
- **ref** = device reference ID

You can check the return value (Integer) to determine if it was successful or not. Return values less than zero (0) indicate an error condition such as the device reference ID being invalid.

See also
- DeviceVSP_AddPair
- DeviceVSP_ChangePair
- DeviceVSP_CountStatus
- DeviceVSP_CountControl
DeviceVSP_CountStatus

Use this function to get a count of only the status type value/status pairs on a device.

Public Function DeviceVSP_CountStatus(ByVal dvRef As Integer) As Integer
Count = hs.DeviceVSP_CountStatus(ref)

Where:
ref = device reference ID

You can check the return value (Integer) to determine if it was successful or not. Return values less than zero (0) indicate an error condition such as the device reference ID being invalid.

See also
DeviceVSP_AddPair
DeviceVSP_ChangePair
DeviceVSP_CountAll
DeviceVSP_CountControl
DeviceVSP_ClearAll
DeviceVSP_ClearAny
DeviceVSP_ClearStatus
DeviceVSP_ClearControl
DeviceVSP_ClearBoth
DeviceVSP_Get
DeviceVSP_GetStatus
DeviceVSP_GetAllStatus
DeviceVSP_PairsProtected

DeviceVSP_CountControl

Use this function to get a count of all control type value/status pairs on a device.

Public Function DeviceVSP_CountControl(ByVal dvRef As Integer) As Integer
Count = hs.DeviceVSP_CountControl(ref)

Where:
ref = device reference ID

You can check the return value (Integer) to determine if it was successful or not. Return values less than zero (0) indicate an error condition such as the device reference ID being invalid.

See also
DeviceVSP_AddPair
DeviceVSP_ChangePair
DeviceVSP_CountAll
DeviceVSP_CountStatus
DeviceVSP_ClearAll
DeviceVSP_ClearAny
DeviceVSP_ClearStatus
DeviceVSP_ClearControl
DeviceVSP_ClearBoth
DeviceVSP_Get
DeviceVSP_GetStatus
DeviceVSP_GetAllStatus
DeviceVSP_PairsProtected

DeviceVSP_ClearAll

Use this function to CLEAR all value/status pairs from a device.

**Public Sub** DeviceVSP_ClearAll(**ByVal** dvRef As Integer, **ByVal** TrueConfirm As Boolean)

hs.DeviceVSP_ClearAll(ref, True)

**Where:**

- **ref** = device reference ID
- **True** = The constant True or a variable indicating True must be passed as the second parameter as confirmation that you wish this to take place.

See also
DeviceVSP_AddPair
DeviceVSP_ChangePair
DeviceVSP_CountAll
DeviceVSP_CountStatus
DeviceVSP_CountControl
DeviceVSP_ClearAny
DeviceVSP_ClearStatus
DeviceVSP_ClearControl
DeviceVSP_ClearBoth
DeviceVSP_Get
DeviceVSP_GetStatus
DeviceVSP_GetAllStatus
DeviceVSP_PairsProtected

DeviceVSP_ClearAny

This will clear any (control, status, or both) value/status pair out of the device that matches the given value parameter.

**Public Function** DeviceVSP_ClearAny(**ByVal** dvRef As Integer, **ByVal** Value As Double) As Boolean

Success = hs.DeviceVSP_ClearAny(ref, Value)

**Where:**

- **ref** = device reference #
- **Value** = The value of the value/status pair you wish removed.

You can check the return value (Boolean) to determine if it was successful or not.

See also
DeviceVSP_AddPair
DeviceVSP_ChangePair
DeviceVSP_CountAll
DeviceVSP_CountStatus
DeviceVSP_ClearStatus

This will clear any status type value/status pair out of the device that matches the given value parameter.

**Public Function** DeviceVSP_ClearStatus(ByVal dvRef As Integer, ByVal Value As Double) As Boolean

Success = hs.DeviceVSP_ClearStatus(ref, Value)

**Where:**

- **ref** = device reference #
- **Value** = The value of the status type value/status pair you wish removed.

You can check the return value (Boolean) to determine if it was successful or not.

See also

DeviceVSP_AddPair
DeviceVSP_ChangePair
DeviceVSP_CountAll
DeviceVSP_CountStatus
DeviceVSP_CountControl
DeviceVSP_ClearAll
DeviceVSP_ClearStatus
DeviceVSP_ClearControl
DeviceVSP_ClearBoth
DeviceVSP_Get
DeviceVSP_GetStatus
DeviceVSP_GetAllStatus
DeviceVSP_PairsProtected

DeviceVSP_ClearControl

This will clear any control type value/status pair out of the device that matches the given value parameter.

**Public Function** DeviceVSP_ClearControl(ByVal dvRef As Integer, ByVal Value As Double) As Boolean

Success = hs.DeviceVSP_ClearControl(ref, Value)

**Where:**

- **ref** = device reference #
- **Value** = The value of the control type value/status pair you wish removed.

You can check the return value (Boolean) to determine if it was successful or not.

See also

DeviceVSP_AddPair
DeviceVSP_ChangePair
DeviceVSP_CountAll
DeviceVSP_CountStatus
DeviceVSP_CountControl
DeviceVSP_ClearAll
DeviceVSP_ClearStatus
DeviceVSP_ClearControl
DeviceVSP_ClearBoth
DeviceVSP_Get
DeviceVSP_GetStatus
DeviceVSP_GetAllStatus
DeviceVSP_PairsProtected
DeviceVSP_ClearBoth

This will clear any "Both" type value/status pair out of the device that matches the given value parameter.

Public Function DeviceVSP_ClearBoth(ByVal dvRef As Integer, ByVal Value As Double) As Boolean

Success = hs.DeviceVSP_ClearBoth(ref, Value)

Where:
ref = device reference #
Value = The value of the "Both" (status and control) type value/status pair you wish removed.

You can check the return value (Boolean) to determine if it was successful or not.

See also
DeviceVSP_AddPair
DeviceVSP_ChangePair
DeviceVSP_CountAll
DeviceVSP_CountStatus
DeviceVSP_CountControl
DeviceVSP_ClearAll
DeviceVSP_ClearAny
DeviceVSP_ClearStatus
DeviceVSP_ClearControl
DeviceVSP_Get
DeviceVSP_GetStatus
DeviceVSP_GetAllStatus
DeviceVSP_PairsProtected

DeviceVSP_Get

This will retrieve a value/status pair object (VSPair) from a device if it matches the value and type provided.

Public Function DeviceVSP_Get(ByVal dvRef As Integer, _
ByVal Value As Double, _
ByVal VSPType As ePairStatusControl) As VSPair

MyPair = hs.DeviceVSP_Get(ref, Value, VSPType)
If MyPair Is Nothing Then
    hs.WriteLog("Error","Could not find the value/status pair for the value " & Value.ToString & " on the device " & hs.DeviceName(ref))
    Exit Sub
End If

Where:
ref = device reference #
Value = The value of the value/status pair you are looking for (use the starting value of the range to retrieve a range type pair).
VSPType = The ePairStatusControl type (Status, Control, Both) that you are looking for.

You can check the return value (VSPair object) to determine if it was successful or not. If the
DeviceVSP_GetStatus

This will retrieve a status string given a specific value.

Public Function DeviceVSP_GetStatus(ByVal dvRef As Integer, _
    ByVal Value As Double, _
    ByVal VSPType As ePairStatusControl) As String

MyStatus = hs.DeviceVSP_GetStatus(ref, Value, VSPType)

Where:

    ref = device reference #
    Value = The value of the value/status pair you are looking for (use any value of the range to retrieve a range type status).
    VSPType = The ePairStatusControl type (Status, Control, Both) that you are looking for. If the type is Status or Control, the string returned will be the formatted status or control string for the value given. If the type is Both, then the string returned will be in the form: Status: (text), Control: (text)

DeviceVSP_GetAllStatus

This will retrieve all value/status pair objects (VSPair) from a device which are defined as STATUS or BOTH.

Public Function DeviceVSP_GetAllStatus(ByVal dvRef As Integer) As VSPair()

MyPair = hs.DeviceVSP_GetAllStatus(ref)

See also
DeviceVSP_AddPair
DeviceVSP_ChangePair
DeviceVSP_CountAll
DeviceVSP_CountStatus
DeviceVSP_CountControl
DeviceVSP_ClearAll
DeviceVSP_ClearAny
DeviceVSP_ClearStatus
DeviceVSP_ClearControl
DeviceVSP_ClearBoth
DeviceVSP_GetStatus
DeviceVSP_GetAllStatus
DeviceVSP_PairsProtected
If MyPair Is Nothing OrElse MyPair.Length < 1 Then
    hs.WriteLog("Error","No pairs are assigned to device * & hsDeviceName(ref))
    Exit Sub
End If
For Each P As VSPair In MyPair
    ...
Next

Where:

ref= device reference #

You can check the return value (VSPair object array) to determine if it was successful or not. If the returned object = Nothing or has a count of zero then there are no status pairs (or both status and control pairs) on the device.

See also
DeviceVSP_AddPair
DeviceVSP_ChangePair
DeviceVSP_CountAll
DeviceVSP_CountStatus
DeviceVSP_CountControl
DeviceVSP_ClearAll
DeviceVSP_ClearAny
DeviceVSP_ClearStatus
DeviceVSP_ClearControl
DeviceVSP_ClearBoth
DeviceVSP_Get
DeviceVSP_GetStatus
DeviceVSP_PairsProtected

DeviceVSP_PairsProtected

To check to see whether the device Value/Status pairs are protected, which applies to devices owned by plugins, use this function:

Function DeviceVSP_PairsProtected(ByVal dvRef As Integer) As Boolean

PStatus = hs.DeviceVSP_PairsProtected(ref)

Where:

ref= device reference #

You can check the return value (Boolean) to determine if the pairs are protected from editing in the HomeSeer UI by the user.

See also
DeviceVSP_AddPair
DeviceVSP_ChangePair
DeviceVSP_CountAll
DeviceVSP_CountStatus
DeviceVSP_CountControl
DeviceVSP_ClearAll
DeviceVSP_ClearAny
DeviceVSP_ClearStatus
DeviceVSP_ClearControl
DeviceVSP_ClearBoth
DeviceVSP_Get
DeviceVSP_GetStatus
DeviceVSP_GetAllStatus
Device Value Graphic Pairs

Devices hold a value property (float) that is normally used to hold the dim level of X10 devices. It may be desirable to use the value to represent status in the device. It's possible to assign graphic->value pairs to a device. When this is done, the status as displayed for the device will include the graphic matching the device's current value.

To assign the graphic->value pairs to a device, use the function call `hs.DeviceVGP_AddPair`. To use the function, call:

```
hs.DeviceVGP_AddPair(dvRef, Pair)
```

Where:

- `dvRef` = device reference # to set values to
- `Pair` = graphic value pairs formatted in the class VGPair

Note

- The setting is saved in the configuration database so the call only needs to be made during some initialization.

See also

dvMISC
eRelationship
deviceScriptChange
deviceValueStatusPairs
deviceType

device_Type_String

VGPair

This is the VGPair object, which is used to describe a single value/graphic relationship or a range of values and associated graphics relationship. Multiples of these objects can be associated with a device to handle different types of graphics to represent different states (values) of a device. Most of the modification of these pairs is done using the HomeSeer scripting/application interface commands that start with DeviceVGP_

```
Public Class VGPair

    Public PairType As SSVGPairType
    Public RangeStart As Double
    Public RangeEnd As Double
    Public WriteOnly Property Graphic As String
    Public ReadOnly Property Value As Double
    Public WriteOnly Property Set_Value As Double

End Class
```

The definition for each member is as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PairType</td>
<td>This enum indicates whether the pair represents a single value or a range of values.</td>
</tr>
<tr>
<td>RangeStart</td>
<td>If this VGPair is a range, this contains the lowest value of the range being specified.</td>
</tr>
<tr>
<td>RangeEnd</td>
<td>If this VGPair is a range, this contains the highest value of the range being specified.</td>
</tr>
<tr>
<td>Value (Read Only)</td>
<td>If this VGPair is a single value pair and not a range, then this holds the value that this pair represents.</td>
</tr>
<tr>
<td>Set_Value (Write Only)</td>
<td>This contains the path, relative to the HomeSeer HTML directory or absolute if outside the HTML directory.</td>
</tr>
</tbody>
</table>

To add a graphics pair to a device:
GPair = New VGPair
GPair.PairType = VSVGPairType.SingleValue
GPair.Set_Value = 0
GPair.Graphic = "~/images/HomeSeer/status/off.gif"
hs.DeviceVGP_AddPair(ref, GPair)

See also
DeviceVGP Methods

VSVGPairType

The VSVGPairType, used with both value/status and value/graphic pairs, is an Enum as follows:

Public Enum VSVGPairType
    SingleValue = 1
    Range = 2
End Enum

See also

DeviceVGP Methods

DeviceVGP_AddPair

To add the graphic->value pair to a device use this function:

Public Function DeviceVGP_AddPair(ByVal dvRef As Integer, ByVal Pair As VGPair) As Boolean
hs.DeviceVGP_AddPair(ref, Pair)

Where:
   ref = device reference #
   Pair = The VGPair object that you want added

You can check the return value (Boolean) to determine if it was successful or not.
DeviceVGP_Count

Use this function to get a count of all value/graphic pairs on a device.

```vbnet
Public Function DeviceVGP_Count(ByVal dvRef As Integer) As Integer
    Count = hs.DeviceVGP_Count(ref)
End Function
```

Where:
- `ref` = device reference ID

You can check the return value (Integer) to determine if it was successful or not. Return values less than zero (0) indicate an error condition such as the device reference ID being invalid.

DeviceVGP_ClearAll

Use this function to CLEAR all value/graphic pairs from a device.

```vbnet
Public Sub DeviceVGP_ClearAll(ByVal dvRef As Integer, ByVal TrueConfirm As Boolean)
    hs.DeviceVGP_ClearAll(ref, True)
End Sub
```

Where:
- `ref` = device reference ID
- `True` = The constant True or a variable indicating True must be passed as the second parameter as confirmation that you wish this to take place.
DeviceVGP_Clear

This will clear any value/graphic pair out of the device that matches the given value parameter.

**Public Function** DeviceVGP_Clear(BYVal dvRef As Integer, BYVal Value As Double) As Boolean

Success = hs.DeviceVGP_Clear(ref, Value)

**Where:**

- ref = device reference #
- Value = The value of the value/graphic pair you wish removed.

You can check the return value (Boolean) to determine if it was successful or not.

See also:

- DeviceVGP_AddPair
- DeviceVGP_Count
- DeviceVGP_ClearAll
- DeviceVGP_Get
- DeviceVGP_GetGraphic
- DeviceVGP_PairsProtected

DeviceVGP_Get

This will retrieve a value/graphic pair object (VGPair) from a device if it matches the value provided.

**Public Function** DeviceVGP_Get(BYVal dvRef As Integer, _
ByVal Value As Double) As VGPair

MyPair = hs.DeviceVGP_Get(ref, Value)
If MyPair Is Nothing Then
    hs.WriteLog("Error","Could not find the value/graphic pair for the value " & Value.ToString & " on the device " & hs.DeviceName(ref))
    Exit Sub
End If

**Where:**

- ref = device reference #
- Value = The value of the value/graphic pair you are looking for (use the starting value of the range to retrieve a range type pair).

You can check the return value (VSPair object) to determine if it was successful or not. If the returned object = Nothing, then the pair matching the provided parameters was not found.

See also:

- DeviceVGP_AddPair
- DeviceVGP_Count
- DeviceVGP_ClearAll
- DeviceVGP_Clear
- DeviceVGP_Get
- DeviceVGP_PairsProtected

DeviceVGP_GetGraphic

This will retrieve a graphic path given a specific value.
**Public Function** DeviceVGP_GetGraphic(ByVal dvRef As Integer, ByVal Value As Double) As String

MyGraphic = hs.DeviceVGP_GetGraphic(ref, Value)

Where:

- **ref** = device reference #
- **Value** = The value of the value/graphic pair you are looking for (use any value of the range to retrieve a range type graphic).

**See also**
DeviceVGP_AddPair
DeviceVGP_Count
DeviceVGP_ClearAll
DeviceVGP_Clear
DeviceVGP_Get
DeviceVGP_PairsProtected

---

**DeviceVGP_PairsProtected**

To check to see whether the device Value/Graphic pairs are protected, which applies to devices owned by plugins, use this function:

**Function** DeviceVGP_PairsProtected(ByVal dvRef As Integer) As Boolean

PStatus = hs.DeviceVGP_PairsProtected(ref)

Where:

- **ref** = device reference #

You can check the return value (Boolean) to determine if the pairs are protected from editing in the HomeSeer UI by the user.

**See also**
DeviceVGP_AddPair
DeviceVGP_Count
DeviceVGP_ClearAll
DeviceVGP_Clear
DeviceVGP_Get
DeviceVGP_GetGraphic

---

**Device Type**

In previous versions of HomeSeer, the device type was a string property and a value that was used to describe the capabilities of the device. The string value was used to find a specific set of device capabilities, and HomeSeer would create a new device type (with a numerical suffix at the end) whenever it discovered that the device was modified from what the device type said it should have as capabilities.

In HomeSeer HS3, the high level meaning is very similar, but the functionality is different enough that NOTHING in reference to the previous versions should be re-used.

The Device Class object has two properties pertaining to device type descriptions:
Device_Type_String
Device_Type_Get (Read Only)
Device_Type_Set (Write Only)
Device_Type_String is a string value which has absolutely no bearing on functionality of the device. This string is what is displayed for the device type if that column of information is enabled on the device utility page. The value is accessed through the plug-in interface, which is a one-way interface requiring the passing of the HomeSeer application interface object when it is changed or when the most current value is retrieved.

DeviceType is an object (DeviceTypeInfo) with properties and procedures. The information stored is used in several of the APIs that HomeSeer supports to describe the device's role in the API. The DeviceType consists of three high-level items: The API designation, the device type, and the device sub-type. These (and the lower-level informational items) are described in the sub-topics to this entry.

See also
dvMISC
eRelationship
DeviceScriptChange
Device Value Status Pairs
Device Value Graphic Pairs
Device_Type_String

DeviceTypeInfo Object

The DeviceTypeInfo object (DeviceTypeInfo) is accessed directly from the device class object using the _Get and _Set properties. When a change is made to the DeviceTypeInfo object, _Set must be called to post the change to the device, and then hs.SaveEventsDevices should be called to force HomeSeer to save the device change.

The prototype for the class is:

```vbnet
Public Class DeviceTypeInfo
   
   Public Property Device_API As eDeviceAPI
   Public ReadOnly Property Device_API_Description As String
   Public Property Device_Type As Integer
   Public ReadOnly Property Device_Type_Description As String
   Public Property Device_SubType As Integer
   Public Property Device_SubType_Description As String

End Class
```

Example:

Retrieve the DeviceTypeInfo object:

```vbnet
Dim DT as DeviceAPI.DeviceTypeInfo = Nothing
'D the device has a reference ID of 1234
dv = hs.GetDeviceByRef(1234)
DT = dv.DeviceType_Get(hs)
If DT IsNot Nothing Then
   ...
End If
```

Change the DeviceTypeInfo and Save the change:

```vbnet
Dim DT As DeviceAPI.DeviceTypeInfo = Nothing
Dim dv As Scheduler.Classes.DeviceClass = Nothing
' the device has a reference ID of 1234
dv = hs.GetDeviceByRef(1234)
If dv Is Nothing Then .... (log an error and exit the procedure)
   DT = dv.DeviceType_Get(hs)
   DT.Device_API = DeviceAPI.DeviceTypeInfo.eDeviceAPI.Plug_In
   dv.DeviceType_Set(hs) = DT
   hs.SaveEventsDevices
End If
```

See also
Device_API

The Device_API property is an enum denoting the type of API, if any, that this device is a part of:

Public Property Device_API As DeviceAPI.DeviceTypeInfo.eDeviceAPI

The Device_API should be set appropriately if the device is a part of an API, set to Plug_In if it is not a part of an API but is owned by a Plug-In, or set to No_API if it is not owned by a plug-in and is not a part of an API.

Example:

Dim DT As New DeviceAPI.DeviceTypeInfo
DT.Device_API = DeviceAPI.DeviceTypeInfo.eDeviceAPI.Plug_In
DT.Device_Type = CInt(ntype)
dv.DeviceType_Set(hs) = DT
hs.SaveEventsDevices

See also
Device_API_Description (Read Only)
Device_Type
Device_Type_Description (Read Only)
Device_SubType
Device_SubType_Description

eDeviceAPI

<Serializable()> _
   Public Enum eDeviceAPI
      No_API = 0 ' All other devices.
      Plug_In = 4 ' Device is owned/managed by a plug-in.
      Thermostat = 16 ' Device is owned/managed by a plug-in and is a thermostat device.
      Media = 32 ' Device is owned/managed by a plug-in and is a media player device.
      Security = 8 ' Device is owned/managed by a plug-in and is a security device.
      SourceSwitch = 64 ' Device is owned/managed by a plug-in and is a matrix switch device.
      Script = 128 ' Device launches a script when the value and/or string changes.
   End Enum

See also

Device_API_Description (Read Only)

Device_API_Description (Read Only)

The Device_API_Description read-only property is a string denoting the type of API, if any, that this device is a part of:

Public ReadOnly Property Device_API_Description As String

The Device_API determines what is returned by this property.

Example:

hs.WriteLog("Info", hs.DeviceName(dv.Ref) & " has a Device Type API of ", _
              & dv.DeviceType_Get(hs).Device_API_Description)
Device_Type

The Device_Type property is an integer denoting the device type of an API, if any, that this device is a part of:

**Public Property** Device_Type As Integer

The Device_API should be set appropriately if the device is a part of an API, set to Plug_In if it is not a part of an API but is owned by a Plug-In, or set to No_API if it is not owned by a plug-in and is not a part of an API. If the Device_API is set to an API type such as Thermostat or Music, then the Device_Type should be set to one of the API types for those APIs. (See DeviceTypeInfo Enums)

Example:
(This sets the device to the thermostat API type "Mode Set")

```vbnet
Dim DT As New DeviceAPI.DeviceTypeInfo
DT.Device_API = DeviceAPI.DeviceTypeInfo.eDeviceAPI.Thermostat
DT.Device_Type = DeviceAPI.DeviceTypeInfo.eDeviceType_Thermostat.Mode_Set
dv.DeviceType_Set(hs) = DT
hs.SaveEventsDevices
```

Example:
(This sets the device to a plug-in custom type.)

```vbnet
Dim DT As New DeviceAPI.DeviceTypeInfo
DT.Device_API = DeviceAPI.DeviceTypeInfo.eDeviceAPI.Plug_In
DT.Device_Type = CInt(PlugDeviceType_X)
DT.Device_SubType = 4
dv.DeviceType_Set(hs) = DT
hs.SaveEventsDevices
```

See also
Device_API
Device_API_Description (Read Only)
Device_Type
Device_Type_Description (Read Only)
Device_SubType
Device_SubType_Description

---

**eDeviceType_GenericRoot**

**Purpose**

The eDeviceType_GenericRoot is not a device type like the other device types - it is a constant value integer (Value = 999) which is to be used when a device is to be a root device for a parent/child relationship, and does not fit any other API specific model.

**Parameters**

Parameter: (Value)
Type: **Integer**
Description: The value of this device type is 999.

See also
- eDeviceType_Media
- eDeviceType_Plugin
- eDeviceType_Script
- eDeviceType_Security
- eDeviceType_SourceSwitch
- eDeviceType_Thermostat

### eDeviceType_Media

**Public Enum** eDeviceType_Media

- **Player_Status** = 1
- **Player_Status_Additional** = 2
- **Player_Control** = 3
- **Player_Volume** = 4
- **Player_Shuffle** = 5
- **Player_Repeat** = 6
- **Music_Genre** = 7
- **Music_Album** = 8
- **Music_Artist** = 9
- **Music_Track** = 10
- **Music_Playlist** = 11
- **Media_Type** = 12
- **Music_Selector_Control** = 20 ' Used to track which instance of MusicAPI and selection mode (e.g. album, artists, playlists)
- **Root** = 99 ' Indicates a root device of a root/child grouping.

End Enum

See also
- eDeviceType_GenericRoot
- eDeviceType_Plugin
- eDeviceType_Script
- eDeviceType_Security
- eDeviceType_SourceSwitch
- eDeviceType_Thermostat

### eDeviceType_Plugin

The Plug-In device type indicates a device type that does NOT fit any of the API specific device types, but is a device type owned by a plug-in. The only defined Enum value is for indicating a Root device in a Parent(Root)/Child relationship.

**Public Enum** eDeviceType_Plugin

- **Root** = 99 ' Indicates a root device of a root/child grouping.

End Enum

See also
- eDeviceType_GenericRoot
- eDeviceType_Plugin
- eDeviceType_Script
- eDeviceType_Security
- eDeviceType_SourceSwitch
eDeviceType_Script

Public Enum eDeviceType_Script
    Disabled = 0 ' Set the device type to this to temporarily stop scripts from being run.
    Run_On_Any_Change = 1 ' Set to this type to run the script on a value or string change.
    Run_On_Value_Change = 2 ' Set to this type to run the script when the value changes.
    Run_On_String_Change = 3 ' Set to this type to run the script when the string changes.
End Enum

See also
  eDeviceType_GenericRoot
  eDeviceType_Media
  eDeviceType_Plugin
  eDeviceType_Security
  eDeviceType_SourceSwitch
  eDeviceType_Thermostat

eDeviceType_Security

Public Enum eDeviceType_Security
    Alarm = 1 ' Alarm status & control (shows alarms that have occurred and can also invoke an alarm - e.g. Duress)
    Arming = 10 ' Arming status & control (shows the state of the security arming and can set arming state)
    Keypad = 20 ' Keypad status & control
    Zone_Perimeter = 30 ' A perimeter zone
    Zone_Perimeter_Delay = 31 ' A perimeter zone with a violation alarm delay
    Zone_Interior = 32 ' An interior zone (not normally armed in stay mode)
    Zone_Interior_Delay = 33 ' An interior zone (with a violation alarm delay when armed)
    Zone_UpperAuxiliary = 34 ' A aux zone, not usually included in any arming mode
    Zone_Other = 35 ' A zone that does not fit any other zone description
    Zone_Safety_Smoke = 40 ' A smoke detector zone (not allowed to be bypassed)
    Zone_Safety_CO = 41 ' A Carbon Monoxide zone (not allowed to be bypassed)
    Zone_Safety_CO2 = 42 ' A Carbon Dioxide zone (not allowed to be bypassed)
    Zone_Safety_Other = 43 ' A zone for some other safety sensor that cannot be bypassed
    Output_Relay = 50 ' A general purpose output relay
    Output_Other = 51 ' A general purpose output (could be virtual as in a 'flag' output)
    Communicator = 60 ' Communicator status and (if available) control
    Siren = 70 ' Siren output - status usually - control follows alarm state.
    Root = 99 ' Indicates a root device of a root/child grouping.
End Enum

See also
  eDeviceType_GenericRoot
  eDeviceType_Media
  eDeviceType_Plugin
  eDeviceType_Script
  eDeviceType_SourceSwitch
  eDeviceType_Thermostat
eDeviceType_SourceSwitch

Public Enum eDeviceType_SourceSwitch
    Invalid = 0
    System = 1
    Source = 10
    Source_Extended = 15
    Zone = 20
    Zone_Extended = 25
    Root = 99
End Enum

See also
eDeviceType_GenericRoot
eDeviceType_Media
eDeviceType_Plugin
eDeviceType_Script
eDeviceType_Security
eDeviceType_Thermostat

eDeviceType_Thermostat

Public Enum eDeviceType_Thermostat
    Operating_State = 1
    Temperature = 2
    Mode_Set = 3
    Fan_Mode_Set = 4
    Fan_Status = 5
    Setpoint = 6
    RunTime = 7
    Hold_Mode = 8
    Operating_Mode = 9
    Additional_Temperature = 10
    Setback = 11
    Filter_Remind = 12
    Root = 99
End Enum

See also
eDeviceType_GenericRoot
eDeviceType_Media
eDeviceType_Plugin
eDeviceType_Script
eDeviceType_Security
eDeviceType_SourceSwitch

Device_Type_Description (Read Only)

The Device_Type_Description read-only property is a string denoting the type of the device:
Public ReadOnly Property Device_Type_Description As String

The Device_API also determines what is returned by this property, as the value of the Device_Type is influenced by the API that the device subscribes to; if the API is the Thermostat API, then the device types are expected to be one of the eDeviceType_Thermostat enum values.

Example:

```vbs
hs.WriteLog("Info", hs.DeviceName(dv.Ref) & " has a Device Type of " & dv.DeviceType_Get(hs).Device_Type_Description)
```

See also
Device_API
Device_API_Description (Read Only)
Device_Type
Device_SubType
Device_SubType_Description

Device_SubType

The Device_SubType property is an integer denoting the device sub-type, if any, that this device is a part of:

Public Property Device_SubType As Integer

The Device_API should be set appropriately if the device is a part of an API, set to Plug_In if it is not a part of an API but is owned by a Plug-In, or set to No_API if it is not owned by a plug-in and is not a part of an API. If the Device_API is set to an API type such as Thermostat or Music, then the Device_Type should be set to one of the API types for those APIs. (See DeviceTypeInfo Enums) and this property can be used to denote the device type further.

**NOTE:** When the API is Thermostat, and the Device_Type is Setpoint, it is required that the Device_SubType be set to indicate which setpoint the device is representing, as found in the enum eDeviceSubType_Setpoint

Example:
(This sets the device to the Thermostat API type "Setpoint" for Cooling)

```vbs
Dim DT As New DeviceAPI.DeviceTypeInfo
DT.Device_API = DeviceAPI.DeviceTypeInfo.eDeviceAPI.Thermostat
DT.Device_Type = DeviceAPI.DeviceTypeInfo.eDeviceType_Thermostat.Setpoint
DT.Device_SubType = DeviceAPI.DeviceTypeInfo.eDeviceSubType_Setpoint.Cooling_1
DT.Device_SubType_Description = "Cool Setpoint"
dv.DeviceType_Set(hs) = DT
hs.SaveEventsDevices
```

Example:
(This sets the device to a plug-in custom type.)

```vbs
Dim DT As New DeviceAPI.DeviceTypeInfo
DT.Device_API = DeviceAPI.DeviceTypeInfo.eDeviceAPI.Plug_In
DT.Device_Type = CInt(PlugDeviceType_X)
DT.Device_SubType = 4
dv.DeviceType_Set(hs) = DT
hs.SaveEventsDevices
```

See also
Device_API
Device_API_Description (Read Only)
Device_Type
Device_Type_Description (Read Only)
Device_SubType_Description
eDeviceSubType_SecurityArea

When the Device_Type is set to Security, and the security panel uses partitions/areas, the Device_SubType should be used to indicate the area number that the device belongs to IN ADDITION to there being a separate root/child device hierarchy per area/partition. When areas/partitions are NOT used, the Device_SubType can be any integer value, but to avoid misinterpretation, it is suggested that values below 20 not be used.

```csharp
<Serializable()> _
Public Enum eDeviceSubType_SecurityArea
    Invalid = 0
    PRIMARY = 1
    Area_Partition_2 = 2
    Area_Partition_3 = 3
    Area_Partition_4 = 4
    Area_Partition_5 = 5
    Area_Partition_6 = 6
    Area_Partition_7 = 7
    Area_Partition_8 = 8
    Area_Partition_9 = 9
End Enum
```

See also
eDeviceSubType_Setpoint

eDeviceSubType_Setpoint

When the Device_Type is set to a Thermostat API Type of Setpoint, the Device_SubType should be set to one of the enum values from this list.

```csharp
<Serializable()> _
Public Enum eDeviceSubType_Setpoint
    Invalid = 0
    Heating_1 = 1
    Cooling_1 = 2
    Furnace = 7
    Dry_Air = 8
    Moist_Air = 9
    Auto_Changeover = 10
    Energy_Save_Heat = 11
    Energy_Save_Cool = 12
    Away_Heating = 13
End Enum
```

See also
eDeviceSubType_SecurityArea

Device_SubType_Description

The Device_SubType_Description property is a string denoting the device sub-type, if any, that this device is a part of. The string is not used and is only for reference/description to the user.

```csharp
Public Property Device_SubType_Description As String
```
This property is a description to go with the Device_SubType property to provide a meaningful description to the user of the device subtype.

Example:
(This sets the device to the thermostat API type "Setpoint")

```vba
Dim DT As New DeviceAPI.DeviceTypeInfo
DT.Device_API = DeviceAPI.DeviceTypeInfo.eDeviceAPI.Thermostat
DT.Device_Type = DeviceAPI.DeviceTypeInfo.eDeviceType_Thermostat.Setpoint
DT.Device_SubType = CInt(SPType)
DT.Device_SubType_Description = "Cool Setpoint"
dv.DeviceType_Set(hs) = DT
hs.SaveEventsDevices
```

Example:  
(This sets the device to a plug-in custom type.)

```vba
Dim DT As New DeviceAPI.DeviceTypeInfo
DT.Device_API = DeviceAPI.DeviceTypeInfo.eDeviceAPI.Plug_In
DT.Device_Type = CInt(PlugDeviceType_X)
DT.Device_SubType = 4
DT.Device_SubType_Description = "Bazinga!"
dv.DeviceType_Set(hs) = DT
hs.SaveEventsDevices
```

See also

- Device_API
- Device_API_Description (Read Only)
- Device_Type
- Device_Type_Description (Read Only)
- Device_SubType

---

**Device_Type_String**

Device_Type_String is a simple string description of the device type and has no actual bearing on the device as seen by HomeSeer or other plug-ins - only the DeviceType is used by HomeSeer and other plug-ins.

Example (Read):
If dv IsNot Nothing Then
    Log("My device has a device type of: " & dv.Device_Type_String(hs), LogType.Info)
End If

Example (Write):
If dv IsNot Nothing Then
    dv.Device_Type_String(hs) = "Joe Bazooka Bubble Gum"
End If

See also

- dvMISC
- eRelationship
- DeviceScriptChange
- Device Value Status Pairs
- Device Value Graphic Pairs
- Device Type

---

**Device Exists, Reference, Address and/or Code**

In This Section
DeviceExistsRef

**Purpose**
This function indicates if the device does or does not exist.

**Parameters**
- **Parameter:** device
  - **Type:** integer
  - **Description:** This is the device reference ID number.

**Returns**
- **Return value:** status
  - **Type:** Boolean
  - **Description:** Returns False if the device does not exist.

See also
DeviceExistsAddress
DeviceExistsAddressFull
DeviceExistsCode
GetDeviceRef
GetDeviceRefByName
GetDeviceParentRefByRef
GetDeviceCode

DeviceExistsAddress

**Purpose**
This function indicates if the device does or does not exist using its Address property (See also DeviceExistsCode).

**Parameters**
- **Parameter:** Address
  - **Type:** string
  - **Description:** This is the device address, such as "Unit1", "0F47ED78-2", or "U2-I45-K2.2"
Parameter: **CaseSensitive**  
Type: **Boolean**  
Description: When True, the address must match exactly. When false, the address match is case insensitive such that apple2=APPLE2

**Returns**

Return value: **status**  
Type: **long (.NET Integer)**  
Description: Returns -1 if the device does not exist, otherwise it returns the device reference ID number of the device. The reference number can then be used with the GetDeviceByRef function.

**Note**

The address field can contain any string of characters. The format and value is determined by a plug-in in the event that the device is owned by a plug-in.

When retrieved, the Address property includes the Code property, separated by a dash (-) if the Code property is set. For example, if the Address was set to "Unit1" and the code field is not used, then retrieving the Address field will result in "Unit1". If the Address was set to "Unit1" and the code was set to "Y55", then retrieving the Address field will result in "Unit1-Y55".

When this function is used, the Code field is NOT combined with the Address field. (See DeviceExistsAddressFull to find a device using its full address-code value.)

**See also**

DeviceExistsRef  
DeviceExistsAddressFull  
DeviceExistsCode  
GetDeviceRef  
GetDeviceRefByName  
GetDeviceParentRefByRef  
GetDeviceCode
When retrieved, the Address property includes the Code property, separated by a dash (-) if the Code property is set. For example, if the Address was set to "Unit1" and the code field is not used, then retrieving the Address field will result in "Unit1". If the Address was set to "Unit1" and the code was set to "Y55", then retrieving the Address field will result in "Unit1-Y55".

When this function is used, the Code field is combined with the Address field. (See DeviceExistsAddressl to find a device using its address value only.)

See also
DeviceExistsRef
DeviceExistsAddress
DeviceExistsAddressFull
GetDeviceRef
GetDeviceRefByAddress
GetDeviceParentRefByRef
GetDeviceCode

DeviceExistsCode

Purpose
This function indicates if the device does or does not exist using its Code property which is in the letter code and unit code format.

Parameters
Parameter: Code
Type: String
Description: This is the house/letter code and unit code of the device, such as "A1" or "q17".

Returns
Return value: status
Type: Integer
Description: Returns -1 if the device does not exist, otherwise it returns the device reference ID number of the device. The reference number can then be used with the GetDeviceByRef function.

See also
DeviceExistsRef
DeviceExistsAddress
DeviceExistsAddressFull
GetDeviceRef
GetDeviceRefByAddress
GetDeviceParentRefByRef
GetDeviceCode

GetDeviceRef

Purpose
This function returns the device reference for a device. The device reference is different than an index to a device. The device reference is only needed for other procedures which explicitly require the device reference.

Parameters
Parameter: sAddress
Type: string

This will only return the reference to the first device matching the address provided. The address is the "Address" property of a device, not the code property. The Address is normally set by a plugin and can be used to find a device.
GetDeviceRefByName

Purpose

This function returns the device reference for a device. The device reference is different than an index to a device. The device reference is only needed for other procedures which explicitly require the device reference.

- This will only return the reference to the first device matching the name provided.

Parameters

Parameter: sName
Type: string
Description: This is the device name including the location, such as "Family Room Lamp".

Returns

Return value: reference
Type: Integer
Description: This is a numerical device reference.

Example

Sub Main()

    Dim dvRef
    Dim dv

    dvRef = hs.GetDeviceRefByName("Family Room Light")
    if dvRef > 0 then
        Set dv = hs.GetDeviceByRef(dvRef)
       else
        hs.WriteLog "Error","Could not find the reference for the device specified."
        exit Sub
    end if
    hs.WriteLog "Info","The address for the device is " & dv.hc & dv.dc

End Sub

See also

DeviceExistsRef
DeviceExistsAddress
DeviceExistsAddressFull
DeviceExistsCode
GetDeviceRefByName
GetDeviceParentRefByRef
GetDeviceCode
GetDeviceParentRefByRef

**Purpose**

This function returns a reference to a given device's parent device. If the device does not exist, then 0 will be returned. If the device reference number provided does not belong to a device, or if the device it references is not associated with a parent device, then 0 will be returned. See [The Device Class](#) for more information on associating devices.

**Parameters**

- Parameter: `dvRef`
  - Type: `Integer`
  - Description: This is the reference ID of a device.

**Returns**

- Return value: `dvRef`
  - Type: `Integer`
  - Description: Returns a reference to the given device's parent device.

See also

- `DeviceExistsRef`
- `DeviceExistsAddress`
- `DeviceExistsAddressFull`
- `DeviceExistsCode`
- `GetDeviceRef`
- `GetDeviceRefByName`
- `GetDeviceCode`

GetDeviceCode

**Purpose**

Returns the device code for the given named device. This function can be used with the `IsOff` and `IsOn` functions as well as other functions that require an actual device code.

**Parameters**

- Parameter: `name`
  - Type: `string`
  - Description: This is the name of the device including its location, such as "den table lamp".

**Returns**

- Return value: **Device Address and Code (if present)**
  - Type: `string`
  - Description: This is the address and code field of the device.

**Example**

```plaintext
dim code

code = hs.GetDeviceCode("den table lamp")
msgbox "The address is: " & code
```

Creating, Deleting, or Accessing Devices

In This Section

NewDeviceRef
GetDeviceEnumerator
GetDeviceByRef
DeleteDevice
DeviceCount
DeviceButtonAdd

The application interface procedures in this section deal with creating a device, deleting a device, or getting a reference to a device so that a script or plug-in can work with it.

See also

The Device Class
Device Exists, Reference, Address and/or Code
Device Value, String, or Last Change
Device Script Buttons
Device Energy Management
Device Control API (CAPI)
Images
GetDeviceEnumerator

Purpose

This object can be used to iterate through all of the devices in HomeSeer, allowing you to work with the DeviceClass directly and make any changes or gather information that you need about events.

A DeviceClass has a number of properties that holds information about a device. You can access these properties to get and set this information.

Parameters

None.

Methods

Method: GetNext
Return value: DeviceClass
Type: object

Method: Restart
Return value: none
Type: n/a

Properties

Property: Finished
Type: Boolean
Description: TRUE when the enumerator reaches the last device.

Property: CountChanged
Type: Boolean
Description: TRUE when the count of devices changes during enumeration.

Example

The following script shows how to reiterate though all devices, get and display the device name.

```vbs
Sub Main(parm as object)
Try
Dim dv As Scheduler.Classes.DeviceClass
Dim EN As Scheduler.Classes.clsDeviceEnumeration
EN = hs.GetDeviceEnumerator
If EN Is Nothing Then
    hs.writelog("Script","Error getting Enumerator")
    Exit Sub
End If
Do
    dv = EN.GetNext
    If dv Is Nothing Then Continue Do
    hs.writelog("Script","Device name: " & dv.Name(nothing))
    Loop Until EN.Finished
Catch ex As Exception
    hs.WriteLog("Error", "Exception in script: " & ex.Message)
End Try
End Sub
```

See also

NewDeviceRef
GetDeviceByRef
GetDeviceByRef

**Purpose**

This function returns a reference to the given device object. If the device does not exist, then an empty reference is returned.

**Parameters**

Parameter: `dvRef`
Type: `Integer`
Description: This is the reference ID of a device.

**Returns**

Return value: `device`
Type: `object as DeviceClass`
Description: Returns a reference to the given device object.

**Example**

```vbscript
Sub Main()

    Dim dvRef
    Dim dv

    dvRef = hs.GetDeviceRefByName("Family Room Light")
    if dvRef > 0 then
        Set dv = hs.GetDeviceByRef(dvRef)
    else
        hs.WriteLog "Error","Could not find the reference for the device specified."
        exit Sub
    end if
    hs.WriteLog "Info","The address for the device is " & dv.hc & dv.dc

End Sub
```

See also

- NewDeviceRef
- GetDeviceEnumerator
- DeleteDevice
- DeviceCount
- DeviceButtonAdd

DeleteDevice

**Purpose**

This function removes a device from HomeSeer. **Use this function with caution!**

**Parameters**
Parameter: device_ref
Type: Integer
Description: This is the device reference number.

Returns

Return value: status
Type: boolean
Description: Indicates the success or failure of the operation.

See also
NewDeviceRef
GetDeviceEnumerator
GetDeviceByRef
DeviceCount
DeviceButtonAdd

DeviceCount

Purpose

This function returns the total number of devices currently configured in the system.

Parameters

None.

Returns

Return value: count of devices
Type: integer

Example

dim count

count = hs.DeviceCount

See also
NewDeviceRef
GetDeviceEnumerator
GetDeviceByRef
DeleteDevice
DeviceButtonAdd

DeviceButtonAdd

This function has been deprecated. The enhanced functionality that this command used to provide can now be found by looking at the information in the Device Class ScriptName and ScriptFunc properties, the Device_Type/Device_API and Device_Type/ eDeviceType_Script.

See also
NewDeviceRef
The procedures below are for working with the device's Value, String, or the date/time it was last changed.

**DeviceValue**

**Purpose**

Returns the value stored for this device. Device values are double integer values that are associated with a device. This is a general-purpose value that you can set and read.

**Parameters**

- Parameter: `dvRef`
  - Type: `Integer`
  - Description: This is the device reference ID number.

**Returns**

- Return value: `Value`
  - Type: `Integer`
  - Description: This is the value stored for the device, which is usually the dim level.

  **Note:** This return is an INTEGER value and values are DOUBLE INTEGER, which means that decimal values are truncated. See `DeviceValueEx` for Double Integer returns.
See also
DeviceValueEx
DeviceValueByName
DeviceValueByNameEx
SetDeviceValue
SetDeviceValueByRef
SetDeviceValueByName
DeviceString
DeviceStringByName
SetDeviceString
SetDeviceStringByName
DeviceTime
DeviceTimeByName
DeviceDateTime
SetDeviceLastChange
DeviceLastChange
DeviceLastChangeRef
On - Off

DeviceValueEx

Purpose

Returns the value stored for this device. Device values are a double integer value that is associated with a device. This is a general-purpose value that you can set and read.

Parameters

Parameter: dvRef
Type: Integer
Description: This is the device reference ID number.

Returns

Return value: Value
Type: Double Integer
Description: This is the value stored for the device, which is usually the dim level.

See also
DeviceValue
DeviceValueByName
DeviceValueByNameEx
SetDeviceValue
SetDeviceValueByRef
SetDeviceValueByName
DeviceString
DeviceStringByName
SetDeviceString
SetDeviceStringByName
DeviceTime
DeviceTimeByName
DeviceDateTime
SetDeviceLastChange
DeviceLastChange
DeviceLastChangeRef
On - Off

DeviceValueByName

Home > Scripting > Devices > Device Value, String, or Last Change > DeviceValueEx

Home > Scripting > Devices > Device Value, String, or Last Change > DeviceValueEx

DeviceValueEx

Purpose

Returns the value stored for this device. Device values are a double integer value that is associated with a device. This is a general-purpose value that you can set and read.

Parameters

Parameter: dvRef
Type: Integer
Description: This is the device reference ID number.

Returns

Return value: Value
Type: Double Integer
Description: This is the value stored for the device, which is usually the dim level.

See also
DeviceValue
DeviceValueByName
DeviceValueByNameEx
SetDeviceValue
SetDeviceValueByRef
SetDeviceValueByName
DeviceString
DeviceStringByName
SetDeviceString
SetDeviceStringByName
DeviceTime
DeviceTimeByName
DeviceDateTime
SetDeviceLastChange
DeviceLastChange
DeviceLastChangeRef
On - Off
Purpose

This function is the same as the DeviceValue function, except that you pass this function the text name of the device.

Parameters

Parameter: device_name
Type: string
Description: This is the name of the device and must contain both the location and name. If the device was named lamp and its location was living room, then the device_name parameter would be living room lamp.

Returns

Return value: device value
Type: Integer
Description: This returns the value associated with the device.

Note: This return is an INTEGER value and values are DOUBLE INTEGER, which means that decimal values are truncated. See DeviceValueByNameEx for Double Integer returns.

See also
DeviceValue
DeviceValueEx
SetDeviceValue
SetDeviceValueByRef
DeviceString
DeviceStringByName
SetDeviceString
DeviceTime
DeviceTimeByName
DeviceDateTime
SetDeviceLastChange
DeviceLastChange
DeviceLastChangeRef
On - Off

DeviceValueByNameEx

Purpose

This function is the same as the DeviceValueEx function, except that you pass this function the text name of the device.

Parameters

Parameter: device_name
Type: string
Description: This is the name of the device and must contain both the location and name. If the device was named lamp and its location was living room, then the device_name parameter would be living room lamp.

Returns

Return value: device value
Type: Double Integer
Description: This returns the value associated with the device.

See also
DeviceValue
DeviceValueEx
SetDeviceValue
SetDeviceValueByRef
SetDeviceValue

Purpose

This function sets a value that is associated with this device. Values are used to hold the dim level of a device. You can also use them as user variables in your scripts. Note that HomeSeer will overwrite this value if a command was received for this device. If you are going to use this as storage for your own information, pick a device that does not exist in your home. You can also use virtual devices (devices in the range "q -> z" or unit codes between 17 and 64).

Parameters

Parameter: **device**  
Type: **string**  
Description: This is the device code, such as "A1".

Parameter: **value**  
Type: **double integer**  
Description: This is a numeric value, such as "50".

Returns

None.

Example

```vbs
sub main()

' set the dim value of device B2 to 60.54
hs.SetDeviceValue("B2", 60.54)

end sub
```

See also

DeviceValue  
DeviceValueEx  
DeviceValueByName  
DeviceValueByNameEx  
SetDeviceValueByRef  
SetDeviceValueByName  
DeviceString  
DeviceStringByName  
SetDeviceString  
SetDeviceStringByName  
DeviceTime  
DeviceTimeByName  
DeviceDateTime  
SetDeviceLastChange  
DeviceLastChange  
DeviceLastChangeRef  
On - Off
SetDeviceValueByRef

Purpose

This function sets a value that is associated with this device. Values are used to hold the dim level of a device. You can also use them as user variables in your scripts. Note that HomeSeer will overwrite this value if a command was received for this device. If you are going to use this as storage for your own information, pick a device that does not exist in your home.

Parameters

Parameter: dvRef
Type: Integer
Description: This is the device reference ID number.

Parameter: value
Type: double integer
Description: This is a numeric value, such as "50".

Parameter: trigger
Type: Boolean
Description: When set to FALSE, the value will be changed without triggering events that are set to trigger when the device changes. Set this to True normally so that events can trigger when the device's value is updated.

Returns

None.

Example

Sub Main(ByVal Parms As Object)
    ' set the value of device whose reference ID is 1234 to 60.54
    hs.SetDeviceValueByRef(1234, 60.54, True)
end sub

See also

DeviceValue
DeviceValueEx
DeviceValueByName
DeviceValueByNameEx
SetDeviceValue
SetDeviceValueByName
DeviceString
DeviceStringByName
SetDeviceString
SetDeviceStringByName
DeviceTime
DeviceTimeByName
DeviceDateTime
DeviceLastChange
DeviceLastChangeRef
On - Off

SetDeviceValueByName

Purpose
This function is the same as `SetDeviceValue` except you pass this function the actual text name of the device.

**Parameters**

Parameter: `device_name`
Type: `string`
Description: This is the name of the device and must contain both the location and name. If the device was named lamp and its location was living room, the `device_name` parameter would be `living room lamp`.

Parameter: `value`
Type: `double integer`

**Returns**

None.

See also
- `DeviceValue`
- `DeviceValueEx`
- `DeviceValueByName`
- `DeviceValueByNameEx`
- `SetDeviceValue`
- `DeviceString`
- `DeviceStringByName`
- `SetDeviceString`
- `SetDeviceStringByName`
- `DeviceTime`
- `DeviceTimeByName`
- `DeviceDateTime`
- `SetDeviceLastChange`
- `DeviceLastChange`
- `DeviceLastChangeRef`
- `On - Off`

---

**DeviceString**

**Purpose**

Returns the character string set for a device. See `SetDeviceString`.

**Parameters**

None.

**Returns**

Return value: `dvRef`
Type: `integer`
Description: This is the device reference ID number for the device.

**Example**

```vbnet
sub main()
    dim s
    s=hs.DeviceString(5678)
    msgbox s
end sub
```

See also
- `DeviceValue`
- `DeviceValueEx`
- `DeviceValueByName`
- `DeviceValueByNameEx`
SetDeviceValue
SetDeviceValueByRef
SetDeviceValueByName
DeviceStringByName
SetDeviceString
SetDeviceStringByName
DeviceTime
DeviceTimeByName
DeviceDateTime
SetDeviceLastChange
DeviceLastChange
DeviceLastChangeRef
On - Off

Home > Scripting > Devices > Device Value, String, or Last Change > DeviceStringByName

DeviceStringByName

**Purpose**

Returns the character string set for a device. See SetDeviceString.

**Parameters**

Parameter: name
Type: string
Description: This is the name of the device. The name includes its location, such as den table lamp.

**Returns**

None.

See also
DeviceValue
DeviceValueEx
DeviceValueByName
DeviceValueByNameEx
SetDeviceValue
SetDeviceValueByRef
SetDeviceValueByName
DeviceString
SetDeviceString
SetDeviceStringByName
DeviceTime
DeviceTimeByName
DeviceDateTime
SetDeviceLastChange
DeviceLastChange
DeviceLastChangeRef
On - Off

Home > Scripting > Devices > Device Value, String, or Last Change > SetDeviceString

SetDeviceString

**Purpose**

This function sets a string as the device status. The string “message” is displayed in the Status screen. This appears on the web page and the local device list. This can be used to display the status of special devices like thermostats and weather stations. Note that this does not affect the actual status/value for the device, which can be accessed by DeviceValue.

The text string can also contain HTML code, so you can add affects to the status like changing its color or making it scroll. See the example below to create some status using the marquee and blink HTML tags. Note the marquee tag is only supported in Internet Explorer and the blink tag is only supported in Netscape.
### Parameters

- **Parameter:** `dvRef`  
  **Type:** Integer  
  **Description:** This is the device reference ID number.

- **Parameter:** `message`  
  **Type:** String  
  **Description:** This is the status string for the device, such as "72 degrees".

- **Parameter:** `reset`  
  **Type:** Boolean  
  **Description:** If this is set to TRUE, the device change date/time will be updated (normally a string change will not update the device last change date/time).

### Returns

None.

### Example

```vbs
sub main()
    hs.SetDeviceString(5678, Motion Detected", True)
    ' add some HTML to the text to create a scrolling status
    hs.SetDeviceString(5678, "<MARQUEE><blink><b>Motion Detected</MARQUEE> </b></blink>", True)
end sub
```

### See also

- DeviceValue
- DeviceValueEx
- DeviceValueByName
- DeviceValueByNameEx
- SetDeviceValue
- SetDeviceValueByRef
- SetDeviceValueByName
- DeviceString
- DeviceStringByName
- SetDeviceStringByName
- DeviceTime
- DeviceTimeByName
- DeviceDateTime
- SetDeviceLastChange
- DeviceLastChange
- DeviceLastChangeRef
- On - Off

---

### SetDeviceStringByName

#### Purpose

This function sets a string as the device status using the actual name of the device combined with its location. See [SetDeviceString](#).

#### Parameters

- **Parameter:** `name`  
  **Type:** String  
  **Description:** This is the name of the device including its location, such as "den table lamp". Note the name is not case-sensitive.

- **Parameter:** `message`  
  **Type:** String  
  **Description:** This is the status string for the device, such as "72 degrees".

- **Parameter:** `reset`  
  **Type:** Boolean  
  **Description:** If this is set to TRUE, the device change date/time will be updated (normally a string change will not update the device last change date/time).
Returns

None.

See also
DeviceValue
DeviceValueEx
DeviceValueByName
DeviceValueByNameEx
SetDeviceValue
SetDeviceValueByRef
SetDeviceValueByName
DeviceString
DeviceStringByName
SetDeviceString
DeviceTime
DeviceTimeByName
DeviceDateTime
SetDeviceLastChange
DeviceLastChange
DeviceLastChangeRef
On - Off

Purpose

Returns the time in minutes since the device status last changed. This can be used to see how long a device has been ON or OFF.

Parameters

Parameter: dvRef
Type: Integer
Description: This is the device reference ID number.

Returns

Return value: time
Type: integer
Description: This is the amount of time in minutes since last device change.

See also
DeviceValue
DeviceValueEx
DeviceValueByName
DeviceValueByNameEx
SetDeviceValue
SetDeviceValueByRef
SetDeviceValueByName
DeviceString
DeviceStringByName
SetDeviceString
DeviceTime
DeviceTimeByName
DeviceDateTime
SetDeviceLastChange
DeviceLastChange
DeviceLastChangeRef
On - Off
DeviceTimeByName

**Purpose**

Returns the time in minutes since the device status last changed. This can be used to see how long a device has been ON or OFF.

**Parameters**

- **Parameter:** device name  
  **Type:** string  
  **Description:** This is the name of the device and it must contain both the location and name. If the device was named lamp and its location was living room, then the device name parameter would be living room lamp.

**Returns**

- **Return value:** time  
  **Type:** integer  
  **Description:** This is the amount of time in minutes since last device status change.

See also
DeviceValue
DeviceValueEx
DeviceValueByName
DeviceValueByNameEx
SetDeviceValue
SetDeviceValueByRef
DeviceString
DeviceStringByName
SetDeviceString
SetDeviceStringByName
DeviceTime
DeviceDateTime
SetDeviceLastChange
DeviceLastChange
DeviceLastChangeRef
On - Off

DeviceDateTime

**Purpose**

Returns the time as a date/time object, of when the device last changed value or string.

**Parameters**

- **Parameter:** dvRef  
  **Type:** Integer  
  **Description:** This is the device reference ID number.

**Returns**

- **Return value:** time  
  **Type:** Date  
  **Description:** This is the date and time of the last change to the device.

See also
DeviceValue
DeviceValueEx
DeviceValueByName
DeviceValueByNameEx
SetDeviceValue
SetDeviceValueByRef
DeviceString
DeviceStringByName
SetDeviceString
SetDeviceStringByName
DeviceTime
DeviceDateTime
SetDeviceLastChange
DeviceLastChange
DeviceLastChangeRef
On - Off
SetDeviceLastChange

Purpose

This function sets the last change time of a device.

Parameters

Parameter: dvRef
Type: Integer
Description: This is the device reference ID number.

Parameter: date-time
Type: date
Description: This is the date and time to set the last change to.

Returns

Return value: none

Example

hs.SetDeviceLastChange(5678, Now)
hs.SetDeviceLastChange(5678, Convert.ToDateTime("1/1/13 4:00 PM"))
Purpose

This function returns the date and time the device last changed its value or character string.

Parameters

Parameter: device
Type: string
Description: This is the house code and unit code or device code of the device, such as "A1" or "q17".

Returns

Return value: device time
Type: date
Description: This is the date and time the device last changed.

Example

sub main()
  Dim last_change As Date
  ' get the last change time for the device name "living room lamp"
  last_change = hs.DeviceLastChange("living room lamp")
end sub

See also
DeviceValue
DeviceValueEx
DeviceValueByName
DeviceValueByNameEx
SetDeviceValue
SetDeviceValueByRef
SetDeviceValueByName
DeviceString
DeviceStringByName
SetDeviceString
SetDeviceStringByName
DeviceTime
DeviceTimeByName
DeviceDateTime
SetDeviceLastChange
DeviceLastChangeRef
On - Off

DeviceLastChangeRef

Purpose

This function returns the date and time the device last changed status.

Parameters

Parameter: dvRef
Type: Integer
Description: This is the device reference ID number.

Returns

Return value: device time
Type: date
Description: This is the date and time the device last changed.

Example

sub main()
Dim last_change As Date
'
' get the last change time for the device whose reference ID number is 5678
last_change = hs.DeviceLastChangeRef(5678)
end sub

See also
DeviceValue
DeviceValueEx
DeviceValueByName
DeviceValueByNameEx
SetDeviceValue
SetDeviceValueByRef
SetDeviceValueByName
DeviceString
DeviceStringByName
SetDeviceString
SetDeviceStringByName
DeviceTime
DeviceTimeByName
DeviceDateTime
SetDeviceLastChange
DeviceLastChange
On - Off

On - Off

These functions are largely deprecated due to the nature of devices not always being the same value or state for On and Off - for example, an X-10 device may have an "On" value of 100, but a Z-Wave device is "On" at values 99 (dimmable device) or 255 (non-dimmable).

For backward compatibility, these functions return True for their query if the value is 0 (for Off), or in the range 1 to 100 or 255 (for On).
IsOn

Purpose
This function checks the status of a device.

Parameters
Parameter: device reference
Type: Integer
Description: This is the device reference ID.

Returns
Return value: status
Type: Boolean
Description: This returns TRUE if the device is on or dimmed or FALSE if the device is off.

Example
Sub Main()
  If hs.IsOn(1234) Then
    hs.Speak("The Light is On")
  End If
End Sub

Note:
On - Off

See also
IsOnByName
IsOff
IsOffByName

IsOnByName

Purpose
This function checks the status of a device, using the device's name.

Parameters
Parameter: device name
Type: string
Description: This is the name of the device. The device name must include the device's location and its name, such as "den table lamp". The name is not case-sensitive.

Returns
Return value: status
Type: boolean
Description: This returns TRUE if a device is on or dimmed or FALSE if it's off.

Note:
On - Off
See also
IsOn
IsOff
IsOffByName

Purpose
This function checks the status of a device.

Parameters
Parameter: device reference
Type: Integer
Description: This is the unique device reference ID.

Returns
Return value: status
Type: Boolean
Description: This returns TRUE if the device is OFF and FALSE if it's not (on or dimmed).

Note:
On - Off

See also
IsOn
IsOnByName
IsOffByName

IsOffByName

Purpose
This function checks the status of a device, using the device's name.

Parameters
Parameter: device name
Type: string
Description: This is the name of the device. The device name must include the device's location and its name like "den table lamp". The name is not case-sensitive.

Returns
Return value: status
Type: boolean
Description: This returns TRUE if a device is off and FALSE if it's on or dimmed.

Note:
On - Off
See also
isOn
isOnByName
isOff

Device Script Buttons

Device Script Buttons are an enhanced version of the functionality previously provided in HS2 which allowed a button to be added to a device, which is then rendered by the UI, and will run a script when the button is activated/controlled.

Unlike previous versions, Device Script Buttons can specify the location for the button or the button can be hidden in the UI so that the button is only available as a CAPI Control object.

Device Script Buttons are added and removed with the script commands described in this section.

See also
The Device Class
Device Exists, Reference, Address and/or Code
Creating, Deleting, or Accessing Devices
Device Value, String, or Last Change
Device Energy Management
Device Control API (CAPI)
Images

DeviceScriptButton:AddButton

Function DeviceScriptButton:AddButton(ByName dvRef As Integer, ByVal Label As String, ByVal Value As Double,
As String, _
ByVal ScriptFile As String, ByVal ScriptFunc As String, ByVal ScriptParm
As Boolean
ByVal Row As UInt16, ByVal Column As UInt16, ByVal ColumnSpan As UInt16)

Purpose
Use DeviceScriptButton:Add to add a button to the device to be shown in the UI, and that will launch a script when the button is activated or controlled.

Parameters

Parameter: dvRef
Type: Integer
Description: This is the device reference number.

Parameter: Label
Type: String
Description: This is the label that you want to have appear on the button when it is rendered in the UI, and is also the label used in the CAPIControl object when the button script is being invoked through CAPI.

Parameter: Value
Type: Double
Description: This is the value associated with this button. The value must be unique amongst all other buttons and value/status pairs on this device, as the value is used by plug-ins such as HSTouch as the trigger for executing the script.

Parameter: ScriptFile
Type: String
Description: This is the name of the script file in the HomeSeer Scripts directory that you wish to have executed when the button is activated (controlled). A path may be provided in front of the file name as long as it is relative to the scripts directory.

Parameter: ScriptFunc
Type: String
Description: This is the name of the procedure (or function) to be executed in the ScriptFile. If this parameter is null (empty), then the Main sub will be
called in the script file.

Parameter: **ScriptParm**
Type: **string**
Description: This is an ADDITIONAL parameter which will be passed to the procedure ScriptFunc (or Main). It is an additional parameter because HomeSeer will always provide the device reference ID number as the first (Parameter 0) parameter. Whatever is provided here will be in parameter location 1 in the array of objects passed as a parameter to your procedure.

Parameter: **Row**
Type: **unsigned 16 bit integer**
Description: When the button is rendered in the UI, you can control where the button will appear by specifying the row number here. A row value of 0 will prevent the button from appearing in the UI, but the CAPI Control object for the device script button will still be available to scripts or plug-ins controlling the device through CAPI.

Parameter: **Column**
Type: **unsigned 16 bit integer**
Description: When the button is rendered in the UI, you can control where the button will appear by specifying the column number here. A column value of 0 will prevent the button from appearing in the UI, but the CAPI Control object for the device script button will still be available to scripts or plug-ins controlling the device through CAPI.

Parameter: **ColumnSpan**
Type: **unsigned 16 bit integer**
Description: When the button is rendered in the UI, you can control how the button will appear by specifying the ColumnSpan number here. If the label for the button is large, the width of the rendered button may mis-align other buttons and controls in the UI. Specifying that the button should take up more than one column can improve the layout of the controls on the DeviceManagement page and when it is rendered by other UIs.

**Returns**

Return value: **status**
Type: **Boolean**
Description: Returns FALSE if the operation was not successful, which can happen if the device reference ID is invalid, the label provided is null or empty, the label already exists with another button, or the script file name is not provided.

See also

DeviceScriptButton_DeleteButton
DeviceScriptButton_DeleteAll
DeviceScriptButton_Locate
DeviceScriptButton_List

---

**DeviceScriptButton_DeleteButton**

**Function** DeviceScriptButton_DeleteButton(ById Val As Integer, ByVal Value As Double) As Boolean

**Purpose**

Use DeviceScriptButton_Delete to remove a script button from the device.

**Parameters**

Parameter: **dvRef**
Type: **integer**
Description: This is the device reference number.

Parameter: **Value**
Type: **double integer**
Description: This is the value that you used when you added the button to the device.

**Returns**

Return value: **status**
Type: **Boolean**
Description: Returns FALSE if the operation was not successful, which can happen if the device reference ID is invalid or the label provided is null or empty.
DeviceScriptButton_DeleteAll

Sub DeviceScriptButton_DeleteAll(ByVal dvRef As Integer)

Purpose

Use DeviceScriptButton_DeleteAll to remove ALL script buttons from the device.

Parameters

Parameter: dvRef
Type: integer
Description: This is the device reference number.

Returns

None.

DeviceScriptButton_Locate

Function DeviceScriptButton_Locate(ByVal dvRef As Integer, ByVal Value As Double, ByVal Row As UInt16, ByVal Column As UInt16, ByVal ColumnSpan As UInt16) As Boolean

Purpose

Use DeviceScriptButton_Location to change the location parameters ONLY on an EXISTING device script button.

Parameters

Parameter: dvRef
Type: integer
Description: This is the device reference number.

Parameter: Value
Type: double integer
Description: This is the value that you used when the script button was added to the device.

Parameter: Row
Type: unsigned 16 bit integer
Description: When the button is rendered in the UI, you can control where the button will appear by specifying the row number here. A row value of 0 will prevent the button from appearing in the UI, but the CAPI Control object for the device script button will still be available to scripts or plug-ins controlling the device through CAPI.
Parameter: Column
Type: unsigned 16 bit integer
Description: When the button is rendered in the UI, you can control where the button will appear by specifying the column number here. A column value of 0 will prevent the button from appearing in the UI, but the CAPI Control object for the device script button will still be available to scripts or plug-ins controlling the device through CAPI.

Parameter: ColumnSpan
Type: unsigned 16 bit integer
Description: When the button is rendered in the UI, you can control how the button will appear by specifying the ColumnSpan number here. If the label for the button is large, the width of the rendered button may mis-align other buttons and controls in the UI. Specifying that the button should take up more than one column can improve the layout of the controls on the DeviceManagement page and when it is rendered by other UIs.

Returns

Return value: status
Type: Boolean
Description: Returns FALSE if the operation was not successful, which can happen if the device reference ID is invalid, the label provided is null or empty or the label does not already exist with a script button on the device.

See also
DeviceScriptButton_AddButton
DeviceScriptButton_DeleteButton
DeviceScriptButton_DeleteAll
DeviceScriptButton_List

DeviceScriptButton_List

Function DeviceScriptButton_List(ByVal dvRef As Integer) As String()

Purpose

Use DeviceScriptButton_List to see all of the parameters for all device script buttons that have already been added to the device.

Parameters

Parameter: dvRef
Type: integer
Description: This is the device reference number.

Returns

Return value: list
Type: array of string
Description: The return array is a COMMA separated list of the parameters of each script button that exists on the device. Each string in the array corresponds to one script button, and commas will be provided between each parameter even if the parameter is empty. THE RETURN IS NOT CHECKED FOR COMMAS WHICH MAY APPEAR AS PART OF THE LABEL OR PARAMETER IN THE DEVICE SCRIPT BUTTON.

The order in which the information is concatenated together is as follows:

- Label
- Value
- Script File
- Script Function
- Script Parameter
- Row
- Column
- ColumnSpan

Example: Press Me,123,\My Scripts\ButtonRun.vb.ButtonFunc,Hello World,2,1,2

See also
DeviceScriptButton_AddButton
Device Energy Management

The methods and objects in this section provide a method of storing energy information and calculating energy usage and cost on a per-device basis. When energy information is added, it is stored in the Energy database. "Calculators" can be added to a device's Energy object to track energy amounts and cost over periods of time. If you want to know the energy used and the cost for the past hour, so far today, yesterday, and last week, that is four calculators. For graphing or more intensive calculations, a method is also provided which allows energy data records to be returned given a starting date/time and a length of time.

See also
The Device Class
Device Exists, Reference, Address and/or Code
Creating, Deleting, or Accessing Devices
Device Value, String, or Last Change
Device Script Buttons
Device Control API (CAPI)
Images

Energy_AddData, Energy_AddDataArray

These methods are used to add one record of energy data to a device (AddData) or several records (AddDataArray). If the return value from this procedure is False, then HomeSeer was unable to accept the data and add it to the data queue to be added to the database. See the EnergyData Class for more information about the information stored in the object referenced in these procedures.

Function Energy_AddData(ByVal dvRef As Integer, ByVal Data As EnergyData) As Boolean
Function Energy_AddDataArray(ByVal dvRef As Integer, ByVal colData As EnergyData()) As Boolean

See also
Energy_SetEnergyDevice
Energy_AddCalculator, Energy_AddCalculatorEvenDay
Energy_CalcCount
Energy_GetCalcByDevice, Energy_GetCalcByIndex
Energy_GetData, Energy_GetArchiveData
Energy_RemoveData

EnergyData Class

The EnergyData object holds a set of energy usage or consumption information, and the rate of that energy at the time it was consumed or produced.

Public Class EnergyData

Public dvRef As Integer 'Device reference ID number for the device this energy data is for.
Public Direction As enumEnergyDirection = enumEnergyDirection.Consumed 'Indicates whether the energy was consumed (used) or produced (created).
Public Amount As Double 'Always measured in Watts


Public Amount_Start As Date 'The start of the time period this measurement is for.
Public Amount_End As Date 'The end of the time period this measurement is for.
Public Rate As Single 'Always measured in kWH
Public UserCode As Integer 'For the user to indicate something about this reading.

End Class

Note: When initialized, the value of Direction must be provided:

Example:
Dim ED As New EnergyData( enumEnergyDirection.Consumed )

See also

enumEnergyDevice

These enum values can be used in the EnergyData to indicate the type of device the energy data came from.

Public Enum enumEnergyDevice
    _Undefined_ = 0 ' Not defined
    Light_Small = 1 ' A small light
    Light_Large = 2 ' A large light or several lights
    Appliance = 10 ' Any appliance
    Appliance_Small = 11 ' A small appliance such as a toaster
    Appliance_Large = 12 ' A large appliance such as an oven
    Utility = 20 ' A utility device
    Utility_Small = 21 ' A small utility device such as a water filter
    Utility_Large = 22 ' A large utility device such as a well pump
    Entertainment = 30 ' An entertainment device
    Entertainment_Small = 31 ' A small entertainment device such as a radio
    Entertainment_Large = 32 ' A large entertainment device such as a home theatre system
    HVAC = 40 ' An HVAC device
    Electric_AC = 41 ' An Air Conditioning device
    Electric_Heat = 42 ' An electric heating device
    Panel = 51 ' An electrical panel providing several branches of electrical service to the home.
    Panel_A = 52
    Panel_B = 53
    Panel_C = 54
    Panel_D = 55
    Panel_E = 56
    Panel_F = 57
    Meter = 61
    Meter_Service = 62 ' An electric meter measuring usage for an unspecified or general purpose.
    Meter_Device = 63 ' An electric meter measuring usage for electrical service such as a house service entrance.
    Generator = 71 ' An electricity producing generator.
    Solar_Panel = 72 ' An electricity producing solar panel.
    Wind_Turbine = 73 ' An electricity producing wind turbine.
    Water_Turbine = 74 ' An electricity producing water (wave) turbine.
    Other = 99 ' A device (consumer or producer) that does not fit any other device type.
End Enum

See also
enumEnergyDirection

HomeSeer HS3 - End User Documentation
enumEnergyDirection

This enum is used in the EnergyData to indicate whether the energy information is for energy consumed or produced.

```vbnet
Public Enum enumEnergyDirection
    Consumed = 1
    Produced = 2
End Enum
```

See also
enumEnergyDevice

Energy_SetEnergyDevice

This procedure is used to set the type of energy consumption or energy producing device for the reference ID `dvRef` in the energy database. This procedure is also used to create an initial energy object in the system if one does not exist. The energy object's device name, location, and location2 properties will also be updated whenever this procedure is called.

```vbnet
Public Function Energy_SetEnergyDevice(ByVal dvRef As Integer, ByVal DeviceType As enumEnergyDevice) As Boolean
```

See also

enumEnergyDevice

These enum values can be used in the EnergyData to indicate the type of device the energy data came from.

```vbnet
Public Enum enumEnergyDevice
    _Undefined_ = 0 ' Not defined
    Light_Small = 1 ' A small light
    Light_Large = 2 ' A large light or several lights
    Appliance = 10 ' Any appliance
    Appliance_Small = 11 ' A small appliance such as a toaster
    Appliance_Large = 12 ' A large appliance such as an oven
    Utility = 20 ' A utility device
    Utility_Small = 21 ' A small utility device such as a water filter
End Enum
```
Utility_Large = 22  ' A large utility device such as a well pump
Entertainment = 30  ' An entertainment device
Entertainment_Small = 31  ' A small entertainment device such as a radio
Entertainment_Large = 32  ' A large entertainment device such as a home theatre system
HVAC = 40  ' An HVAC device
Electric_AC = 41  ' An Air Conditioning device
Electric_Heat = 42  ' An electric heating device
Panel = 51  ' An electrical panel providing several branches of electrical service to the home.
    Panel_A = 52
    Panel_B = 53
    Panel_C = 54
    Panel_D = 55
    Panel_E = 56
    Panel_F = 57
Meter = 61  ' An electric meter measuring usage for an unspecified or general purpose.
    Meter_Service = 62  ' An electric meter measuring usage for electrical service such as a house service entrance.
    Meter_Device = 63  ' An electric meter measuring usage for a single device.
Generator = 71  ' An electricity producing generator.
Solar_Panel = 72  ' An electricity producing solar panel.
Wind_Turbine = 73  ' An electricity producing wind turbine.
Water_Turbine = 74  ' An electricity producing water (wave) turbine.
Other = 99  ' A device (consumer or producer) that does not fit any other device type.

End Enum

See also

Energy_AddCalculator, Energy_AddCalculatorEvenDay

These functions are used to add an energy calculator to an energy object in HomeSeer. An energy calculator updates when energy data is added, and calculates the total energy consumed or produced for the time period, as well as the cost of that energy. The return value is a string that is empty if the procedure succeeded, and contains error information if it did not.

Function Energy_AddCalculator(ByVal dvRef As Integer, ByVal Name As String, ByVal Range As TimeSpan, ByVal StartBack As TimeSpan) As String
    Function Energy_AddCalculatorEvenDay(ByVal dvRef As Integer, ByVal Name As String, ByVal Range As TimeSpan, ByVal StartBack As TimeSpan) As String

Calculators that are for days, for example the amount of energy used a week ago today, can use _AddCalculatorEvenDay and the calculation will automatically be truncated at even day boundaries.

Parameters

Parameter: dvRef
Type: Integer
Description: The unique device reference ID number.

Parameter: Name
Type: String
Description: This is the name of the calculator, which may be used to identify the calculation being done.

Parameter: Range
Type: TimeSpan
Description: This is the time period that you wish the calculation to be done over.

Parameter: StartBack
Type: TimeSpan
Description: This is the period of time, starting from “Now”, to go back to and set as the start time for the calculation.
**Returns**

Return value: **Result**

Type: **String**

Description: This is the result of the operation - if it succeeded, it will be an empty string - if it failed, it will contain information about the error.

**Example:**

To create a calculator for the energy used in the past hour:

```vba
Result = hs.Energy_AddCalculator(1234, "Last Hour Used", New TimeSpan(1, 0, 0), New TimeSpan(0, 0, 0))
If Not String.IsNullOrEmpty(Result) Then
    hs.WriteLog("Error", "Calculator add failed, reason=" & Result)
End If
```

To create a calculator for the energy used in the past hour yesterday:

```vba
Result = hs.Energy_AddCalculator(1234, "Last Hour Used Yesterday", New TimeSpan(1, 0, 0), New TimeSpan(1, 0, 0, 0))
```

See also

- Energy_AddData, Energy_AddDataArray
- Energy_SetEnergyDevice
- Energy_CalcCount
- Energy_GetCalcByName, Energy_GetCalcByIndex
- Energy.GetData, Energy_GetArchiveData
- Energy_RemoveData

---

**Energy_CalcCount**

This function will return the number of energy calculators currently attached to the energy object for the device referenced by dvRef. This can be used in conjunction with Energy_GetCalcByIndex to retrieve all of the calculator data for a device.

```vba
Function Energy_CalcCount(ByVal dvRef As Integer) As Integer
```

See also

- Energy_AddData, Energy_AddDataArray
- Energy_SetEnergyDevice
- Energy_AddCalculator, Energy_AddCalculatorEvenDay
- Energy_GetCalcByName, Energy_GetCalcByIndex
- Energy.GetData, Energy_GetArchiveData
- Energy_RemoveData

---

**Energy_GetCalcByName, Energy_GetCalcByIndex**

These functions return energy calculation results for the device referenced by dvRef. The EnergyCalcData objects contain energy results as well as other parameters used when the calculator was created, but they are for reference only and changing those properties will NOT be reflected back to the real calculator object on the device.

See the EnergyCalcData Class object definition for more information about its members.

Use Energy_GetCalcByIndex after using Energy_CalcCount to iterate through each calculator on an energy object without having to know its name.

```vba
Function Energy_GetCalcByName(ByVal dvRef As Integer, ByVal Name As String) As EnergyCalcData
```

---

Home > Scripting > Devices > Device Energy Management > Energy_CalcCount

Home > Scripting > Devices > Device Energy Management > Energy_GetCalcByName, Energy_GetCalcByIndex
Function Energy_GetCalcByIndex(ByVal dvRef As Integer, ByVal Index As Integer) As EnergyCalcData

See also
Energy_AddData, Energy_AddDataArray
Energy_SetEnergyDevice
Energy_AddCalculator, Energy_AddCalculatorEvenDay
Energy_CalcCount
Energy_GetData, Energy_GetArchiveData
Energy_RemoveData

EnergyCalcData Class

This class object is used as a return value from Energy_GetCalcByName or Energy_GetCalcByIndex. It contains the results of the most recent data calculation performed on the energy data added to the device.

Public Class EnergyCalcData
    Public Range As TimeSpan ' The amount of time to be included in the calculation starting from the starting point.
    Public StartBack As TimeSpan ' The amount of time to be subtracted from NOW to get our starting point.
    Public RoundDay As Boolean = False ' Whether to round the time to an even day.
    Public Property Name As String ' The name of the energy calculator this data belongs to.
    Public ReadOnly Property Amount As Double ' The amount of energy rounded to 3 decimal places.
    Public ReadOnly Property AmountPrecise As Double ' The amount of energy without any rounding.
    Public ReadOnly Property Cost As Double ' The cost of the energy calculated, rounded to 2 decimal places.
    Public ReadOnly Property CostPrecise As Double ' The cost of the energy calculated without any rounding.
End Class

See also

Energy_GetData, Energy_GetArchiveData

Function Energy_GetData(ByVal dvRef As Integer, ByVal dteStart As Date, ByVal dteEnd As Date) As Collections.Generic.List(Of EnergyData)

Function Energy_GetArchiveData(ByVal dvRef As Integer, ByVal dteStart As Date, ByVal dteEnd As Date) As Collections.Generic.List(Of EnergyData)
EnergyData Class

The EnergyData object holds a set of energy usage or consumption information, and the rate of that energy at the time it was consumed or produced.

Public Class EnergyData
    Public dvRef As Integer 'Device reference ID number for the device this energy data is for.
    Public Direction As EnumEnergyDirection = EnumEnergyDirection.Consumed 'Indicates whether the energy was consumed (used) or produced (created).
    Public Amount As Double 'Always measured in Watts
    Public Amount_Start As Date 'The start of the time period this measurement is for.
    Public Amount_End As Date 'The end of the time period this measurement is for.
    Public Rate As Single 'Always measured in kWH
    Public UserCode As Integer 'For the user to indicate something about this reading.
End Class

Note: When initialized, the value of Direction must be provided:
Example: Dim ED As New EnergyData(EnumEnergyDirection.Consumed)

See also

enumEnergyDevice

These enum values can be used in the EnergyData to indicate the type of device the energy data came from.

Public Enum EnumEnergyDevice
    _Undefined_ = 0 ' Not defined
    Light_Small = 1 ' A small light
    Light_Large = 2 ' A large light or several lights
    Appliance = 10 ' Any appliance
    Appliance_Small = 11 ' A small appliance such as a toaster
    Appliance_Large = 12 ' A large appliance such as an oven
    Utility = 20 ' A utility device
    Utility_Small = 21 ' A small utility device such as a water filter
    Utility_Large = 22 ' A large utility device such as a well pump
    Entertainment = 30 ' An entertainment device
    Entertainment_Small = 31 ' A small entertainment device such as a radio
    Entertainment_Large = 32 ' A large entertainment device such as a home theatre system
    HVAC = 40 ' An HVAC device
    Electric_AC = 41 ' An Air Conditioning device
    Electric_Heat = 42 ' An electric heating device
Panel = 51
' An electrical panel providing several branches of electrical service to the home.
Panel_A = 52
Panel_B = 53
Panel_C = 54
Panel_D = 55
Panel_E = 56
Panel_F = 57
Meter = 61
Meter_Service = 62
' An electric meter measuring usage for an unspecified or general purpose.
Meter_Device = 63
Generator = 71
Solar_Panel = 72
Wind_Turbine = 73
Water_Turbine = 74
Other = 99
' An electricity producing generator.
' An electricity producing solar panel.
' An electricity producing wind turbine.
' An electricity producing water (wave) turbine.
' A device (consumer or producer) that does not fit any other device type.
End Enum

See also
enumEnergyDirection

enumEnergyDirection
See also
enumEnergyDevice

Energy_RemoveData
This command will remove energy data for the device referenced by dvRef, from the date/time specified in dteStart and older.

Function Energy_RemoveData(ByVal dvRef As Integer, ByVal dteStart As Date) As Integer

Example:
To remove energy records from the system and the database which are more than a year old...

Dim Result As Integer
Result = hs.Energy_RemoveData(1234, Now.Subtract(New TimeSpan(365, 0, 0, 0)))
If Result < 0 Then
    hs.WriteLogEx("Error", "Error removing energy data from a year ago for " & DeviceName(1234), COLOR_RED)
Else
    hs.WriteLog("Info", "Maintenance on energy data for " & DeviceName(1234) & 
    " resulted in " & Result.ToString & " records being removed.")
End If

HomeSeer HS3 - End User Documentation
Device Control API (CAPI)

The device control API is the sole method for scripts and plug-ins to control devices. The object of the device type class model is to provide a way for devices to be self-describing in how they are to be visually rendered, how they are to be controlled, and for as many common device types as we can create that they subscribe to a standard model that enables other applications to use them.

The device control API is a means for programmers creating alternative interfaces for HomeSeer to be able to obtain status information and control devices, regardless of whether the device was designed to use buttons, status, or values (or a combination) as the main control mechanism.

How It Works

The device control API works from the premise that HomeSeer always knows how to render a device’s status and provide control options on the device status page. All devices will use status/values/graphics pairs to represent both control and status information. If the device has buttons associated with it, the buttons are displayed. If the device has a string value, the string is displayed under the status column, etc.… Using the code that HomeSeer uses, the control API returns a status object for a device, and a collection of control capabilities can be obtained. Codes in the control capability objects tell the user of the API what kind of a control method is used and the data/value that corresponds to it.

All of the procedures of the device control API use the unique device reference number (dv.ref) - it is best to have a good working knowledge of the HomeSeer scripting interface commands that work with the device reference IDs. In most cases, the device enumerator is your friend.

The pages herein describe the API procedures and objects.

A plug-in will access the control API through the HomeSeer scripting interface or HSApplicationAPI interface. This is accessed through the “hs” object that is obtained when a plug-in connects to HomeSeer.

See also
The Device Class
Creating, Deleting, or Accessing Devices
Device Value, String, or Last Change
Device Script Buttons
Device Energy Management
Images

CAPIGetStatus

Public Function CAPIGetStatus(ByVal dvRef As Integer) As iCAPIStatus

The return value is a CAPIStatus object. See the iCAPIStatus subject for a description of its properties and members.

Example
Sub Main(DvVal Param As Object)
    Dim enx As Scheduler clsDeviceEnumeration
    Dim dv As Scheduler Classes.DeviceClass
    Dim CS As Scheduler CAPIStatus
    Dim s As String

    enx = hs.GetDeviceEnumerator()
    If enx Is Nothing Then
        Else
            hs.WriteLog("Test", "It is not an object.")
        End If
    Do While enx.Finished = False
        dv = enx.GetNext()
        If Not dv Is Nothing Then
            CS = hs.CAPIGetStatus(dv.ref)
            s = "Device " & dv.location & ", " & dv.Name
            s = s & ". Status is " & CS.Status & ". Image is " & CS.ImageFile
            hs.WriteLog("Enumeration", s)
        End If
    Loop
End Sub

iCAPIStatus

iCAPIStatus is an interface class with the following definition, used with the CAPIStatus procedures:

Public Interface ICAPIStatus
    Property Status() As String
    Property StatusHTML() As String
    Property ImageFile() As String
    Property ClassName() As String
    Property Value As Double
End Interface

Property Description
Status This is the status text as displayed on the HomeSeer device utility page and other UI pages.
StatusHTML If the device's status contains HTML, which is sometimes stripped away when the status is displayed, this property contains the status with HTML.
ImageFile This is the path to a graphics file that corresponds with the device's current status value. It is the path to the same graphic as shown on the device utility page.
ClassName This is the class name assigned to the cell that the device status is displayed within on the HomeSeer device utility page.
Value This is the current value of the device which corresponds to the status information present in the other properties.

See also
CAPIGetControl, CAPIGetControlEx, CAPIGetSingleControl

Public Function CAPIGetControl(ByVal dvRef As Integer) As CAPIControl()

Public Function CAPIGetControlEx(ByVal dvRef As Integer, ByVal SingleRangeEntry As Boolean) As CAPIControl()

Public Function CAPIGetSingleControl(ByVal dvRef As Integer, ByVal SingleRangeEntry As Boolean, ByVal Label As String, ByVal ExactCase As Boolean, ByVal Contains As Boolean) As CAPIControl

Public Function CAPIGetSingleControlByUse(ByVal dvRef As Integer, ByVal UseType As ePairControlUse) As CAPIControl

Because a device has multiple control options, two of these functions return an array of CAPIControl objects, the third returns a specific control option.

See CAPIControl for more information about the object or array of objects returned by these functions.

CAPIGetSingleControlByUse returns a single CAPI control object based on the simple control functions ON/OFF/DIM. The UseType parameter is an Enum as defined in the file HSApplicationAPI.vb. In some cases you may have a control system that simply turns devices ON and OFF. Finding the correct control pair for ON and OFF can be difficult as not all hardware will name the pairs the same. The default pairs for ON/OFF/DIM are noted by the plugin developer with the ePairControlUse enum so they are easy to find.

Example:

The following vb.net code will get the CAPIControl oject to turn a device ON:

```vbnet
Dim objCAPIControl As CAPIControl
Dim dvRef as Integer

dvRef = hs.GetDeviceRefByName("My Device")

objCAPIControl = hs.CAPIGetSingleControlByUse(dvRef,ePairControlUse._On)

If objCAPIControl IsNot Nothing Then
    hs.CAPIControlHandler(objCAPIControl)
End If
```

See also
CAPIGetStatus, CAPIControlHandler, CAPIControlsHandler

CAPI Control

The CAPIControl object holds information about a control state for a device. Generated from value/status, value/graphic pairs, and buttons, CAPIControl objects are used by other plug-ins to render control options for a device properly, and to invoke a control method on a device.

Here are the members of CAPIControl:

```
Public Class CAPIControl
    Inherits MarshalByRefObject
    Public Do_Update As Boolean
    Public SingleRangeEntry As Boolean
    Public Property CCIndex As Integer
    Public Property Range As clsValueRange
    Public ReadOnly Property Ref As Integer
    Public Property Label As String
    Public Property ControlType As Enums.CAPIControlType
    Public Property ControlLocation As Enums.CAPIControlLocation
```
### Public Property

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ControlLoc_Row</strong></td>
<td>This is the Row property of the ControlLocation accessible as an individual integer value.</td>
</tr>
<tr>
<td><strong>ControlLoc_Column</strong></td>
<td>This is the Column property of the ControlLocation accessible as an individual integer value.</td>
</tr>
<tr>
<td><strong>ControlValue</strong></td>
<td>When the ControlType is one of the types that indicate a value is used to achieve the desired control state, this property holds the value to be used.</td>
</tr>
<tr>
<td><strong>ControlString</strong></td>
<td>When the ControlType is one of the types that indicate a string is used to achieve the desired control state, this property holds the string to be used. This property is also used to hold the contents of a custom button assignment which calls a script when pressed.</td>
</tr>
<tr>
<td><strong>ControlStringList</strong></td>
<td>Similar to that of a value being capable of a range, a ControlStringList contains the list of string values which may be set upon the device's string to invoke a change. Use this when the list is dynamic and specific values cannot be assigned to status strings.</td>
</tr>
<tr>
<td><strong>ControlFlag</strong></td>
<td>Used only for rendering control options in a UI, this flag is set to True when a button is added to a device and indicates that a NewLine should be generated after the button so that the next button or control element can start on a new row.</td>
</tr>
<tr>
<td><strong>ControlUse</strong></td>
<td>This property is brought over from the value/status pair, and is used to indicate special control values for lighting devices. See the definition under VSPairs for more information.</td>
</tr>
</tbody>
</table>
clsValueRange

clsValueRange is an object used in the CAPIControl object to hold information about a RANGE value for a device. If, for example, a device can operate within a range from 1 to 99, you can denote this with a value/status pair that contains information about that range, rather than adding 99 value/status pair entries. A device that contains value/status pairs containing a range result in the CAPIControl object containing much the same information, including options for rendering the range properly on a user interface.

Here are the members of clsValueRange:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RangeStart</td>
<td>This is the lowest value possible for this range definition.</td>
</tr>
<tr>
<td>RangeEnd</td>
<td>This is the highest value possible for this range definition.</td>
</tr>
</tbody>
</table>
| RangeStatusDecimals          | This value indicates how many decimal places the value range should be shown with. For example, if the range is 1 to 10 and the number of decimals is 1, then the actual range for purposes of display and selection is 1, 1.1, 1.2, 1.3... 9.8, 9.9, 10.
| RangeStatusPrefix            | This is a prefix to be placed in front of the value when displayed as a status. For example, if you set this to "Dim ", and the RangeStatusDecimals is 1, then the status when at the value 5.235689 will display as "Dim 5.2" |
| RangeStatusSuffix            | This is a suffix to be appended to the end of the value when displayed as a status. For example, given the Prefix example above, set RangeStatusSuffix to "%" and the display will be "Dim 5.2%" |
| RangeStatusValueOffset       | For situations where it is desired to have one range for control (set) and another for status (get), you can use this to indicate an offset from the value to get the desired display result. For example, if the range for controlling a device is 1 to 100 and the prefix/suffix is set to yield a control option such as "Set To 50 Percent" for the value 50, you can establish another range from 101 to 200 for status where the prefix is set to "Dimmed " and the suffix is set to "%", and the RangeStatusValueOffset is set to 100 such that when the value is set to 150, it results in "Dimmed 50%". |
| RangeStatusDivisor           | This value is a divisor applied to the value before it is displayed. For example, if you have a hardware interface that produces values of 10,000 to 100,000, you may wish to represent this as "K" or Kilo rather than displaying all of the digits. To do this, set the suffix to "K" and set the RangeStatusDivisor to 1000. When the value is 55,555 it will result in a display of 55K, or if the RangeStatusDecimals are set to 2, the result would be 55.55K. |
| HasScale                     | When set to True, the range indicates that there is a ScaleReplace (scale replacement) indicator, and when the status is obtained, the scale text provided in the DeviceClass ScaleText property is inserted where the ScaleReplace is found. |
When HasScale is True, HomeSeer will look for the string contained here and will replace it with the string value in the device's ScaleText property. For example, if you have a device that displays temperature but it is not known until runtime whether it will display in Celcius or Fahrenheit, set HasScale to True, and set ScaleReplace to a unique string such as "@S@". Set the suffix to " Degrees @S@", and then when the value is obtained from the device, set ScaleText to the proper scale such as "F", and when the status is obtained the result will be "xx Degrees F".

See also
- `CAPIControlType`
- `CAPIControlLocation`

---

**CAPIControlType**

CAPIControlType is an Enum used within the CAPIControl object.

```vbnet
Enum CAPIControlType
  Not_Specified = 1 'This is the default to use if one of the others is not specified.
  Values = 2
  Single_Text_from_List = 3
  List_Text_from_List = 4
  Button = 5
  ValuesRange = 6 'Rendered as a drop-list by default.
  ValuesRangeSlider = 7
  TextList = 8
  TextBox_Number = 9
  TextBox_String = 10
  Radio_Option = 11
  Button_Script = 12 'Rendered as a button, executes a script when activated.
End Enum
```

See also
- `clsValueRange`
- `CAPIControlLocation`

---

**CAPIControlLocation**

This structure is part of the CAPIControl object, and holds the Row and Column values in a single structure.

```vbnet
Structure CAPIControlLocation
  Public Row As Integer
  Public Column As Integer
End Structure
```

See `CAPIControl` for more information regarding the use of these properties.

See also
- `clsValueRange`
- `CAPIControlType`
CAPIControlHandler, CAPIControlsHandler

Public Function CAPIControlHandler(ByVal CC As CAPIControl) As CAPIControlResponse
Public Function CAPIControlsHandler(ByVal CC() As CAPIControl) As CAPIControlResponse

These functions are used to invoke a control method on a device using a CAPIControl object. Control of a single device to a single control state can be done using CAPIControlHandler, or several devices may be controlled at once using CAPIControlsHandler and passing an array of CAPIControl objects.

In both cases, the return is a single CAPIControlResponse Enum indicating the result of the operation.

See also
CAPIGetStatus
CAPIGetControl, CAPIGetControlEx, CAPIGetSingleControl

CAPIControlResponse

CAPIControlResponse is an Enum which indicates the result of the CAPIControlHandler call made by a plug-in or script. It is defined as:

Public Enum CAPIControlResponse
    Indeterminate = 0
    All_Success = 1
    Some_Failed = 2
    All_Failed = 3
End Enum

See also

Images

Purpose
This procedure...

Parameters

WriteHTMLImage

Purpose

Save an image to the HomeSeer images folder. Useful for plugins that run remotely but have dynamic images that need to be assigned to devices. Use this function to “push” an image to the HomeSeer web server so it can be displayed by web pages or devices. The image passed will put in a subdirectory under the images folder. A directory name must be included in the file path. For example: acme_plugin\image.jpg.

This function accepts an Image object for the image. See WriteHTMLImageFile for a function that accepts an array of bytes.

The only image formats supported are: gif,jpg,png,tif,bmp

Parameters

Parameter: Image
Type: System.Drawing.Image
Description: The image to be saved

Parameter: Dest
Type: String
Description: The relative path to the file like "acme_plugin\image.jpg"

Parameter: Overwrite
Type: Boolean
Description: If true, any existing file will be overwritten

Returns

Return value: Result
Type: Boolean
Description: When True, the procedure was successful.

Example

Private Sub SaveImageFileToHS(src_filename As String, des_filename As String)
Dim im As Drawing.Image = Drawing.Image.FromFile(src_filename)
hs.WriteHTMLImage(im, des_filename, True)
End Sub
WriteHTMLImageFile

Purpose

Save an image to the HomeSeer images folder. Useful for plugins that run remotely but have dynamic images that need to be assigned to devices. Use this function to "push" an image to the HomeSeer web server so it can be displayed by web pages or devices. The image passed will put in a subdirectory under the images folder. A directory name must be included in the file path. For example: acme_plugin\image.jpg.

This function accepts an array of bytes to be saved.

The only image formats supported are: gif,jpg,png,tif,bmp

Parameters

Parameter: ImageFile()
Type: byte
Description: The image to be saved as an array of bytes

Parameter: Dest
Type: String
Description: The relative path to the file like "acme_plugin\image.jpg"

Parameter: Overwrite
Type: Boolean
Description: If true, any existing file will be overwritten

Returns

Return value: Result
Type: Boolean
Description: When True, the procedure was successful.

Example

Private Sub SaveFileToHS(src_filename As String, des_filename As String)
    Dim bytes() As Byte = System.IO.File.ReadAllBytes(src_filename)
    If bytes IsNot Nothing Then
        hs.WriteHTMLImageFile(bytes, des_filename, True)
    End If
End Sub

See also
WriteHTMLImage
See also
About Scripts
Applications and Plugins
Computer
Devices
Events
Internet
Phone
Scripts
Speech Recognition
Strings, Global Variables, and Encryption
Time and Calendar
Text-To-Speech and Media

MailDate

**Purpose**

This function returns the date of the indexed mail message.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>integer</td>
<td>This is the index number of the message to be retrieved.</td>
</tr>
</tbody>
</table>

**Returns**

<table>
<thead>
<tr>
<th>Return value</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>string</td>
<td>This is the date of the indexed mail message.</td>
</tr>
</tbody>
</table>

See also
MailDelete
MailFrom
MailFromDisplay
MailMsgCount
MailSubject
MailText
MailTo
MailToDisplay
MailTrigger
SendEmail

MailDelete

HomeSeer HS3 - End User Documentation
**Purpose**
This function deletes the specified message.

**Parameters**
Parameter: **index**  
Type: **integer**  
Description: This is the index number of the message to be deleted.

**Returns**
None.

See also  
MailDate  
MailFrom  
MailFromDisplay  
MailMsgCount  
MailSubject  
MailText  
MailTo  
MailToDisplay  
MailTrigger  
SendEmail

**MailFrom**

**Purpose**
This function returns the E-mail address of the person who sent the message.

**Parameters**
Parameter: **index**  
Type: **integer**  
Description: This is the index number of the message to be retrieved.

**Returns**
Return value: **from address**  
Type: **string**  
Description: This is the E-mail address of the sender for the indexed message.

See also  
MailDate  
MailDelete  
MailFromDisplay  
MailMsgCount  
MailSubject  
MailText  
MailTo  
MailToDisplay  
MailTrigger  
SendEmail

**MailFromDisplay**
**Purpose**

This function returns the name of the person who sent the message.

**Parameters**

Parameter: **index**  
Type: **integer**  
Description: This is the index number of the message to be retrieved.

**Returns**

Return value: **from address**  
Type: **string**  
Description: This is the name of the sender for the indexed message.

See also

MailDate
MailDelete
MailFrom
MailMsgCount
MailSubject
MailText
MailTo
MailToDisplay
MailTrigger
SendEmail

---

**MailMsgCount**

**Purpose**

This function returns the total number of unread messages that are in your Inbox. This will only work if MAPI is enabled as your E-mail interface (see the E-mail Setup screen).

**Parameters**

None.

**Returns**

Return value: **count**  
Type: **integer**

See also

MailDate
MailDelete
MailFrom
MailFromDisplay
MailSubject
MailText
MailTo
MailToDisplay
MailTrigger
SendEmail

---

**MailSubject**
**Purpose**

This function returns the subject of the specified message.

**Parameters**

Parameter: `index`
Type: `integer`
Description: This is the index number of the message to be retrieved.

**Returns**

Return value: `subject`
Type: `string`
Description: This is the subject of the indexed mail message.

See also

MailDate
MailDelete
MailFrom
MailFromDisplay
MailMsgCount
MailText
MailTo
MailToDisplay
MailTrigger
SendEmail

---

**MailText**

**Purpose**

This function returns the body of the E-mail message.

**Parameters**

Parameter: `index`
Type: `integer`
Description: This is the index number of the message to be retrieved.

**Returns**

Return value: `body`
Type: `string`
Description: This is the body of the E-mail message.

See also

MailDate
MailDelete
MailFrom
MailFromDisplay
MailMsgCount
MailText
MailTo
MailToDisplay
MailTrigger
SendEmail
MailTo

**Purpose**
This function returns the E-mail address of the person the message was sent to.

**Parameters**
- **Parameter:** index
  - **Type:** integer
  - **Description:** This is the index number of the message to be retrieved.

**Returns**
- **Return value:** to address
  - **Type:** string
  - **Description:** This is the E-mail address of the recipient of the indexed message.

See also
- MailDate
- MailDelete
- MailFrom
- MailFromDisplay
- MailMsgCount
- MailSubject
- MailText
- MailToDisplay
- MailTrigger
- SendEmail

MailToDisplay

**Purpose**
This function returns the name of the person the message was sent to.

**Parameters**
- **Parameter:** index
  - **Type:** integer
  - **Description:** This is the index number of the message to be retrieved.

**Returns**
- **Return value:** name
  - **Type:** string
  - **Description:** This is the name of the recipient.

See also
- MailDate
- MailDelete
- MailFrom
- MailFromDisplay
- MailMsgCount
- MailSubject
- MailText
- MailTo
- MailTrigger
- SendEmail
MailTrigger

Purpose
This function returns the index in the MAPI message list of the message that caused the last trigger. If you created an event that triggers when an E-mail is received, then you can run a script as the action to the event. In the script, you call this function to get the index for the message that caused the trigger. You can now examine the message to take more action.

Parameters
None.

Returns
Return value: index
Type: long
Description: This is the index of the E-mail message that was last received.

Example
' this script will access the E-mail message that caused this event to trigger
' it will then speak the subject line of the message
sub main()
    dim index
    dim subject
    index = hs.MailTrigger
    subject = hs.MailSubject(index)
    hs.speak "You have mail! The subject is "
    hs.speak subject
end sub

See also
MailDate
MailDelete
MailFrom
MailFromDisplay
MailMsgCount
MailSubject
MailText
MailTo
MailToDisplay
SendEmail

SendEmail

Purpose
This function will send an E-mail message.

The attach parameter is useful if you would like to E-mail a picture taken with a digital camera. Just attach the path to any picture file created by the camera.

Parameters
Parameter: mto
Type: string
Description: This is the address you are sending the E-mail to.
Parameter: `mfrom`
Type: `string`
Description: This is the address you are sending from. Note that some ISPs will not allow you to put just anything in this field. You may be required to put your real E-mail address here. If you are using MAPI to handle your E-mail, MAPI will enter your E-mail address that is associated with your default E-mail account. In that case, this field will be ignored.

Parameter: `mCC`
Type: `string`
Description: CC address

Parameter: `mBCC`
Type: `string`
Description: This is BCC to address.

Parameter: `msubject`
Type: `string`
Description: This is the subject of the E-mail.

Parameter: `message`
Type: `string`
Description: This is the body of the E-mail.

Parameter: `attach` (optional)
Type: `string`
Description: This is the absolute path name to the file to be attached to the E-mail.

Returns

None.

See also
MailDate
MailDelete
MailFrom
MailFromDisplay
MailMsgCount
MailSubject
MailText
MailTo
MailToDisplay
MailTrigger
Get Information

In This Section

- Event_Group_Info_All
- Event_Group_Info
- Event_Info_All
- Event_Info
- Event_Info_Group
- EventCount
- EventExists
- GetLastEvent

See also
- Get Event References
- Modify Automatic Triggering
- Triggering Events
- Modifying Events
- SetSecurityMode

---

Event_Group_Info_All

**Function** Event_Group_Info_All() As strEventGroupData()

**Purpose**
This function returns information about all of the event groups in the system.

**Parameters**
- Parameter: None

**Returns**
- Return value: strEventGroupData
- Type: array of structure
- Description: This structure is described in this section.

See also
- Event_Group_Info
- Event_Info_All
- Event_Info
- Event_Info_Group
- EventCount
- EventExists
- GetLastEvent
strEventGroupData

This structure is used as a single return value or an array return value for functions that request information about event groups. The description of each member is below.

**Public Structure strEventGroupData**
- **Public GroupID As Integer** ' This is the event group reference ID.
- **Public GroupName As String** ' The name of the group.
- **Public Global_Actions_Count As Integer** ' The number of global actions in this event group.
- **Public Global_Actions As String()** ' The list (array) of global actions in action_type : action_name format.
- **Public Global_Conditions_Count As Integer** ' The number of global conditions in this event group.
- **Public Global_Conditions As String()** ' The list (array) of global conditions in trigger_type : trigger_name format.

End Structure

See also

Event_Group_Info

**Function** Event_Group_Info(ByVal GroupRef As Integer) As strEventGroupData

**Purpose**
This function returns information about a single event group using its Group Reference ID number.

**Parameters**
- **Parameter:** GroupRef
  - **Type:** integer
  - **Description:** This is the group reference ID number for the event group you want to return information about.

**Returns**
- **Return value:** strEventGroupData
  - **Type:** structure
  - **Description:** This structure is described in this section.

See also
Event_Group_Info_All
Event_Info_All
Event_Info
Event_Info_Group
EventCount
EventExists
GetLastEvent

strEventGroupData

This structure is used as a single return value or an array return value for functions that request information about event groups. The description of each member is below.
Public Structure strEventGroupData
    Public GroupID As Integer ' This is the event group reference ID.
    Public GroupName As String ' The name of the group.
    Public Global_Actions_Count As Integer ' The number of global actions in this event group.
    Public Global_Actions As String() ' The list (array) of global actions in action_type : action_name format.
    Public Global_Conditions_Count As Integer ' The number of global conditions in this event group.
    Public Global_Conditions As String() ' The list (array) of global conditions in trigger_type : trigger_name format.
End Structure

See also
Event_Info_All

Function Event_Info_All() As strEventData()

Purpose
This function returns information about all events in the system.

Parameters
Parameter: None

Returns
Return value: strEventData
Type: array of structure
Description: This structure is described in this section.

See also
Event_Group_Info_All
Event_Group_Info
Event_Info
Event_Info_Group
EventCount
EventExists
GetLastEvent

strEventData

This structure is used as a single return or array return from functions providing information about events. Descriptions of each member of the structure are below.

Public Structure strEventData
    Public Event_Ref As Integer ' The event reference ID number.
    Public Event_Name As String ' The event name
    Public Event_Type As String ' The event type, if used.
Public GroupID As Integer  ' The event group reference ID number.
Public GroupName As String  ' The event group name.
Public UserNote As String  ' The user's note contents.
Public Last_Triggered As Date  ' The time the event was last triggered or
    Date.MinValue if it has not been triggered before.
Public Retrigger_Delay As TimeSpan  ' If the event is prevented from triggering within a given amount of time,
    this timespan will contain that time period.
Public Flag_Enabled As Boolean  ' True if the event is enabled for automatic triggering.
Public Flag_Delete_After_Trigger As Boolean  ' True if the event is deleted from the system after it triggers.
Public Flag_Do_Not_Log As Boolean  ' True if the event is set to not log information when it is triggered.
Public Flag_Delayed_Event As Boolean  ' True if the event was created as a result of a delayed action or
    trigger.
    Public Flag_Include_in_Powerfail As Boolean  ' True if the event is to be included in powerfailure recovery.
    Public Flag_Security As Boolean  ' True if the event trigger(s) can be modified by a random amount
        when the security feature is enabled.
    Public Flag_Priority_Event As Boolean  ' True if the event is set to not have its execution queued.
    Public Action_Count  ' The number of actions in this event.
    Public Actions As String()  ' The list (array) of actions in action_type : action_name format.
    Public Trigger_Count As Integer  ' The total number of triggers and conditions in this event.
    Public Trigger_Group_Count As Integer  ' The number of trigger groups (If / Or If) in the event.
    Public Trigger_Groups As strEventTriggerGroupData()  ' The list (array of structure) of triggers in each trigger
        group.
End Structure

See also
strEventTriggerGroupData

strEventTriggerGroupData

This structure, which is used within the strEventData structure, is used by functions that return information about events. The description of each member is below.

Public Structure strEventTriggerGroupData
    Public GroupNumber As Integer  ' The trigger group number for this group of triggers and conditions.
    Public Triggers As String()  ' The list (array) of triggers in this group in trigger_type : trigger_name format.
End Structure

See also
strEventData

Event_Info
Function Event_Info(ByVal evRef As Integer) As strEventData

Purpose
This function returns information about a single event using its Event Reference ID number.

Parameters
Parameter: evRef
Type: integer
Description: This is the event reference ID number for the event you want to return information about.

Returns
Return value: strEventData
Type: structure
Description: This structure is described in this section.

See also
Event_Group_Info_All
Event_Group_Info
Event_Info_Group
EventCount
EventExists
GetLastEvent

strEventData
This structure is used as a single return or array return from functions providing information about events. Descriptions of each member of the structure are below.

Public Structure strEventData
    Public Event_Ref As Integer ' The event reference ID number.
    Public Event_Name As String ' The event name
    Public Event_Type As String ' The event type, if used.
    Public GroupID As Integer ' The event group reference ID number.
    Public GroupName As String ' The event group name.
    Public UserNote As String ' The user's note contents.
    Public Last_Triggered As Date ' The time the event was last triggered or Date.MinValue if it has not been triggered before.
    Public Retrigger_Delay As TimeSpan ' If the event is prevented from triggering within a given amount of time, this timespan will contain that time period.
    Public Flag.Enabled As Boolean ' True if the event is enabled for automatic triggering.
    Public Flag_Delete_After_Trigger As Boolean ' True if the event is deleted from the system after it triggers.
    Public Flag.Do_Not_Log As Boolean ' True if the event is set to not log information when it is triggered.
    Public Flag.Delayed_Event As Boolean ' True if the event was created as a result of a delayed action or trigger.
    Public Flag.Include_in_Powerfail As Boolean ' True if the event is to be included in powerfailure recovery.
    Public Flag.Security As Boolean ' True if the event trigger(s) can be modified by a random amount when the security feature is enabled.
    Public Flag.Priority_Event As Boolean ' True if the event is set to not have its execution queued.
    Public Action_Count As Integer ' The number of actions in this event.
    Public Actions As String() ' The list (array) of actions in action_type : action_name format.
    Public Trigger_Count As Integer ' The total number of triggers and conditions in this event.
    Public Trigger_Group_Count As Integer ' The number of trigger groups (If / Or If) in the event.
    Public Trigger_Groups As strEventTriggerGroupData() ' The list (array of structure) of triggers in each trigger group.
End Structure
strEventTriggerGroupData

This structure, which is used within the strEventData structure, is used by functions that return information about events. The description of each member is below.

Public Structure strEventTriggerGroupData
    Public GroupNumber As Integer  ' The trigger group number for this group of triggers and conditions.
    Public Triggers As String()   ' The list (array) of triggers in this group in trigger_type : trigger_name format.
End Structure

See also
strEventData

Event_Info_Group

Function Event_Info_Group(ByVal GroupID As Integer) As strEventData()

Purpose
This function returns information about all of the events within an event group using the Event Group Reference ID number.

Parameters
Parameter: GroupID
Type: integer
Description: This is the event group reference ID number for the event group you want to return information about.

Returns
Return value: strEventData
Type: structure
Description: This structure is described in this section.

Note:
Use Event_Group_Info_All to find the GroupID for the event group you want to get event information about.

See also
Event_Group_Info_All
This structure is used as a single return or array return from functions providing information about events. Descriptions of each member of the structure are below.

**Public Structure strEventData**

- **Public Event_Ref As Integer**  ' The event reference ID number.
- **Public Event_Name As String**  ' The event name.
- **Public Event_Type As String**  ' The event type, if used.
- **Public EventID As Integer**  ' The event group reference ID number.
- **Public GroupName As String**  ' The event group name.
- **Public UserNote As String**  ' The user's note contents.
- **Public Last_Triggered As Date**  ' The time the event was last triggered or Date.MinValue if it has not been triggered before.
- **Public Retrigger_Delay As TimeSpan**  ' If the event is prevented from triggering within a given amount of time, this timespan will contain that time period.
- **Public Flag_Enabled As Boolean**  ' True if the event is enabled for automatic triggering.
- **Public Flag_Delete_After_Trigger As Boolean**  ' True if the event is deleted from the system after it triggers.
- **Public Flag_Do_Not_Log As Boolean**  ' True if the event is set to not log information when it is triggered.
- **Public Flag_Delayed_Event As Boolean**  ' True if the event was created as a result of a delayed action or trigger.
- **Public Flag_Include_In_Powerfail As Boolean**  ' True if the event is to be included in powerfailure recovery.
- **Public Flag_Security As Boolean**  ' True if the event trigger(s) can be modified by a random amount when the security feature is enabled.
- **Public Flag_Priority_Event As Boolean**  ' True if the event is set to not have its execution queued.
- **Public Action_Count**  ' The number of actions in this event.
- **Public Actions As String()**  ' The list (array) of actions in action_type : action_name format.
- **Public Trigger_Count As Integer**  ' The total number of triggers and conditions in this event.
- **Public Trigger_Group_Count As Integer**  ' The number of trigger groups (If / Or If) in the event.
- **Public Trigger_Groups As strEventTriggerGroupData()**  ' The list (array of structure) of triggers in each trigger group.

**End Structure**

---

**strEventTriggerGroupData**

This structure, which is used within the strEventData structure, is used by functions that return information about events. The description of each member is below.

**Public Structure strEventTriggerGroupData**

- **Public GroupNumber As Integer**  ' The trigger group number for this group of triggers and conditions.
Public Triggers As String() ' The list (array) of triggers in this group in trigger_type : trigger_name format.
End Structure

See also
strEventData

EventCount

Purpose
This function returns the total number of events configured in the system.

Parameters
None.

Returns
Return value: value
Type: integer

See also
Event_Group_Info_All
Event_Group_Info
Event_Info_All
Event_Info
Event_Info_Group
EventExists
GetLastEvent

EventExists

Purpose
This function checks if a given events exists in HomeSeer's event list.

Parameters
Parameter: name
Type: string
Description: This is the name of the event. The name is not case-sensitive.

Returns
Return value: event index
Type: boolean
Description: This returns TRUE if the given event exists and FALSE if it doesn't.

Example
sub main()
    if hs.EventExists("evening") then
        hs.speak "The evening event exists",TRUE
    else
        hs.speak "The evening event does not exist",TRUE
    end if
end sub

See also
Event_Group_Info_All
Event_Group_Info
Event_Info_All
Event_Info
Event_Info_Group
EventCount
GetLastEvent

GetLastEvent

Purpose
This function returns the name of the last event that was triggered. This can be used in a script to detect which event the script was executed from.

Parameters
None.

Returns
Return value: last event
Type: string

Example
sub main()
    dim t
    t = hs.GetLastEvent
    msgbox "This script is run from the event: " & t
end sub

See also
Event_Group_Info_All
Event_Group_Info
Event_Info_All
Event_Info
Event_Info_Group
EventCount
EventExists
Get Event References

In This Section

GetEventRefByName
GetEventRefByNameAndGroup

See also
Get Information
Modify Automatic Triggering
Triggering Events
Modifying Events
SetSecurityMode

GetEventRefByName

Purpose

This function returns the event reference for an event. The event reference is only needed for other procedures which explicitly require the event reference ID.

- This will only return the reference to the first event matching the name provided.

Parameters

Parameter: Name
Type: string
Description: This is the event name excluding the group, such as "Wake-Up Time".

Returns

Return value: reference
Type: integer
Description: This is a numerical event reference ID.

See also
GetEventRefByNameAndGroup

GetEventRefByNameAndGroup

Purpose

This function returns the event reference for an event. The event reference is only needed for other procedures which explicitly require the event reference ID.

- This will only return the reference to the first event matching the name and group provided.

Parameters

Parameter: Name
Type: string
Description: This is the event name excluding the group, such as "Wake-Up Time".

Parameter: Group
Type: string
Description: This is the event name excluding the group, such as "Wake-Up Time".
Description: This is the event group, such as "Morning Events".

**Returns**

Return value: reference  
Type: integer  
Description: This is a numerical event reference ID.

See also

GetEventRefByName

---

Modify Automatic Triggering

In This Section

EnableEvent  
EnableEventByRef  
DisableEvent  
DisableEventByRef

See also

Get Information  
Get Event References  
Triggering Events  
Modifying Events  
SetSecurityMode

---

EnableEvent

**Purpose**

This function marks an event as enabled. All triggers are active.

**Parameters**

Parameter: evname  
Type: string  
Description: This is the event name to enable. Note that the name is not case-sensitive, and the event must have already been disabled.

**Returns**

None.

See also

EnableEventByRef  
DisableEvent  
DisableEventByRef
EnableEventByRef

**Purpose**

This function marks an event as enabled. All triggers are active.

**Parameters**

Parameter: `evref`

Type: `long (.NET Integer)`

Description: This is the event reference ID of the event to enable.

**Returns**

None.

**Example**

```vba
sub main()
    dim eref
    eref = hs.GetEventRefByName("My Event")
    hs.EnableEventByRef eref
end sub
```

See also

EnableEvent
DisableEvent
DisableEventByRef

DisableEvent

**Purpose**

This function marks an event as disabled. All triggers are suspended until the event is re-enabled.

**Parameters**

Parameter: `event`

Type: `string`

Description: This is the event name to disable. Note that the name is not case-sensitive

**Returns**

None.

**Example**

```vba
sub main()
    hs.DisableEvent "evening"
end sub
```

See also

EnableEvent
EnableEventByRef
DisableEventByRef
DisableEventByRef

Purpose
This function marks an event as disabled. All triggers are suspended until the event is re-enabled.

Parameters
Parameter: `evref`
- **Type:** long (.NET Integer)
- **Description:** This is the event reference ID of the event to be disabled.

Returns
None.

Example
```vbnet
sub main()
    dim eref
    eref = hs.GetEventRefByName("My Event")
    hs.DisableEventByRef eref
end sub
```

See also
- EnableEvent
- EnableEventByRef
- DisableEvent

Triggering Events

In This Section
- TriggerEvent
- TriggerEventEx
- DelayTrigger
- TriggerEventAndWait
- RemoveDelayedEvent

See also
- Get Information
- Get Event References
- Modify Automatic Triggering
- Modifying Events
- SetSecurityMode

TriggerEvent
Purpose

This function forces an event to be triggered.

Parameters

Parameter: name
Type: string
Description: This is the name of the event you want to trigger. The actions for the event are executed. Note that the name is not case-sensitive. For instance, the events “Evening” and “evening” would be considered the same.
- If there were duplicate event names, only the first one found would run.

Returns

Return value: status
Type: integer
Description: This is 0 if there was an error or 1 if there was no error. If an error is detected, then an error message is written to the event log.

Example

'Trigger the event named "turn all lights on"
sub main()
    hs.TriggerEvent "turn all lights on"
end sub

See also
TriggerEventEx
DelayTrigger
TriggerEventAndWait
RemoveDelayedEvent

---

TriggerEventEx

Purpose

This function forces an event to be triggered and is used instead of TriggerEvent when it is necessary to specify phone line information at the same time.

Parameters

Parameter: line
Type: integer
Description: This is the phone line number that you wish the event to have been triggered from.

Parameter: name
Type: string
Description: This is the name of the event you want to trigger. The actions for the event are executed. Note that the name is not case-sensitive. For instance, the events “Evening” and “evening” would be considered the same.
- If there were duplicate event names, only the first one found would run.

Parameter: voice command (Optional)
Type: string
Description: If the event processes voice commands, you can provide the string of what the recognized voice command would be that you want the event to process.

Returns

Return value: status
Type: integer
Description: This is 0 if there was an error or 1 if there was no error. If an error is detected, then an error message is written to the event log.

Example

'Trigger the event named "turn all lights on"
sub main()
  hs.TriggerEvent 1, "turn all lights on"
end sub

See also
TriggerEvent
DelayTrigger
TriggerEventAndWait
RemoveDelayedEvent

DelayTrigger

Purpose
This function triggers the given event after the specified number of seconds have elapsed. This is handy if you would like to turn a device on or off a few seconds after the initial event triggers. Note that you can call this as many times as you like, as new events are created and may be viewed and deleted from your events view.

Parameters

Parameter: secs
Type: long
Description: This is the number of seconds before the event name evname is triggered.

Parameter: evname
Type: string
Description: This is the text name of the event that will be triggered.

Returns
None.

Example
sub main()
  ' delay the execution of the event named "lights off" by 5 minutes
  hs.DelayTrigger 300, "lights off"
end sub

See also
TriggerEvent
TriggerEventEx
TriggerEventAndWait
RemoveDelayedEvent

TriggerEventAndWait

Purpose
This function forces an event to be triggered and does not return until the event has completed.

Parameters

See also
TriggerEvent
TriggerEventEx
TriggerEventAndWait
RemoveDelayedEvent
Parameter: **name**  
Type: **string**  
Description: This is the name of the event you want to trigger. The actions for the event are executed. Note that the name is not case-sensitive. For instance, the events "Evening" and "evening" would be considered the same.

- If there were duplicate event names, only the first one found would run.

**Returns**

Return value: **status**  
Type: **integer**  
Description: This is 0 if there was an error or 1 if there was no error. If an error is detected, then an error message is written to the event log.

**Example**

Trigger the event named "turn all lights on" using a VB.NET script.

```vbnet
iReturn = hs.TriggerEventAndWait("turn all lights on")
If iReturn = 0 Then
    hs.WriteLog("Error","There was an error triggering the event: Turn All Lights On")
End If
```

See also  
TriggerEvent  
TriggerEventEx  
DelayTrigger  
RemoveDelayedEvent

---

**RemoveDelayedEvent**

**Purpose**

This function removes a previously queued event from the pending event queue. There are two types of pending events. The first is a device operation and the pending event contains only the house code and unit code of the device. The second is the queuing of an actual event name. In this case, only the name of the event is in the queue. This is typically queued from the DelayTrigger script function.

When a delayed action is used on a device, it will appear under the group "Delayed Actions" in HomeSeer's Events screen.

**Parameters**

Parameter: **device Reference ID**  
Type: **integer**  
Description: This is the device reference number

Parameter: **event_name**  
Type: **string**  
Description: This is the name of the event that is to be removed from the queue.

**Returns**

None.

**Example**

Here is a sample of how to use **RemoveDelayedEvent** to remove a queued device action for the device at address A7:

```
hs.RemoveDelayedEvent 3457,""
```

Here is a sample of how to use **RemoveDelayedEvent** to remove a queued event action for the event named "Reset Dryer Reminder":

```
hs.RemoveDelayedEvent 0, "Reset Dryer Reminder"
```
AddDeviceActionToEvent

Public Function AddDeviceActionToEvent(ByVal evRef As Integer, ByVal CC As CAPIControl) As String

Purpose

This procedure will add a device action to an existing event.

Parameters

Parameter: evRef
Type: Integer
Description: This is the event reference ID number of the event you wish to add a device action to.

Parameter: CC
Type: CAPIControl
Description: The CAPIControl options for a device can be obtained by using CAPIGetControl. Once the CAPIControls of a device are retrieved, find the desired control action, and use that as the CC parameter to this function to have that device action added to the event.

Returns

Return value: Result
Type: String
Description: When empty, the procedure was successful. If this string is not empty, it will contain information about the failure encountered with this function call.

Example

The following script will ...
EventSetRecurringTrigger

Public Function EventSetRecurringTrigger(ByVal evRef As Integer, ByVal Frequency As TimeSpan, ByVal Once_Per_Hour As Boolean, ByVal Reference_To_Hour As Boolean) As Boolean

Purpose

This procedure sets the trigger on an existing event to be a recurring time trigger.

Parameters

Parameter: evRef
Type: Integer
Description: This is the event reference ID number of the event that you wish to set to a recurring trigger.

Parameter: Frequency
Type: TimeSpan
Description: The time period held within this parameter will be the recurrence frequency for the event. The TimeSpan should not have any value for the Days part and the Hours part value should be less than 24.

Parameter: Once_Per_Hour
Type: Boolean
Description: If set to True, the event will only trigger once per hour even if the Frequency is less than one hour.

Parameter: Reference_To_Hour
Type: Boolean
Description: When set to True, the Frequency will be calculated from the top of the hour rather than the previous trigger time or the first trigger time.

Returns

Return value: Result
Type: Boolean
Description: When True, the procedure was successful.

Example

See also
AddDeviceActionToEvent
EventSetTimeTrigger
EventSetVRTrigger
NewEventEx
NewEventGetRef
SaveEventsDevices
DeleteEvent

EventSetTimeTrigger

Public Function EventSetTimeTrigger(ByVal evRef As Integer, ByVal DT As Date) As Boolean
Purpose

This procedure will set a time trigger on an event. The trigger is always the first trigger in the first trigger group. If the event already has a trigger in the first trigger group, the group will be wiped out and replaced with this time trigger. The date component of the date parameter is ignored.

Parameters

Parameter: evRef
Type: Integer
Description: This is the event reference ID number for the event you wish to set a time trigger on.

Parameter: DT
Type: Date
Description: This DATE data type is used to set the time you wish to trigger the event at. The date part of the DT parameter is ignored.

Returns

Return value: Result
Type: Boolean
Description: When True, the procedure was successful.

Example

See also
AddDeviceActionToEvent
EventSetRecurringTrigger
EventSetVRTrigger
NewEventEx
NewEventGetRef
SaveEventsDevices
DeleteEvent

EventSetVRTrigger

Public Function EventSetVRTrigger(ByVal evRef As Integer, ByVal VR As String, _
                                ByVal VRFor As Enums.enumVCMDType) As Boolean

Purpose

This procedure sets the trigger on an existing event to be a voice recognition string.

Parameters

Parameter: evRef
Type: Integer
Description: This is the event reference ID number of the event that you wish to set to the VR trigger.

Parameter: VR
Type: String
Description: The voice recognition string to use to trigger the event on matched voice phrases.

Parameter: VRFor
Type: enumVCMDType
Description: This specifies on what device(s) the VR string is valid to come from/for: 0=Disabled, 1=Microphone, 2=Telephone, 3=Both.

Returns

Return value: Result
Type: Boolean
Description: When True, the procedure was successful.

Example
See also
AddDeviceActionToEvent
EventSetRecurringTrigger
EventSetTimeTrigger
NewEventEx
NewEventGetRef
SaveEventsDevices
DeleteEvent

NewEventEx

Public Function NewEventEx(ByVal Name As String, ByVal Group As String, ByVal sType As String) As Integer

Public Function NewEventGetRef(ByVal Name As String, ByVal Group As String, ByVal sType As String) As Integer

Purpose
This function creates a new empty event. The trigger mode is set to MANUAL and the event is DISABLED if the setup option "New Events are Disabled by Default" is enabled. All other properties are cleared and the name is set to the name given. The EventClass object reference to the new event is returned and may be used in a script to set properties of the new event.

Parameters

Parameter: Name
Type: string
Description: This is the name of the new event.

Parameter: Group
Type: string
Description: This is the name of the event group you want the new event created in. If the event group does not exist, it will be created.

Parameter: sType
Type: string
Description: This is the type description for the event that you wish to use.

Returns

Return value: event reference
Type: Integer
Description: This is the event reference ID number.

Example

See also
AddDeviceActionToEvent
EventSetRecurringTrigger
EventSetTimeTrigger
EventSetVRTrigger
NewEventGetRef
SaveEventsDevices
DeleteEvent

NewEventGetRef

Public Function NewEventEx(ByVal Name As String, ByVal Group As String, ByVal sType As String) As Integer
Public Function NewEventGetRef(ByVal Name As String, ByVal Group As String, ByVal sType As String) As Integer

Purpose

This function creates a new empty event. The trigger mode is set to MANUAL and the event is DISABLED if the setup option "New Events are Disabled by Default" is enabled. All other properties are cleared and the name is set to the name given. The EventClass object reference to the new event is returned and may be used in a script to set properties of the new event.

Parameters

Parameter: Name
Type: string
Description: This is the name of the new event.

Parameter: Group
Type: string
Description: This is the name of the event group you want the new event created in. If the event group does not exist, it will be created.

Parameter: sType
Type: string
Description: This is the type description for the event that you wish to use.

Returns

Return value: event reference
Type: Integer
Description: This is the event reference ID number.

Example

See also
AddDeviceActionToEvent
EventSetRecurringTrigger
EventSetTimeTrigger
EventSetVRTrigger
NewEventEx
SaveEventsDevices
DeleteEvent

SaveEventsDevices

SaveEventsDevices

Purpose

If an event or device was modified by a script, this function should be called to update HomeSeer with the changes. For example, if you change a voice command in an event, calling this function tells HomeSeer to re-initialize the voice recognition so the new voice command is available to the user. This function will also update all displays in the Control screen.

Parameters

None.

Returns

None.

See also
AddDeviceActionToEvent
EventSetRecurringTrigger
EventSetTimeTrigger
DeleteEvent

DeleteEvent

Purpose

This function deletes the specified event.

Parameters

Parameter: evname
Type: string
Description: This is the event name to delete. Note that the name is not case-sensitive.

Returns

None.

Example

sub main()
    hs.DeleteEvent "evening"
end sub

See also
AddDeviceActionToEvent
EventSetRecurringTrigger
EventSetTimeTrigger
EventSetVRTrigger
NewEventEx
NewEventGetRef
SaveEventsDevices

SetSecurityMode

SetSecurityMode

Purpose

This function enables or disables the security mode. When security mode is enabled, the event trigger time is randomly set to plus or minus 30 minutes from the actual set time. This can give your home a lived-in look because lights and other devices won't be turned on at the same time day after day.

Parameters

Parameter: mode
Type: integer
Description: Use 0 to disable the security mode and 1 to enable it.

Returns

None.

Example
The following script statement will enable security mode.

```
hs.SetSecurityMode 1
```

See also
- Get Information
- Get Event References
- Modify Automatic Triggering
- Triggering Events
- Modifying Events

---

Home > Scripting > Internet

Internet

In This Section

- FTP
- GetURL
- GenCookieString

See also
- About Scripts
- Applications and Plugins
- Computer
- Devices
- Email
- Events
- Phone
- Scripts
- Speech Recognition
- Strings, Global Variables, and Encryption
- Time and Calendar
- Text-To-Speech and Media

---

Home > Scripting > Internet > FTP

FTP

In This Section

- FTP
- FTPLastError
- SetRemoteTimeout

See also
- GetURL
- GenCookieString

---

Home > Scripting > Internet > FTP > FTP
FTP

Purpose
This function gives access to ftp servers. This command is used mostly for downloading files using the ftp protocol. Use FTPLastError to check for errors executing this command.

When using the get and put commands, the local_file and remote_file parameters must be valid. For a put command, the command will copy the file at the path given in the local_file parameter to the path given in the remote_file parameter. Note that the remote_file specification should not include any path information. Set the path parameter to the correct path for the file. For get commands, the file at the remote_file location is copied to the file at the local_file location.

For the del command, the file at the remote_file location is deleted.

The dir command returns the directory as a string.

The rename command uses local_file as the old name, and remote_file as the new name.

Parameters

Parameter: host
Type: string
Description: This is the name or IP address of host to connect to, such as HomeSeer.com.

Parameter: username
Type: string
Description: This is the username for access to the server.

Parameter: password
Type: string
Description: This is the password for access to the server.

Parameter: command
Type: string
Description: This can be one of the following FTP commands: put, get, del, dir, or rename.

Parameter: path
Type: string
Description: This is the path to the file, such as public.

Parameter: local_file
Type: string
Description: This is the file name where the downloaded file will be saved.

Parameter: remote_file
Type: string
Description: This is the name of the file on the remote server to download.

Returns

Return value: depends on command
Type: string
Description: For the dir command, a directory listing is returned. For all other commands, if no error occurs an empty string is returned, else an error message is returned which starts with the text "ERROR".

Example
sub main()
    dim s
    dim host
    dim user
    dim password
    dim command
    dim rfile
    dim lfile
    dim path
    host = "homeuser.com"
    user = "anonymous"
    password = "user@company.com"
    command = "get"
    rfile = "remote_test.htm"
    lfile = "c:\remote_test.htm"
    path = "pub"
    s = hs.ftp(host, user, password, command, path, lfile, rfile)
end sub

See also
FTPLastError
SetRemoteTimeout

FTPLastError

Purpose
This command should be used after an FTP command to check for any errors that may have been encountered with the FTP command. A null (empty) return indicates that the command completed successfully.

Parameters
None.

Returns
Return value: Error
Type: string
Description: This is the text of the last FTP command error.

See also
FTP
SetRemoteTimeout

SetRemoteTimeout

Purpose
This function sets the number of seconds to wait for a remote host to respond when using hs.GetURL, hs.GetURLIE, or hs.ftp.

- If this function is never called, the remote timeout is set to 60 seconds.

Parameters
Parameter: **timeout**  
Type: `integer`  
Description: This is the number of seconds to wait.

**Returns**  
None.

**Example**

```plaintext
' set the remote timeout to 30 seconds
hs.SetRemoteTimeout 30
```

See also  
FTP  
FTPLastError

---

**GetURL**

**Purpose**

This function returns a web page. This is useful for retrieving pages like news and weather, and then having a Speaker Client speak the contents for you. This method can also be used to retrieve images, including JPG or GIF images.

**Parameters**

- **Parameter:** `host`  
  Type: `string`  
  Description: This is the name or IP address of host to connect to, such as "HomeSeer.com".

- **Parameter:** `page`  
  Type: `string`  
  Description: This is the page to retrieve from the server, such as "/news.htm".

- **Parameter:** `strip_tags`  
  Type: `boolean`  
  Description: Use TRUE to strip HTML tags from the returned page or FALSE to not alter the page.

- **Parameter:** `port`  
  Type: `integer`  
  Description: This is the port number on the server to connect with (80 = standard web server).
Optional Parameter: **UTF8**
Type: boolean
Description: When set to TRUE, the HomeSeer will decode the data received from the web server using UTF-8. If this is FALSE or is not provided, then the data will be decoded using the default encoding (usually Windows-1252).

**Returns**

Return value: page contents
Type: string
Description: This is the contents of the requested web page. If an error occurs, the text "ERROR:" will be returned followed by a reason for the error.

**Example**

```vbs
sub main()
    dim page
    page = hs.GetURL("HomeSeer.com","/",TRUE,80)
    msgbox page
end sub
```

**See also**
GetURLEx
GetURLIE
GetURLImage
GetURLImageEx
URLAction
SetRemoteTimeout

GetURLEx

**Purpose**

This function returns a web page. This is useful for retrieving pages like news and weather, and then having a Speaker Client speak the contents for you. This method can also be used to retrieve images, including JPG or GIF images. This function has extended features compared to the similar function GetURL - it can return a byte array, which is useful when retrieving binary data, and it attempts to decode the page data encoding so that the data is properly decoded using Windows-1252 or UTF-8.

**Parameters**

Parameter: host
Type: string
Description: This is the name or IP address of host to connect to, such as "HomeSeer.com".

Parameter: page
Type: string
Description: This is the page to retrieve from the server, such as "/news.htm".

Parameter (ByRef): ElapsedTime
Type: string
Description: Provide an empty string variable for this parameter, and when the procedure is finished, it will contain a formatted string with the total time the page download required.

Optional Parameter: port
Type: integer
Description: This is the port number on the server to connect with (80 = standard web server). If this parameter is not provided, a value of 80 will be used.

Optional Parameter: strip_tags
Type: boolean
Description: Use TRUE to strip HTML tags from the returned page or FALSE to not alter the page.

Optional Parameter: ByteArray
Type: boolean
Description: If set to TRUE, then the return from the function will be an array of bytes instead of a string.

Optional Parameter: FileName
Type: string
Description: If this parameter is not null or empty, then the downloaded web page will be automatically saved into the file named with this parameter.
HomeSeer must have write access to the directory where the file is to be placed. An existing file by the same name will be OVERWRITTEN.

**Returns**

Return value: **page contents**  
Type: *string* or *byte array*  
Description: This is the contents of the requested web page. If an error occurs, the text “ERROR:” will be returned followed by a reason for the error.

See also  
GetURL  
GetURLIE  
GetURLImage  
GetURLImageEx  
URLAction  
SetRemoteTimeout

---

**GetURLIE**

**Purpose**

This function returns a web page using Internet Explorer. Note that only the HTML of the page is returned, but the entire page will be downloaded from the specified website. If the page contains any sounds, the sounds may be played out your computer speakers. Try using `hs.GetURL` before using this function.

**Parameters**

- **host**  
  Type: *string*  
  Description: This is the name or IP address of the host to connect to, such as "HomeSeer.com".

- **strip_tags**  
  Type: *boolean*  
  Description: Use TRUE to strip HTML tags from the returned page or FALSE to not alter the page.

**Returns**

Return value: **page contents**  
Type: *string*  
Description: This is the contents of the requested web page. If an error occurs, the text “ERROR:” will be returned followed by a reason for the error.

See also  
GetURL  
GetURLEx  
GetURLImage  
GetURLImageEx  
URLAction  
SetRemoteTimeout

---

**GetURLImage**

**Purpose**

This function returns a web page image file. This is useful for retrieving pages like weather satellite maps, and then displaying the maps in a HomeSeer device.

**Parameters**

See also  
GetURL  
GetURLEx  
GetURLImage  
GetURLImageEx  
URLAction  
SetRemoteTimeout
Parameter: **host**  
Type: *string*  
Description: This is the name or IP address of host to connect to, such as "HomeSeer.com".

Parameter: **page**  
Type: *string*  
Description: This is the image to retrieve from the server, such as "/logo.gif". It should be fully qualified as referenced from the host parameter above, such as "/images/something/other/logo.gif" if necessary.

Optional Parameter: **strip_tags**  
Type: *boolean*  
Description: This parameter is ignored in GetURLImage.

Optional Parameter: **port (Default=80)**  
Type: *integer*  
Description: This is the port number on the server to connect with (80 = standard web server).

Optional Parameter: **filename**  
Type: *string*  
Description: This is the file that you would like the downloaded image to be stored in. If the filename is not fully qualified, then the HomeSeer path will be prepended to the string provided. This is recommended for VBS scripts to prevent trying to work with the byte array return which cannot be written to a file easily using VBS script accessible objects.

**Returns**

Return value: **page image**  
Type: *byte array (.NET Object or VBScript Variant)*  
Description: This is the contents of the requested web page image. If an error occurs, the text "ERROR:" will be returned followed by a reason for the error.

See also  
GetURL  
GetURLEx  
GetURLIE  
GetURLImageEx  
URLAction  
SetRemoteTimeout

---

**GetURLImageEx**

**Purpose**

This function returns a web page image file and saves it in the file specified. This is useful for retrieving pages like weather satellite maps, and then displaying the maps in a HomeSeer device.

**Parameters**

Parameter: **host**  
Type: *string*  
Description: This is the name or IP address of host to connect to, such as "HomeSeer.com".

Parameter: **page**  
Type: *string*  
Description: This is the image to retrieve from the server, such as "/logo.gif". It should be fully qualified as referenced from the host parameter above, such as "/images/something/other/logo.gif" if necessary.

Parameter: **filename**  
Type: *string*  
Description: This is the file that you would like the downloaded image to be stored in. If the filename is not fully qualified, then the HomeSeer path will be prepended to the string provided.

Optional Parameter: **port (Default=80)**  
Type: *integer*  
Description: This is the port number on the server to connect with (80 = standard web server).

**Returns**

Return value: **result**  
Type: *string*
Description: This is the result of the operation - if empty, the operation was successful. If the return value is not empty, then it will be an error message.

See also
GetURL
GetURLEx
GetURLIE
GetURLImage
URLAction
SetRemoteTimeout

URLAction

Purpose
This command provides access to HomeSeer's internal Internet control. With it, you can post data to a web server, or get the headers from a web page.

Following are a few simple examples of the various “actions” that can be performed with this command:

**POST**
(Posts data to the server)

```dim s
cost server_url = "http://someserver.com/datapost/hereitis.html"
cost headers="Content-Type: application/x-www-form-urlencoded"
s = hs.URLAction(server_url, "POST", data, headers)
```

**GET**
(Retrieves a web page - see GetURL)

```dim s
dim data
cost website = "http://www.google.com/search?sourceid=navclient&ie=UTF-8&oe=UTF-8&q=homeseer"
s = hs.URLAction(website, "GET", "", "")
```

**HEAD**
(Retrieves web page headers)

```dim s
s = hs.URLAction("http://someserver.com/data/homepage.htm", "HEAD", "", "")
```

**PUT**
(Replaces [puts] a file at the URL)

```dim s
dim sPage
dim sHead
sPage = (routine to read a file into the variable sPage)
s = hs.URLAction("http://someserver.com/putithere/putit.htm", "PUT", sPage, "")
```

Parameters

Parameter: **url**
Type: string
Description: This is the URL to post to.

Parameter: **action**
Type: string
Description: This is the action for the URLAction command, which is one of POST, PUT, HEAD, or GET.

Parameter: **data**
Type: string
Description: This is the URL data parameters.
SetRemoteTimeout

**Purpose**

This function sets the number of seconds to wait for a remote host to respond when using hs.GetURL, hs.GetURLIE, or hs.ftp.

- If this function is never called, the remote timeout is set to 60 seconds.

**Parameters**

Parameter: **timeout**

Type: **integer**

Description: This is the number of seconds to wait.

**Returns**

None.

**Example**

```javascript
' set the remote timeout to 30 seconds
hs.SetRemoteTimeout 30
```

See also

GetURL
GetURLEx
GetURLIE
GetURLImage
GetURLImageEx
SetRemoteTimeout

---

GenCookieString

**Purpose**

This function returns a properly formatted cookie string given the parameters supplied to the function. The cookie string can then be appended to the head section of an HTML page being generated for the browser to take the appropriate cookie action. The string is created as a META set-cookie tag.

---

Parameter: **headers**

Type: **string**

Description: This is the web page headers.

**Returns**

Return value: **web page**

Type: **string**

Description: This is the returned web page from the URLAction command, if any.

See also

GetURL
GetURLEx
GetURLIE
GetURLImage
GetURLImageEx
URLAction
Parameters

Parameter: name
Type: string
Description: This is the name of the cookie. If commas, whitespace, semi-colons or other non-HTML friendly characters are used, then it should be noted that the name string is URL encoded in the returned string, so the name used to retrieve the cookie may have to be updated to use the URL encoded version.

Parameter: value
Type: string
Description: This is the value of the cookie. If commas, whitespace, semi-colons or other non-HTML friendly characters are used, then it should be noted that the value string is URL encoded in the returned string, so the value returned when you read the cookie back may have to be URL decoded. (See System.Web.HTTPServerUtility.URLDecode)

Parameter: expire (optional)
Type: string
Description: This is the expiration date and time for the cookie. If this parameter is omitted, then no expiration will be provided and the cookie will be erased when the web browser session ends. The date can be in any string format which can be converted by DateTime.Parse such as "April 1, 2006 11:27 PM". You may also use your system's local date/time format converted to a string value. The date provided is converted to GMT for purposes of formatting according to RFCs 822, 850, 1036 and 1123.

Parameter: path (optional)
Type: string
Description: This is the path that the cookie is valid for under the server host. If a cookie has already passed domain matching, then the pathname component of the URL is compared with the path attribute, and if there is a match, the cookie is considered valid and is sent along with the URL request. The path "/foo" would match "/foo/bar" and "/foo/bar.html". The path "/" is the most general path and is the default value if this parameter is omitted.

Returns

Return value: cookie contents
Type: string
Description: This is the contents of the requested cookie.

Example

This script:

```vbscript
Sub Main(parm as object)
  Dim sCookie As String = ""
  hs.WriteLog("Cookie", sCookie)
End Sub
```

Generates this result:

```
9/21/2006 7:01:50 PM - Cookie - <meta http-equiv="Set-Cookie" content="Test=MyValue; expires=Sun, 01-Oct-2006 23:01:50 GMT; path=/;">```

See also
- FTP
- GetURL
See also
About Scripts
Applications and Plugins
Computer
Scripting_Phone_LINEClearDTMF

**LINEClearDTMF**

**Purpose**

This function clears both the DTMF counter and the associated buffer.

**Parameters**

- **Parameter:** Line
- **Type:** Integer
  - **Description:** The phone line to clear.

**Returns**

None.

See also

- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_LINESetVoice
- Scripting_Phone_LINESetRingsCurrent
- Scripting_Phone_LINESetSpeakingSpeed
- Scripting_Phone_LINESetRings
- Scripting_Phone_LINESetGreeting
- Scripting_Phone_LINESetCIDNumber
- Scripting_Phone_LINESetCIDName
- Scripting_Phone_LINESetCIDInfo
- Scripting_Phone_LINESetAnswerMode
WaitMS

Purpose

This function waits the number of specified milliseconds. The application still processes events, but will sleep so the script does not use all the CPU.

Parameters

Parameter: Millisecs
Type: Integer
Description: The number of milliseconds to wait.

Returns

None.

Examples

' wait 2 seconds
  hsp.WaitMS 2000

See also

Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINEisSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_PhoneILINE
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_ContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByName
Scripting_Phone_MBGet
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBDelteMessage
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESetSendTones
Scripting_Phone_LINESetSendDT
Scripting_Phone_LINESetHasControl
Scripting_Phone_LINESetRingCount
Scripting_Phone_LINESetCallTimeout
Scripting_Phone_LINESetReset
Scripting_Phone_LINESetRecordStop
Scripting_Phone_LINESetRecordStart
Scripting_Phone_LINESetMuteRings
Scripting_Phone_LINESetIsSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINESetDTMFString
Scripting_Phone_LINESetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINESetDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_Linedial
Scripting_Phone_LINESetAnswerSpeakerPhone
Scripting_Phone_LINESetAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_CreateContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCnt
StopListening

Purpose
This function disabled the recognition engine the current line. No voice recognition will take place. A call must be in progress.

If you would like your script to work over a microphone as well as over the phone, use the system version of this command.

Parameters
Parameter: Line
Type: Integer
Description: The phone line to access.

Returns
None.

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessagetime
Scripting_Phone_MBMessagename
Scripting_Phone_MBMessagelength
Scripting_Phone_MBMessagefrom
Scripting_Phone_MBMessagedate
Scripting_Phone_MBMarkunread
Scripting_Phone_MBMarkread
Scripting_Phone_MBMGetloggedin
Scripting_Phone_MBMGetdefault
Scripting_Phone_MBMGetbyname
Scripting_Phone_MBMGet
Scripting_Phone_MBFirstunreadmessage
Scripting_Phone_MBFirstrmessage
Scripting_Phone_MBDeltemessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelpendingnotifications
Scripting_Phone_MBAnswerMode
Scripting_Phone_Mailboxclass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINESpkspeaking
Scripting_Phone_LINEShangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFSstring
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_StartListening

StartListening

Purpose

This function enables the recognition engine to start listening on the current line. A call must be in progress.

If you would like your script to work over a microphone as well as over the phone, use the system version of this command.

Parameters

Parameter: **Line**
Type: **Integer**
Description: The phone line to access.

Returns

None.

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLogedin
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByName
Scripting_Phone_MBGet
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBDelteMessage
Scripting_Phone_MBCount
Scripting_Phone_Speak

Speak

Purpose

This function speaks text or a WAV file over the phone line given.

Parameters

Parameter: Line
Type: Integer
Description: The phone line to speak to.

Parameter: Text
Type: String
Description: The text to speak. This may also be the full path to a WAV file to play over the phone.

Parameter: Wait
Type: **Boolean**
Description: Boolean value that causes the function to not return if set to TRUE. If set to FALSE, the speaking text is queued and the function returns immediately.

**Returns**

None.

**See Also**

*Using Replacement Variables*

See also
- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MBNew
- Scripting_Phone_MBMessageTime
- Scripting_Phone_MBMessageName
- Scripting_Phone_MBMessageLength
- Scripting_Phone_MBMessageFrom
- Scripting_Phone_MBMessageDate
- Scripting_Phone_MBMarkUnRead
- Scripting_Phone_MBMarkRead
- Scripting_Phone_MBClearLogIn
- Scripting_Phone_MBGetDefault
- Scripting_Phone_MBGetByName
- Scripting_Phone_MBGet
- Scripting_Phone_MBFirstUnreadMessage
- Scripting_Phone_MBFirstReadMessage
- Scripting_Phone_MBDelteMessage
- Scripting_Phone_MBClearPendingNotifications
- Scripting_Phone_MBAnswerMode
- Scripting_Phone_MailboxClass
- Scripting_Phone_LINEStopSpeaking
- Scripting_Phone_LINEStatus
- Scripting_Phone_LINESetVoice
- Scripting_Phone_LINESetRingCurrent
- Scripting_Phone_LINESetSpeakingSpeed
- Scripting_Phone_LINESetRings
- Scripting_Phone_LINESetGreeting
- Scripting_Phone_LINESetCIDNumber
- Scripting_Phone_LINESetCIDName
- Scripting_Phone_LINESetCIDInfo
- Scripting_Phone_LINESetAnswerMode
- Scripting_Phone_LINESendTones
- Scripting_Phone_LINESendAT
- Scripting_Phone_LINEScriptHasControl
- Scripting_Phone_LINERingCount
- Scripting_Phone_LINEResetCallTimeout
- Scripting_Phone_LINEReset
- Scripting_Phone_LINERecordStop
- Scripting_Phone_LINERecordStart
- Scripting_Phone_LINEmuteRings
- Scripting_Phone_LINEisSpeaking
- Scripting_Phone_LINEHangup
- Scripting_Phone_LINEGetVoice
- Scripting_Phone_LINEGetDTMFString
- Scripting_Phone_LINEGetDTMFCount
- Scripting_Phone_LINEEnableSpeakerPhone
- Scripting_Phone_LINEDisableSpeakerPhone
- Scripting_Phone_LINECount
- Scripting_Phone_LINEReset
- Scripting_Phone_LINEAnswerSpeakerPhone
- Scripting_Phone_LINEAnswerLocal
- Scripting_Phone_LINEReset
- Scripting_Phone_LastVoiceMailInfo
- Scripting_Phone_LastCallerInfo
- Scripting_Phone_HIPSetCallWaitingLED
- Scripting_Phone_HIPSendLocalCID

*HomeSeer HS3 - End User Documentation*
Scripting_Phone_SetSpeaker

SetSpeaker

Purpose
This function sets the speaker for voice recognition. If you train multiple users for the voice recognition, you can use this function to switch to their profile. Call this function before performing any training. Training should be done in the HomeSeer application as there is no way to train over the phone.

Parameters
Parameter: Line
Type: Integer
Description: The number of the telephone line to set the speaker on. Different speakers may be set for each line.

Parameter: Name
Type: String
Description: The name of the user's profile to switch to.

Returns
None.

Examples
' set the speaker to "bill"
hspt.SetSpeaker 1,"bill"

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MSBGetLoggedIn
Scripting_Phone_MSBGetByName
Scripting_Phone_MSBGet
Scripting_Phone_MSBFirstUnreadMessage
Scripting_Phone_MSBFirstReadMessage
**RestoreSettings**

**Purpose**

This function restores all program settings from the `settings.ini` file. To make some settings active, you must call `hsp.LINEReset` so the appropriate modem driver is reset to the new settings.

**Parameters**

None.

**Returns**

None.
See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBmessageTime
Scripting_Phone_MBmessageLength
Scripting_Phone_MBmessageFrom
Scripting_Phone_MBmessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetcName
Scripting_Phone_MBfirstUnreadMessage
Scripting_Phone_MBfirstReadMessage
Scripting_Phone_MBDeltemessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAwserMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswer
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINELast
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_CONTACTClass
Scripting_Phone_CLEARLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADGGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount
Scripting_Phone_MBSort

MBSort

Purpose
This function updates the mailbox status and sorts voice messages by date. Should be called at least once before calling MBGet to get the voice messages.

Parameters
None.

Returns
None.

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessagetime
Scripting_Phone_MBMessagename
Scripting_Phone_MBMessagelength
Scripting_Phone_MBMessagefrom
Scripting_Phone_MBMessagedate
Scripting_Phone_MBMarkUnread
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedin
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByName
Scripting_Phone_MBGet
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBDropMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendDAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINESendCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINEIsSpeaking
Scripting_Phone_MBSave

MBSave

Purpose

This function saves all configured mailbox information. Useful if a script modifies any properties of a mailbox.

Parameters

None.

Returns

None.

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIN
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGet
Scripting_Phone_MBFIRSTUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBDelMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESetHasControl
Scripting_Phone_LINEResetCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINEIsSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEIsLocal
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_LastCallerInfo
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount

Scripting_Phone_MBNextUnreadMessage

MBNextUnreadMessage

Purpose
This function returns the next unread message in the given mailbox. To get all the unread messages, call MBFirstUnreadMessage, then call MBNextUnreadMessage.

Parameters
Parameter: Line
Type: Integer
Description: The phone line to access.

Parameter: mb
Type: Mailbox
Description: The mailbox to access. This is a reference to a mailbox object. Use MBGet to get a mailbox.

Returns

Return value: File name
Type: String
Description: The file name of the voice message. Note that this file name does not include the full path to the file. Voice messages are saved in the directory messages in the HomeSeer application directory. To create the full path to the file, use the GetAppPath function like:

```plaintext
path = hsp.GetAppPath + "\messages\" + message
```

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBSave
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedln
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByName
Scripting_Phone_MBGet
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBDel eteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendDAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINEIsSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFFrom
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount

**Scripting_Phone_MBNextReadMessage**

**Purpose**

This function returns the next read message in the given mailbox. To get all the read messages, call `MBFirstReadMessage`, then call `MBNextReadMessage`.

**Parameters**

- **Parameter**: Line
  - **Type**: Integer
  - **Description**: The phone line to access.

- **Parameter**: mb
  - **Type**: Mailbox
  - **Description**: The mailbox to access. This is a reference to a mailbox object. Use `MBGet` to get a mailbox.

**Returns**

- **Return value**: File name
  - **Type**: String
  - **Description**: The file name of the voice message. Note that this file name does not include the full path to the file. Voice messages are saved in the `messages` directory in the HomeSeer application directory. To create the full path to the file, use `GetAppPath` function like:

```
path = hsp.GetAppPath & "\messages\" & message
```

**See also**

- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNew
- Scripting_Phone_MBMessageTime
- Scripting_Phone_MBMessageName
- Scripting_Phone_MBMessageLength
- Scripting_Phone_MBMessageFrom
- Scripting_Phone_MBMessageDate
- Scripting_Phone_MBMarkUnRead
- Scripting_Phone_MBMarkRead
- Scripting_Phone_MBGetLoggedIn
- Scripting_Phone_MBGetDefault
- Scripting_Phone_MBGetByName
- Scripting_Phone_MBGet
- Scripting_Phone_MBfirstUnreadMessage
- Scripting_Phone_MBFirstReadMessage
- Scripting_Phone_MBDelteMessage
- Scripting_Phone_MBCnt
- Scripting_Phone_MBCancelPendingNotifications
**Scripting_Phone_MBNew**

**MBNew**

**Purpose**

This function creates a new empty mailbox.

**Parameters**

None.

**Returns**

Return value: **Reference**
Type: **Mailbox**
Description: A reference to a new mailbox

**Examples**
See also
- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MBMessageTime
- Scripting_Phone_MBMessageName
- Scripting_Phone_MBMessageLength
- Scripting_Phone_MBMessageFrom
- Scripting_Phone_MBMessageDate
- Scripting_Phone_MBMarkUnRead
- Scripting_Phone_MBMarkRead
- Scripting_Phone_MBGetLogFile
- Scripting_Phone_MBGetDefault
- Scripting_Phone_MBGetByUserName
- Scripting_Phone_MBGet
- Scripting_Phone_MBFirstUnreadMessage
- Scripting_Phone_MBFirstReadMessage
- Scripting_Phone_MBDeleteMessage
- Scripting_Phone_MBCancelPendingNotifications
- Scripting_Phone_MBGetAnswerMode
- Scripting_Phone_MailboxClass
- Scripting_Phone_LINEStopSpeaking
- Scripting_Phone_LINEStatus
- Scripting_Phone_LINESetVoice
- Scripting_Phone_LINESetRingsCurrent
- Scripting_Phone_LINESetSpeakingSpeed
- Scripting_Phone_LINESetRings
- Scripting_Phone_LINESetGreeting
- Scripting_Phone_LINESetCIDNumber
- Scripting_Phone_LINESetCIDName
- Scripting_Phone_LINESetCIDInfo
- Scripting_Phone_LINESetAnswerMode
- Scripting_Phone_LINESendDTMFString
- Scripting_Phone_LINEGetDTMFString
- Scripting_Phone_LINEGetDTMFCount
- Scripting_Phone_LINEEnableSpeakerPhone
- Scripting_Phone_LINEDisableSpeakerPhone
- Scripting_Phone_LINECount
- Scripting_Phone_LINEDial
- Scripting_Phone_LINEAnswerSpeakerPhone
- Scripting_Phone_LINEAnswerLocal
- Scripting_Phone_LINEAnswer
- Scripting_Phone_LastVoiceMailInfo
- Scripting_Phone_LastCallerInfo
- Scripting_Phone_HIPSetCallWaitingLED
- Scripting_Phone_HIPSendLocalCID
- Scripting_Phone_HIPCmd
- Scripting_Phone_HandsetOnHook
- Scripting_Phone_GetLastVoiceCommand
- Scripting_Phone_CreateMessageFilename
- Scripting_Phone_ContactClass
Scripting_Phone_MBMessageTime

MBMessageTime

Purpose

This function returns a string representing the time the given message was left. The time is encoded in the file name of a voice message and this function extracts the time information.

Parameters

Parameter: Message
Type: String
Description: The file name of the voice message.

Returns

Return value: Time
Type: String

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextunreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnread
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByName
Scripting_Phone_MBGet
Scripting_Phone_MBFirstunreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBDeleteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAntwerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_MBMessageName

MBMessageName

Purpose

This function returns the name of the user who sent the voice message. This field is the actual name returned by Caller ID. This information is encoded in the file name of the voice message and this function simply extracts it. If the Caller ID information does not include the name, this field will return the name of the caller if there is a match in the address book for the Caller ID phone number.

Parameters

Parameter: Message
Type: String
Description: The file name of the voice message.

Returns

Return value: Caller
Type: String
Description: The person who left the message. The actual name of the caller as provided by your phone company. Note that you may need to subscribe to the Caller ID name service before any value will be visible here.

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_MBMessageLength

MBMessageLength

**Purpose**

This function returns the length of a voicemail message in seconds.

**Parameters**

Parameter: **Message**  
Type: **String**  
Description: The file name of the voice message.

**Returns**

Return value: **Length**  
Type: **Integer**  
Description: The length of the voice message, in seconds.

See also

- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MNew
- Scripting_Phone_MBMessageTime
- Scripting_Phone_MBMessageName
- Scripting_Phone_MBMessageFrom
- Scripting_Phone_MBMessageDate
- Scripting_Phone_MMarkUnRead
- Scripting_Phone_MMarkRead
- Scripting_Phone_MGetLoggedln
- Scripting_Phone_MGetDefault
- Scripting_Phone_MGetByName
- Scripting_Phone_MGet
- Scripting_Phone_MFirstUnreadMessage
- Scripting_Phone_MFirstReadMessage
- Scripting_Phone_MDeleteMessage
- Scripting_Phone_MCount
- Scripting_Phone_MBCancelPendingNotifications
- Scripting_Phone_MBAnswerMode
- Scripting_Phone_MMailboxClass
- Scripting_Phone_LINEStopSpeaking
- Scripting_Phone_LINEStatus
- Scripting_Phone_LINESetVoice
- Scripting_Phone_LINESetRingCurrent
- Scripting_Phone_LINESetSpeakingSpeed
- Scripting_Phone_LINESetRings
- Scripting_Phone.LinesetGreeting
- Scripting_Phone_LINESetCIDNumber
- Scripting_Phone_LINESetCIDName
- Scripting_Phone_LINESetCIDInfo
- Scripting_Phone_LINESetAnswerMode
- Scripting_Phone_LINESetSendTones
- Scripting_Phone_LINESetSendAT
- Scripting_Phone_LINESetHasControl
- Scripting_Phone_LINESetRingCount
- Scripting_Phone_LINESetCallTimeout
- Scripting_Phone_LINESetReset
- Scripting_Phone_LINESetRecordStop
- Scripting_Phone_LINESet_RecordStart
- Scripting_Phone_LINESetMuteRings
- Scripting_Phone_LINESetIsSpeaking
- Scripting_Phone_LINESetHangup
- Scripting_Phone_LINESetGetVoice
- Scripting_Phone_LINESetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_ContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount

Home > Scripting > Phone > Scripting_Phone_MBMessageFrom

Scripting_Phone_MBMessageFrom

MBMessageFrom

Purpose
This function returns the name of the user who sent the voice message. The user will be either a Caller ID name or phone number. This information is encoded in the file name of the voice message and this function simply extracts it.

Parameters
Parameter: Message
Type: String
Description: The file name of the voice message.

Returns
Return value: Caller
Type: String
Description: The person who left the message. Either a name or phone number.

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageDate
Scripting_Phone_MBBmarkUnRead
Scripting_Phone_MBBmarkRead
Scripting_Phone_MBBgetLoggedin
Scripting_Phone_MBBgetDefault
Scripting_Phone_MBBgetByFileName
Scripting_Phone_MBGet
Scripting_Phone_MBFIRSTUnreadMessage
Scripting_Phone_MBFIRSTReadMessage
Scripting_Phone_MBDeleteMessage
Scripting_Phone_MBCOUNT
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAAnswerMode
Scripting_Phone_MBoxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINESetRingCount
Scripting_Phone_LINESetCallTimeout
Scripting_Phone_LINESetReset
Scripting_Phone_LINESetRecordStop
Scripting_Phone_LINESetRecordStart
Scripting_Phone_LINESetMuteRings
Scripting_Phone_LINESetSpeaking
Scripting_Phone_LINESetHangup
Scripting_Phone_LINESetGetVoice
Scripting_Phone_LINESetGetDTMFString
Scripting_Phone_LINESetGetDTMFCount
Scripting_Phone_LINESetEnableSpeakerPhone
Scripting_Phone_LINESetDisableSpeakerPhone
Scripting_Phone_LINESetCount
Scripting_Phone_LINESetDialed
Scripting_Phone_LINESetAnswerSpeakerPhone
Scripting_Phone_LINESetAnswerLocal
Scripting_Phone_LINESetMailInfo
Scripting_Phone_LINESetCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_CreateContactClass
Scripting_Phone_CLEARLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount

Home > Scripting > Phone > Scripting_Phone_MBMessageDate

Scripting_Phone_MBMessageDate

MBMessageDate

Purpose

This function returns a string representing the date the given message was left. The date is encoded in the file name of a voice message and this function extracts the date information.

Parameters

Parameter: Message
Type: String
Description: The file name of the voice message.
Returns

None.

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetName
Scripting_Phone_MBGet
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBDeleteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnternMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESetTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINEIsSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_HIPGetLastVoiceCommand
Scripting_Phone_HIPCreateMessageFilename
Scripting_Phone_HIPContactClass
Scripting_Phone_HIPClearLastVoiceCommand
Scripting_Phone_HIPCIDNumber
Scripting_Phone_HIPCIDName
Scripting_Phone_HIPSave
MBMarkUnRead

Purpose

This function marks the voice message as not read.

Parameters

Parameter: Message
Type: String
Description: The file name of the voice message.

Returns

None.

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByIndex
Scripting_Phone_MBGet
Scripting_Phone_MBFIRSTUnreadMessage
Scripting_Phone_MBFIRSTReadMessage
Scripting_Phone_MBDelMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetRingsSpeed
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetCIDAnswerMode
Scripting_Phone.LinesendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINESetRingCount
Scripting_Phone_LINESetCallTimeout
Scripting_Phone_MBMarkRead

MBMarkRead

Purpose

This function marks the voice message as read.

Parameters

Parameter: Message
Type: String
Description: The file name of the message to mark.

Returns

None.

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBGetLoggedIn

**MBGetLoggedIn**

**Purpose**

This function returns the mailbox index of the mailbox the caller is currently logged into. If the caller has not logged into a mailbox, this function returns 0. A caller logs into a mailbox using their passcode.
Parameters

Parameter: Line
Type: Integer
Description: The phone line to access.

Returns

Return value: Index
Type: Integer
Description: The index of the mailbox the caller is logged into.

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_ResetLine
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetName
Scripting_Phone_MBGet
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBDeleteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINEReset
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEHangup
Scripting_Phone_LINELastVoiceMailInfo
Scripting_Phone_LINELastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_ContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount

Scripting_Phone_MBGetDefault

MBGetDefault

Purpose
This function returns a reference to the default mailbox. The default mailbox is a specially marked mailbox that is used when the system is set up to single mailbox mode. All voice mail is left in this mailbox.

Parameters
None.

Returns
Return value: Mailbox Class
Type: Object as MailboxClass
Description: A reference to the default mailbox of class mailbox.

Examples
The following example gets the default mailbox and then accesses each voice mail message.

```
sub main()
    dim mb
    dim messages
    dim mfile
    dim count
    dim i
    set mb=hsp.MBGetDefault ' get the default mailbox
    set messages = mb.messages ' get the collection of messages
    count = messages.count ' get the total number of messages
    msgbox cstr(count)
    for i=1 to count
        set mfile = messages(i) ' get a reference to a message of type message_file
        msgbox mfile.filename ' get the filename of the voice message
    next
end sub
```

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIN
Scripting_Phone_MBSetByName
Scripting_Phone_MBGet
Scripting_Phone_MBFIRSTUnreadMessage
Scripting_Phone_MBFIRSTReadMessage
Scripting_Phone_MBDelteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnswerMode
Scripting_Phone_MBoxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESetSendTones
Scripting_Phone_LINESetSendDIAL
Scripting_Phone_LINESetScriptHasControl
Scripting_Phone_LINESetRingCount
Scripting_Phone_LINESetCallTimeout
Scripting_Phone_LINESetStop
Scripting_Phone_LINESetRecordStop
Scripting_Phone_LINESetRecordStart
Scripting_Phone_LINESetMuteRings
Scripting_Phone_LINESetIsSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDMFString
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_GetCIDNumber
Scripting_Phone GetCIDName
Scripting_Phone ADRSave
Scripting_Phone ADRNew
Scripting_Phone ADRGet
Scripting_Phone ADRDelete
Scripting_Phone ADRCount

Home > Scripting > Phone > Scripting_Phone_MBGtByNAme

Scripting_Phone_MBGtByNAme
**MBGetByName**

**Purpose**
This function returns a reference to a mailbox class using the name of the mailbox.

**Parameters**
Parameter: **Username**
Type: **String**
Description: The user name string of the mailbox to retrieve. The name is not case-sensitive.

**Returns**
Return value: **Mailbox Class**
Type: **Object as MailboxClass**
Description: A reference to a mailbox class.

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_MBSort
Scripting_Phone_SAVE
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGet
Scripting_Phone_MBFIRSTUnreadMessage
Scripting_Phone_MBFIRSTReadMessage
Scripting_Phone_MBDeleteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAntiVirus
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEIsMuteRings
Scripting_Phone_LINEsetSpeaking
Scripting_Phone_LINEsetHangup
Scripting_Phone_LINESetVoice
Scripting_Phone_LINEGetDTMFSend
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESetCallTimeout
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone<ContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount

Home > Scripting > Phone > Scripting_Phone_MBGet

Scripting_Phone_MBGet

MBGet

Purpose

This function returns a reference to a mailbox class.

Parameters

Parameter: **Index**

Type: **Integer**

Description: The index number of the mailbox to retrieve.

Returns

Return value: **Mailbox Class**

Type: **object as MailboxClass**

Description: A reference to a mailbox class.

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBBestNextUnreadMessage
Scripting_Phone_MBBestNextReadMessage
Scripting_Phone_MBNextMessage
Scripting_Phone_MBNormalMessageTime
Scripting_Phone_MBNormalMessageName
Scripting_Phone_MBNormalMessageLength
Scripting_Phone_MBNormalMessageFrom
Scripting_Phone_MBNormalMessageDate
Scripting_Phone_MBNormalMessage
Scripting_Phone_MBNormalMessageRead
Scripting_Phone_MBNormalMessageUnRead
Scripting_Phone_MBNormalMessageLogged
Scripting_Phone_MBNormalMessageDefault
Scripting_Phone_MBBestByName
Scripting_Phone_MBBestFirstUnreadMessage
Scripting_Phone_MBBestFirstReadMessage
Scripting_Phone_MBBestDeleteMessage
Scripting_Phone_MBBestCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINESetHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINEIsSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_ContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount

**Scripting_Phone_MBFirstUnreadMessage**

**Purpose**

This function returns the first unread message in the given mailbox. To get all the unread messages, this function should be called first, then call MBNextUnreadMessage.

**Parameters**

- **Parameter:** `Line`  
  **Type:** `Integer`  
  Description: The phone line to access.

- **Parameter:** `mb`  
  **Type:** `Mailbox`  
  Description: The mailbox to access. This is a reference to a mailbox object. Use MBGet to get a mailbox.
**Returns**

Return value: **File name**  
Type: **String**  
Description: The file name of the voice message. Note that this file name does not include the full path to the file. Voice messages are saved in the directory messages in the HomeSeer application directory. To create the full path to the file, use the GetAppPath function like:

\[ path = hsp.GetAppPath & "messages" & message \]

See also
- Scripting_Phone_LINEClearDTMF  
- Scripting_Phone_WaitMS  
- Scripting_Phone_StopListening  
- Scripting_Phone_StartListening  
- Scripting_Phone_Speak  
- Scripting_Phone_SetSpeaker  
- Scripting_Phone_RestoreSettings  
- Scripting_Phone_MBSort  
- Scripting_Phone_MBSave  
- Scripting_Phone_MBNextUnreadMessage  
- Scripting_Phone_MBNextReadMessage  
- Scripting_Phone_MBNew  
- Scripting_Phone_MBMessageTime  
- Scripting_Phone_MBMessageName  
- Scripting_Phone_MBMessageLength  
- Scripting_Phone_MBMessageFrom  
- Scripting_Phone_MBMessageDate  
- Scripting_Phone_MBMarkUnRead  
- Scripting_Phone_MBMarkRead  
- Scripting_Phone_MBGetLoggedln  
- Scripting_Phone_MBGetDefault  
- Scripting_Phone_MBGetByName  
- Scripting_Phone_MBGet  
- Scripting_Phone_MBFirstReadMessage  
- Scripting_Phone_MBdeleteMessage  
- Scripting_Phone_MBCount  
- Scripting_Phone_MBCancelPendingNotifications  
- Scripting_Phone_MBAAnswerMode  
- Scripting_Phone_MailboxClass  
- Scripting_Phone_LINEStopSpeaking  
- Scripting_Phone_LINEStatus  
- Scripting_Phone_LINESetVoice  
- Scripting_Phone_LINESetRingsCurrent  
- Scripting_Phone_LINESetSpeakingSpeed  
- Scripting_Phone_LINESetRings  
- Scripting_Phone_LINESetGreeting  
- Scripting_Phone_LINESetCIDNumber  
- Scripting_Phone_LINESetCIDName  
- Scripting_Phone_LINESetCIDInfo  
- Scripting_Phone_LINESetAnswerMode  
- Scripting_Phone_LINESendTones  
- Scripting_Phone_LINESendAT  
- Scripting_Phone_LINEScriptHasControl  
- Scripting_Phone_LINERingCount  
- Scripting_Phone_LINEResetCallTimeout  
- Scripting_Phone_LINEReset  
- Scripting_Phone_LINERecordStop  
- Scripting_Phone_LINERecordStart  
- Scripting_Phone_LINEMuteRings  
- Scripting_Phone_LINEIsSpeaking  
- Scripting_Phone_LINEHangup  
- Scripting_Phone_LINEGetVoice  
- Scripting_Phone_LINEGetDTMFSString  
- Scripting_Phone_LINEGetDTMFCount  
- Scripting_Phone_LINEEnableSpeakerPhone  
- Scripting_Phone_LINEDisableSpeakerPhone  
- Scripting_Phone_LINECount  
- Scripting_Phone_LINEDial  
- Scripting_Phone_LINEAnswerSpeakerPhone  
- Scripting_Phone_LINEAnswerLocal  
- Scripting_Phone_LINEAnswer  
- Scripting_Phone_LastVoiceMailInfo  
- Scripting_Phone_LastCallerInfo  
- Scripting_Phone_HIPSetCallWaitingLED  
- Scripting_Phone_HIPSendLocalCID  
- Scripting_Phone_HIPCmd  
- Scripting_Phone_HandsetOnHook  
- Scripting_Phone_GetLastVoiceCommand  
- Scripting_Phone_CreateMessageFilename  
- Scripting_Phone_ContactClass
Scripting_Phone_MBFirstReadMessage

MBFirstReadMessage

Purpose

This function returns the first read message in the given mailbox. To get all the read messages, this function should be called first, then call MBNextReadMessage.

Parameters

Parameter: Line
Type: Integer
Description: The phone line to access.

Parameter: mb
Type: Mailbox
Description: The mailbox to access. This is a reference to a mailbox object. Use hsp.MBGet to get a mailbox.

Returns

Return value: File name
Type: String
Description: The filename of the voice message. Note that this filename does not include the full path to the file. Voice messages are saved in the directory messages in the HomeSeer application directory. To create the full path to the file, use the GetAppPath function like:

```
path = hsp.GetAppPath & "\messages\" & message
```

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessagingTime
Scripting_Phone_MBMessagingName
Scripting_Phone_MBMessagingLength
Scripting_Phone_MBMessagingFrom
Scripting_Phone_MBMessagingDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByName
Scripting_Phone_MBGet
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBBDeleteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBSendMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
MBDeleteMessage

Purpose
This function deletes the given voice message. The message file is deleted.

Parameters
Parameter: Message
Type: String
Description: The file name of the voice message. The file name must not include the path.

Returns
None.

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_MBCount

MBCount

Purpose

This function returns the total number of mailboxes configured. This function can be used to iterate through all the configured mailboxes.

Parameters

None.

Returns

Return value: **Number**
Type: **Integer**
Description: The total number of mailboxes configured in the application.

See also

- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MBNew
- Scripting_Phone_MBMessagingTime
- Scripting_Phone_MBMessagingName
- Scripting_Phone_MBMessagingLength
- Scripting_Phone_MBMessagingFrom
- Scripting_Phone_MBMessagingDate
- Scripting_Phone_MBMakUnRead
- Scripting_Phone_MBMakRead
- Scripting_Phone_MBGetLoggedIn
- Scripting_Phone_MBGetDefault
- Scripting_Phone_MBGetByName
- Scripting_Phone_MBGet
- Scripting_Phone_MBFIRSTUnreadMessage
- Scripting_Phone_MBFIRSTReadMessage
- Scripting_Phone_MBDeleteMessage
- Scripting_Phone_MBCancelPendingNotifications
- Scripting_Phone_MBAnswerMode
- Scripting_Phone_MailboxClass
- Scripting_Phone_LINEStopSpeaking
- Scripting_Phone_LINESetVoice
- Scripting_Phone_LINESetRingsCurrent
- Scripting_Phone_LINESetSpeakingSpeed
- Scripting_Phone_LINESetRings
- Scripting_Phone_LINESetGreeting
- Scripting_Phone_LINESetCIDNumber
- Scripting_Phone_LINESetCIDName
- Scripting_Phone_LINESetCIDInfo
- Scripting_Phone_LINESetAnswerMode
- Scripting_Phone_LINESendTones
- Scripting_Phone_LINESendDAI
- Scripting_Phone_LINEScriptHasControl
- Scripting_Phone_LINERingCount
- Scripting_Phone_LINEResetCallTimeout
- Scripting_Phone_LINEReset
- Scripting_Phone_LINERecordStop
- Scripting_Phone_LINERecordStart
- Scripting_Phone_LINEMuteRings
- Scripting_Phone_LINEIsSpeaking
- Scripting_Phone_LINEHangup
- Scripting_Phone_LINEGetVoice
- Scripting_Phone_LINEGetDTMFString
- Scripting_Phone_LINEGetDTMFCount
- Scripting_Phone_LINEMuteSpeakerPhone
- Scripting_Phone_LINESetSpeakerPhone
- Scripting_Phone_LINESetCount
Scripting_Phone_MBCancelPendingNotifications

**MBCancelPendingNotifications**

**Purpose**

This function will cancel all pending notifications such as cell phone notifications (dialing out to notify someone that a message is in their mailbox), E-mail notifications, and pager notifications.

**Parameters**

None.

**Returns**

None.

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitSMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByName
Scripting_Phone_MBGet
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MDBeepMessage
Scripting_Phone_MBCount
MBAnswerMode

Purpose
This function either sets or gets the current answer mode of the system.
The answer mode is one of:

1 = multiple mailbox mode (the caller must enter a mailbox where they wish to leave a message)
2 = single mailbox mode (the caller simply leaves a message in the default mailbox)

Parameters
Parameter: Mode (for set)
Type: Integer
Description: The mode to set, either 1 or 2.
Returns

Return value: **Mode**
Type: **Integer**
Description: The current operating mode, either 1 or 2.

Examples

```plaintext
hsp.MBAnswerMode = modemode
return = hsp.MBAnswerMode

To set the operating mode to a single mailbox:

```plaintext
sub main()
    hsp.MBAnswerMode = 2
end sub
```

See also

- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBsort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MNew
- Scripting_Phone_MBMessageTime
- Scripting_Phone_MBMessageName
- Scripting_Phone_MBMessageLength
- Scripting_Phone_MBMessageFrom
- Scripting_Phone_MBMessageDate
- Scripting_Phone_MBMarkUnRead
- Scripting_Phone_MBMarkRead
- Scripting_Phone_MGetLogged
- Scripting_Phone_MGetDefault
- Scripting_Phone_MGetByName
- Scripting_Phone_MGet
- Scripting_Phone_MFirstUnreadMessage
- Scripting_Phone_MFirstReadMessage
- Scripting_Phone_MDeleteMessage
- Scripting_Phone_MCCount
- Scripting_Phone_MCCancelPendingNotifications
- Scripting_Phone_MailboxClass
- Scripting_Phone_LINEStopSpeaking
- Scripting_Phone_LINEStatus
- Scripting_Phone_LINESetVoice
- Scripting_Phone_LINESetRingsCurrent
- Scripting_Phone_LINESetSpeakingSpeed
- Scripting_Phone_LINESetRings
- Scripting_Phone_LINESetGreeting
- Scripting_Phone_LINESetCIDNumber
- Scripting_Phone_LINESetCIDName
- Scripting_Phone_LINESetCIDInfo
- Scripting_Phone_LINESetAnswerMode
- Scripting_Phone_LINESendTones
- Scripting_Phone_LINESendAT
- Scripting_Phone_LINEScriptHasControl
- Scripting_Phone_LINESetRingCount
- Scripting_Phone_LINESetCallTimeout
- Scripting_Phone_LINESet
- Scripting_Phone_LINESetRecordStop
- Scripting_Phone_LINESetRecordStart
- Scripting_Phone_LINESetMuteRings
- Scripting_Phone_LINESetSpeaking
- Scripting_Phone_LINESetHangup
- Scripting_Phone_LINESetGetVoice
- Scripting_Phone_LINESetGetDTMFString
- Scripting_Phone_LINESetGetDTMFCount
- Scripting_Phone_LINESetEnableSpeakerPhone
- Scripting_Phone_LINESetDisableSpeakerPhone
- Scripting_Phone_LINESetCount
- Scripting_Phone_LINESetDial
```
The Mailbox Class

MailboxClass Object

Various properties of a mailbox may be set and retrieved. The properties of a mailbox are defined as follows:

<table>
<thead>
<tr>
<th>Keypad</th>
<th>Command Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>cellphone_number</td>
<td>The cell phone number to call with new voice messages.</td>
</tr>
<tr>
<td>email_forward</td>
<td>The greeting to be played to the caller entering this mailbox.</td>
</tr>
<tr>
<td>greeting</td>
<td>(from any level) Exit back to main menu.</td>
</tr>
<tr>
<td>notify_hi_water</td>
<td>The number of voice messages that must be left before a page or callback is executed. E-mail notifications are not subject to this value.</td>
</tr>
<tr>
<td>number</td>
<td>The mailbox number.</td>
</tr>
<tr>
<td>pager_number</td>
<td>The phone number of the user's pager.</td>
</tr>
<tr>
<td>passcode</td>
<td>A string of DTMF digits that is the passcode for this user.</td>
</tr>
<tr>
<td>tag</td>
<td>Holds user-defined information.</td>
</tr>
<tr>
<td>total_messages</td>
<td>A count of the total number of voice messages in this mailbox.</td>
</tr>
<tr>
<td>unread_messages</td>
<td>A count of the total number of voice messages unread in this mailbox.</td>
</tr>
<tr>
<td>username</td>
<td>The owner of the mailbox.</td>
</tr>
</tbody>
</table>
attributes

Bits defined:

- `MB_ALLOW_MESSAGES = 2`  
  * callers can leave messages in this mailbox
- `MB_ALLOW_HS_VOICE_COMMANDS = 4`  
  * callers can access voice commands (if # enabled)
- `MB_DEFAULT = 8`  
  * default, cannot delete
- `MB_FWD_EMAIL = &H10`  
  * forward messages to given E-mail address
- `MB_NOTIFY_PAGE = &H20`  
  * notify to pager number
- `MB_FWD_CELLPHONE = &H40`  
  * forward messages to cell phone
- `MB_NOTIFY_NO_VOICE = &H80`  
  * do not include the voice file in notifications
- `MB_ATTACH_CID = &H100`  
  * include Caller ID number in notification

See also

- `Scripting_Phone_LINEClearDTMF`
- `Scripting_Phone_WaitMS`
- `Scripting_Phone_StopListening`
- `Scripting_Phone_StartListening`
- `Scripting_Phone_Speak`
- `Scripting_Phone_MBSort`
- `Scripting_Phone_MBSave`
- `Scripting_Phone_MBNextUnreadMessage`
- `Scripting_Phone_MBNextReadMessage`
- `Scripting_Phone_MBBNew`
- `Scripting_Phone_MBBMessageTime`
- `Scripting_Phone_MBBMessageName`
- `Scripting_Phone_MBBMessageLength`
- `Scripting_Phone_MBBMessageFrom`
- `Scripting_Phone_MBBMessageDate`
- `Scripting_Phone_MBBMarkUnRead`
- `Scripting_Phone_MBBMarkRead`
- `Scripting_Phone_MBBGetLoggedIn`
- `Scripting_Phone_MBBGetDefault`
- `Scripting_Phone_MBBGetByName`
- `Scripting_Phone_MBBGet`
- `Scripting_Phone_MBBFirstUnreadMessage`
- `Scripting_Phone_MBBFirstReadMessage`
- `Scripting_Phone_MBBDeleteMessage`
- `Scripting_Phone_MBBCount`
- `Scripting_Phone_MBCancelPendingNotifications`
- `Scripting_Phone_MBAnswerMode`
- `Scripting_Phone_LINEStopSpeaking`
- `Scripting_Phone_LINEStatus`
- `Scripting_Phone_LINESetVoice`
- `Scripting_Phone_LINESetRingsCurrent`
- `Scripting_Phone_LINESetSpeakingSpeed`
- `Scripting_Phone_LINESetRings`
- `Scripting_Phone_LINESetGreeting`
- `Scripting_Phone_LINESetCIDNumber`
- `Scripting_Phone_LINESetCIDName`
- `Scripting_Phone_LINESetCIDInfo`
- `Scripting_Phone_LINESetAnswerMode`
- `Scripting_Phone_LINESendTones`
- `Scripting_Phone_LINESendAT`
- `Scripting_Phone_LINEScriptHasControl`
- `Scripting_Phone_LINERingCount`
- `Scripting_Phone_LINEResetCallTimeout`
- `Scripting_Phone_LINEReset`
- `Scripting_Phone_LINERecordStop`
- `Scripting_Phone_LINERecordStart`
- `Scripting_Phone_LINEMuteRings`
- `Scripting_Phone_LINEIsSpeaking`
- `Scripting_Phone_LINEHangup`
- `Scripting_Phone_LINEGetVoice`
- `Scripting_Phone_LINEGetDTMFString`
- `Scripting_Phone_LINEGetDTMFCount`
- `Scripting_Phone_LINEEnableSpeakerPhone`
- `Scripting_Phone_LINEDisableSpeakerPhone`
- `Scripting_Phone_LINECount`
- `Scripting_Phone_LINEDial`
- `Scripting_Phone_LINEAnswerSpeakerPhone`
- `Scripting_Phone_LINEAnswerLocal`
- `Scripting_Phone_LINEAnswer`
- `Scripting_Phone_LastVoiceMailInfo`
- `Scripting_Phone_LastCallerInfo`
Scripting_Phone_LINEStopSpeaking

**LINEStopSpeaking**

**Purpose**

This function stops the speaking of text-to-speech or the playing of a WAV file on the given line.

**Parameters**

Parameter: **Line**  
Type: **Integer**  
Description: The phone line to stop speaking/playing on.

**Returns**

None.

See also

Scripting_Phone_LINEClearDTMF  
Scripting_Phone_WaitMS  
Scripting_Phone_StopListening  
Scripting_Phone_StartListening  
Scripting_Phone_Speak  
Scripting_Phone_SetSpeaker  
Scripting_Phone_RestoreSettings  
Scripting_Phone_MBSort  
Scripting_Phone_MBSave  
Scripting_Phone_MBBinMessage  
Scripting_Phone_MBBinNewMessage  
Scripting_Phone_MBBinMessageTime  
Scripting_Phone_MBBinMessageName  
Scripting_Phone_MBBinMessageLength  
Scripting_Phone_MBBinMessageFrom  
Scripting_Phone_MBBinMessageDate  
Scripting_Phone_MBBinMarkUnRead  
Scripting_Phone_MBBinMarkRead  
Scripting_Phone_MBBinGetLoggedIn  
Scripting_Phone_MBBinGetDefault  
Scripting_Phone_MBBinGetName  
Scripting_Phone_MBBinGet  
Scripting_Phone_MBBinFirstUnreadMessage  
Scripting_Phone_MBBinFirstReadMessage  
Scripting_Phone_MBBinDeleteMessage  
Scripting_Phone_MBBinCount  
Scripting_Phone_MBBinCancelPendingNotifications  
Scripting_Phone_MBBinAnswerMode  
Scripting_Phone_MBBinMailboxClass  
Scripting_Phone_LINEStatus  
Scripting_Phone_LINESetVoice  
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_PhoneILINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINEIsSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_ContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount

Scripting_Phone_LINEStatus

LINEStatus

Purpose

This function returns the status of a call. This call can be used in a script to determine if the call is ended. If the status is LINE_IDLE, there is no call, and the script should exit immediately.

Parameters

Parameter: Line
Type: Integer
Description: The phone line to access.

Returns

Return value: Code
Type: String
Description: One of the following codes:

- **LINE_IDLE** = 0 ' waiting for call
- **LINE_OFFERING** = 1 ' incoming call before first ring
- **LINE_RINGING** = 2 ' incoming call
- **LINE_CONNECTED** = 3 ' line is active and connected to remote party
- **LINE_INACTIVE** = 4 ' not waiting for call, maybe no modem selected on line
LINE_BUSY = 5 ' line busy
LINE_INUSE = 6 ' line is in use
LINE_TIMEOUT = 7 ' for calling, no answer
LINE_ERROR = 8 ' line error event
LINE_DIALING = 9 ' dialing out in progress
LINE_REORDER = 10 ' fast busy, Hi-Phone only

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessagetime
Scripting_Phone_MBMessagename
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessagefrom
Scripting_Phone_MBMessagedate
Scripting_Phone_MBMarksUnRead
Scripting_Phone_MBMarksRead
Scripting_Phone_MBGetsLogged
Scripting_Phone_MBGetsDefault
Scripting_Phone_MBGetsByName
Scripting_Phone_MFBFirstUnreadMessage
Scripting_Phone_MFBFirstReadMessage
Scripting_Phone_MFBDeleteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnterMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINESetHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINEIsSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_ContactClass
Scripting_Phone_LINESetVoice

**LI N ESetVoice**

**Purpose**

This function sets a new text-to-speech voice for the given line.

**Parameters**

Parameter: **Line**  
Type: **Integer**  
Description: The phone line to retrieve the voice from.

Parameter: **Voice**  
Type: **String**  
Description: The name of the new voice to set. Only SAPI5-compatible voices are supported.

**Returns**

None.

See also

- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MBNew
- Scripting_Phone_MBMessageTime
- Scripting_Phone_MBMessageName
- Scripting_Phone_MBMessageLength
- Scripting_Phone_MBMessageFrom
- Scripting_Phone_MBMessageDate
- Scripting_Phone_MBMarkUnRead
- Scripting_Phone_MBMarkRead
- Scripting_Phone_MBGetLoggedIn
- Scripting_Phone_MBGetDefault
- Scripting_Phone_MBGetByName
- Scripting_Phone_MBBGet
- Scripting_Phone_MBFirstUnreadMessage
- Scripting_Phone_MBFirstReadMessage
- Scripting_Phone_MBDeletemessage
- Scripting_Phone_MBCount
- Scripting_Phone_MBCancelPendingNotifications
- Scripting_Phone_MBAwswerMode
- Scripting_Phone_MailboxClass
- Scripting_Phone_LINEStopSpeaking
- Scripting_Phone_LINEStatus
- Scripting_Phone_LINESetRingsCurrent
- Scripting_Phone_LINESetSpeakingSpeed
- Scripting_Phone_LINESetRings
- Scripting_Phone_LINESetGreeting
- Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetRingsCurrent

**LINESetRingsCurrent**

**Purpose**

This function sets the number of rings to answer the current ringing call. Note that this call can only be made while the line is currently ringing. It may be used after Caller ID information has been examined and it has been determined that the call should be answered in a different number of rings than the default. Calling this function when the line is not ringing has no affect.

**Parameters**

- **Parameter:** Line  
  **Type:** Integer  
  **Description:** The phone line to access.

- **Parameter:** Rings  
  **Type:** Integer  
  **Description:** The number of rings this call will answer in.

**Returns**

None.

**See also**

- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
LINESetSpeakingSpeed

Purpose
This function sets the speaking speed for text-to-speech on the given line.

Parameters
Parameter: Line
Type: Integer
Description: The phone line to set the speed to.

Parameter: Speed
Type: Integer
Description: The rate to set the speech to.

Returns
None.

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBnextUnreadMessage
Scripting_Phone_MBnextReadMessage
Scripting_Phone_MNew
Scripting_Phone_MMessageTime
Scripting_Phone_MMessageName
Scripting_Phone_MMessageLength
Scripting_Phone_MMessageFrom
Scripting_Phone_MMessageDate
Scripting_Phone_MMarkUnRead
Scripting_Phone_MMarkRead
Scripting_Phone_MGetLoggedIn
Scripting_Phone_MGetDefault
Scripting_Phone_MGetByName
Scripting_Phone_MGet
Scripting_Phone_MFirstUnreadMessage
Scripting_Phone_MFirstReadMessage
Scripting_Phone_MDeleteMessage
Scripting_Phone_MCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MAnswerMode
Scripting_Phone_MMailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEstatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINESpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
LINESetRings

Purpose

This function sets the number of rings to answer the call. This is the same as setting the number of rings in the modem tab in the options. This is useful for setting a “Do not disturb” mode where you want to dump all callers to the voice system. Set the number of rings to 2 so that you can gather Caller ID information before answering.

Parameters

Parameter: Line
Type: Integer
Description: The phone line to access.

Parameter: Rings
Type: Integer
Description: The number of rings to set.

Returns

None.

Examples

Set the number of rings to answer to 4 on line 1:

    hsp.LINESetRings 1, 4
Scripting_Phone_LINESetGreeting

LINESetGreeting

Purpose
This function sets the default greeting for the given phone line. There are two default greetings, one for a specific time range and the other for all other times. This function sets both greetings to the same phrase. This is useful if you want to set a different greeting throughout the day. In HomeSeer, you can create an event that will set the greeting for you. See the example below.

**Parameters**

Parameter: **Greeting**  
Type: **String**  
Description: The phrase to set the greeting to.

**Returns**

None.

**Examples**

To have HomeSeer set a greeting at a specific time, create an event that is triggered by the desired time, such as 8:00 AM. Then enter the following script command in the script run box on the scripting tab for the event:

```csharp
$hsp.LINESetGreeting "Good morning, please leave a message at the beep"
```

See also

- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MBNew
- Scripting_Phone_MBMessageTime
- Scripting_Phone_MBMessageName
- Scripting_Phone_MBMessageLength
- Scripting_Phone_MBMessageFrom
- Scripting_Phone_MBMessageDate
- Scripting_Phone_MBMarkUnRead
- Scripting_Phone_MBMarkRead
- Scripting_Phone_MBGetLoggedIn
- Scripting_Phone_MBGetDefault
- Scripting_Phone_MBGetByName
- Scripting_Phone_MBGet
- Scripting_Phone_MBFirstUnreadMessage
- Scripting_Phone_MBFirstReadMessage
- Scripting_Phone_MBDeleteMessage
- Scripting_Phone_MBCount
- Scripting_Phone_MBCancelPendingNotifications
- Scripting_Phone_MBAnswerMode
- Scripting_Phone_MailboxClass
- Scripting_Phone_LINEStopSpeaking
- Scripting_Phone_LINEStatus
- Scripting_Phone_LINESetVoice
- Scripting_Phone_LINESetRingsCurrent
- Scripting_Phone_LINESetSpeakingSpeed
- Scripting_Phone_LINESetRings
- Scripting_Phone_LINESetCIDNumber
- Scripting_Phone_LINESetCIDName
- Scripting_Phone_LINESetCIDInfo
- Scripting_Phone_LINESetAnswerMode
- Scripting_Phone_LINESendTones
- Scripting_Phone_LINESendAT
- Scripting_Phone_LINEScriptHasControl
- Scripting_Phone_LINEResetCallTimeout
- Scripting_Phone_LINEReset
- Scripting_Phone_LINERecordStop
- Scripting_Phone_LINERecordStart
- Scripting_Phone_LINEIsSpeaking
- Scripting_Phone_LINEHangup
- Scripting_Phone_LINEMuteRings
- Scripting_Phone_LINEGetDTMFString
- Scripting_Phone_LINEGetDTMFCount
- Scripting_Phone_LINEEnableSpeakerPhone
- Scripting_Phone_LINEDisableSpeakerPhone
- Scripting_Phone_LINECount
Scripting_Phone_LINESetCIDNumber

LI NESetCIDNumber

Purpose
This function sets the Caller ID number parameter to the given number. Useful if Caller ID information is gathered from some other device. This function would need to be called before the second ring, as the application handles the Caller ID information after the second ring is detected.

Parameters

Parameter: Line
Type: Integer
Description: The phone line to answer.

Parameter: Number
Type: String
Description: The phone number of the caller.

Returns
None.

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessagetime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetPosition
Scripting_Phone_MBGetName
Scripting_Phone_LINESetCIDName

### Purpose

This function sets the Caller ID name parameter to the given name. Useful if Caller ID information is gathered from some other device. This function would need to be called before the second ring, as the application handles the Caller ID information after the second ring is detected.

### Parameters

- **Parameter:** `Line`  
  **Type:** `Integer`  
  **Description:** The phone line to answer.

- **Parameter:** `Name`
Type: String
Description: The name of the caller.

Returns
None.

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MNw
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByIndex
Scripting_Phone_MBGetByIndex
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBDeleteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAntMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINESpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_ContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_LINESetCIDInfo

LINESetCIDInfo

Purpose

This function sets the Caller ID name and number parameter to the given information. Useful if the Caller ID information for name and number needs to be presented at the same time to HomeSeer Phone. Unlike LINESetCIDName and LINESetCIDNumber, which have to be called after the first ring, this command signals HomeSeer Phone that Caller ID information has been set and can cause Caller ID-based events to trigger anytime the indicated line is in the ringing state.

Parameters

Parameter: Line
Type: Integer
Description: The phone line to answer.

Parameter: Name
Type: String
Description: The name of the caller.

Parameter: Number
Type: String
Description: The phone number of the caller.

Returns

None.

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNear
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedln
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByeName
Scripting_Phone_MBGet
Scripting_Phone_MBBFirstUnreadMessage
Scripting_Phone_MBBFirstReadMessage
Scripting_Phone_MBBDeleteMessage
Scripting_Phone_MBBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetAnswerMode

LINESetAnswerMode

Purpose

This function sets the current answer mode to one of four modes. A HomeSeer event can be used to control when the answering system is turned on and off. See the parameters for the values for mode.

Parameters

Parameter: Line
Type: Integer
Description: The phone line to access

Parameter: Mode
Type: Integer
Description: The mode to set the answering system to. Must be one of the following:

1 = answer after the number of rings set (use LINESetRings to adjust the ring count)
2 = look for Caller ID information only and don't answer calls
3 = answer external call as internal call on first ring
4 = system is disabled

Returns
Examples

Set the answering system to answer on the set number of rings on line 1.

```
sub main()
    hsp.LINESetAnswerMode 1,1
end sub
```

See also
- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MBNew
- Scripting_Phone_MBMessageTime
- Scripting_Phone_MBMessageName
- Scripting_Phone_MBMessageLength
- Scripting_Phone_MBMessageFrom
- Scripting_Phone_MBMessageDate
- Scripting_Phone_MBMarkUnRead
- Scripting_Phone_MBMarkRead
- Scripting_Phone_MBGetLoggedIn
- Scripting_Phone_MBGetDefault
- Scripting_Phone_MBGetByNumber
- Scripting_Phone_MBGetNavBar
- Scripting_Phone_MBFirstUnreadMessage
- Scripting_Phone_MBFirstReadMessage
- Scripting_Phone_MBDelMessage
- Scripting_Phone_MBCount
- Scripting_Phone_MBCancelPendingNotifications
- Scripting_Phone_MBAnswerMode
- Scripting_Phone_MAILboxClass
- Scripting_Phone_LINEStopSpeaking
- Scripting_Phone_LINESetVoice
- Scripting_Phone_LINESetRingsCurrent
- Scripting_Phone_LINESetSpeakingSpeed
- Scripting_Phone_LINESetRings
- Scripting_Phone_LINESetGreeting
- Scripting_Phone.LinesetCIDNumber
- Scripting_Phone_LINESetCIDName
- Scripting_Phone_LINESetCINfo
- Scripting_Phone_LINESendTones
- Scripting_Phone_LINESendAT
- Scripting_Phone_LINEScriptHasControl
- Scripting_Phone_LINEReset
- Scripting_Phone_LINERecordStop
- Scripting_Phone_LINERecordStart
- Scripting_Phone_LINECloseRings
- Scripting_Phone_LINEIsSpeaking
- Scripting_Phone_LINEHangup
- Scripting_Phone_LINEGetVoice
- Scripting_Phone_LINEGetDTMFString
- Scripting_Phone_LINEGetDTMFCount
- Scripting_Phone_LINEEnableSpeakerPhone
- Scripting_Phone_LINEDisableSpeakerPhone
- Scripting_Phone_LINECount
- Scripting_Phone_LINEReset
- Scripting_Phone_LINEAnswerSpeakerPhone
- Scripting_Phone_LINEAnswerLocal
- Scripting_Phone_LINEAnswer
- Scripting_Phone_LastVoiceMailInfo
- Scripting_Phone_LastCallerInfo
- Scripting_Phone_HIPSectorWaitingLED
- Scripting_Phone_HIPSectorLocalCID
- Scripting_Phone_HIPSectorCmd
- Scripting_Phone_HandsetOnHook
- Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_LINESendTones

**LINESendTones**

**Purpose**

This function sends DTMF tones over the phone line. A call must be active.

**Parameters**

- **Parameter:** Line
  - **Type:** Integer
  - **Description:** The phone line to access.

- **Parameter:** Digits
  - **Type:** String
  - **Description:** A string of digits. Valid values are "1234567890#*"

- **Parameter:** Duration
  - **Type:** Integer
  - **Description:** The time in milliseconds for each tone.
  
  This parameter is ignored for the Hi-Phone device.

**Returns**

None.

**See also**

- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MBNew
- Scripting_Phone_MBMessageTime
- Scripting_Phone_MBMessageName
- Scripting_Phone_MBMessageLength
- Scripting_Phone_MBMessageFrom
- Scripting_Phone_MBMessageDate
- Scripting_Phone_MBMarkUnRead
- Scripting_Phone_MBMarkRead
- Scripting_Phone_MBGetLoggedIn
- Scripting_Phone_MBGetDefault
- Scripting_Phone_MBGetByName
- Scripting_Phone_MBGet
- Scripting_Phone_MBFirstUnreadMessage
- Scripting_Phone_MBFirstReadMessage
- Scripting_Phone_MBDeleteMessage
- Scripting_Phone_MBCount
- Scripting_Phone_MBCancelPendingNotifications
- Scripting_Phone_MBAntlerMode
- Scripting_Phone_MailboxClass

---

Home > Scripting > Phone > Scripting_Phone_LINESendTones

---

HomeSeer HS3 - End User Documentation
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINESisSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_ContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount

Scripting_Phone_LINESendAT

LI NESendAT

Purpose
This function sends a raw text string directly to the attached modem. Useful for enabling special features of the modem.

- This command is not supported if you are using the Hi-Phone device.

Parameters
Parameter: Line
Type: Integer
Description: The phone line to access.

Parameter: Command
Type: String
Description: The command string to send to the modem. The string should be terminated with a carriage-return/linefeed pair (see the example below).

Returns
Return value: Response
Type: **String**
Description: The response from the modem. Normally this is "OK" or "ERROR".

**Examples**

The example below includes a carriage return and linefeed.

```vba
dim r
r = hsp.LINESendAT(1,"AT" & VBCRLF)
```

See also
- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MBNew
- Scripting_Phone_MBMessageTime
- Scripting_Phone_MBMessageName
- Scripting_Phone_MBMessageLength
- Scripting_Phone_MBMessageFrom
- Scripting_Phone_MBMessageDate
- Scripting_Phone_MBMarkUnRead
- Scripting_Phone_MBMarkRead
- Scripting_Phone_MBGetDefault
- Scripting_Phone_MBGetByName
- Scripting_Phone_MBGet
- Scripting_Phone_MBFirstUnreadMessage
- Scripting_Phone_MBFirstReadMessage
- Scripting_Phone_MBDeleteMessage
- Scripting_Phone_MBCount
- Scripting_Phone_MBCancelPendingNotifications
- Scripting_Phone_MBAnterimMode
- Scripting_Phone_MailboxClass
- Scripting_Phone_LINEStopSpeaking
- Scripting_Phone_LINEStatus
- Scripting_Phone_LINESetVoice
- Scripting_Phone_LINESetRingsCurrent
- Scripting_Phone_LINESetSpeakingSpeed
- Scripting_Phone_LINESetRings
- Scripting_Phone_LINESetGreeting
- Scripting_Phone_LINESetCIDNumber
- Scripting_Phone_LINESetCIDName
- Scripting_Phone_LINESetCIDInfo
- Scripting_Phone_LINESetAnswerMode
- Scripting_Phone_LINESendTones
- Scripting_Phone_LINEScriptHasControl
- Scripting_Phone_LINERingCount
- Scripting_Phone_LINEResetCallTimeout
- Scripting_Phone_LINEReset
- Scripting_Phone_LINESetRecordStop
- Scripting_Phone_LINESetRecordStart
- Scripting_Phone_LINEEnableRings
- Scripting_Phone_LINEIsSpeaking
- Scripting_Phone_LINEHangup
- Scripting_Phone_LINEGetVoice
- Scripting_Phone_LINEGetDTMFString
- Scripting_Phone_LINEGetDTMFCount
- Scripting_Phone_LINEEnableSpeakerPhone
- Scripting_Phone_LINEDisableSpeakerPhone
- Scripting_Phone_LINECount
- Scripting_Phone_LINEDial
- Scripting_Phone_LINEAnswerSpeakerPhone
- Scripting_Phone_LINEAnswerLocal
- Scripting_Phone_LINEAnswer
- Scripting_Phone_LastVoiceMailInfo
- Scripting_Phone_LastCallerInfo
- Scripting_Phone_HIPSetCallWaitingLED
- Scripting_Phone_HIPSendLocalCID
- Scripting_Phone_HIPCmd
- Scripting_Phone_HandsetOnHook
- Scripting_Phone_LastVoiceCommand
Scripting_Phone_LINEScriptHasControl

Purpose

This function tells HomeSeer Phone that a script is controlling the call. If called after the first ring, HomeSeer Phone will not answer the call. Ring events will still be fired. The script must answer the call with the function hsp.LINEAnswer.

Parameters

Parameter: Line
Type: Integer
Description: The phone line to access.

Parameter: Mode
Type: Boolean
Description: The mode to set. If TRUE, only the script can answer the call. If FALSE, HomeSeer Phone will answer the call normally. The mode is reset to FALSE before the first ring on each new call.

Returns

None.

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SETsPeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedin
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByName
Scripting_Phone_MBGet
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBDeleteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINESetVoice

HomeSeer HS3 - End User Documentation
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESetDialTones
Scripting_Phone_LINESendDAT
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINEIsSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_ContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount

Home > Scripting > Phone > Scripting_Phone_LINERingCount

Scripting_Phone_LINERingCount

LINERingCount

Purpose

This function returns the number of rings received on the given phone line. This function can be called after a ring event in HomeSeer to take some action after a certain number of rings have been received.

Parameters

Parameter: Line
  Type: Integer
  Description: The phone line to access.

Returns

Return value: Count
  Type: Integer

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByName
Scripting_Phone_MBGet
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBIdeleteMessage
Scripting_Phone_MBSort
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendDTMF
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEGetLastVoiceCommand
Scripting_Phone_LINELastCallerInfo
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEGetLastVoiceCommand
Scripting_Phone_LINEGetLocalCID
Scripting_Phone_LINEHandlerOnHook
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HIPSetMailInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount
LINEResetCallTimeout

Purpose
This function resets the call timeout timer for the current call. The timeout timer is used to disconnect the call in the event the caller hangs up the phone. Most voice modems cannot detect when a caller hangs up.

You may need to call this function if you perform a task that takes longer than the timeout value. The timeout is set in the Phone Setup screen.

Parameters
Parameter: Line
Type: Integer
Description: The phone line to reset the timer on.

Returns
None.

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedln
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByName
Scripting_Phone_MBGet
Scripting_Phone_MBFIRSTUnreadMessage
Scripting_Phone_MBFIRSTReadMessage
Scripting_Phone_MBBcleateMessage
Scripting_Phone_MBCnt
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnswerMode
Scripting_Phone_MbClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEReset
Scripting_Phone_LINEResetRecord
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEStartListening
Scripting_Phone_LINESendDTMF
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEReset

**Purpose**

This function resets the given line and disconnects any call that is in progress. This call differs from the LINEHangup call in that it forces a reset to the line even if the line was already reset.

**Parameters**

Parameter: **Line**
Type: **Integer**
Description: The line to reset.

**Returns**

None.

**Examples**

```
hs.ILINEReset 1
```

See also

- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MBNew
- Scripting_Phone_MBMessageTime
- Scripting_Phone_MBMessageName
- Scripting_Phone_MBMessageLength
- Scripting_Phone_MBMessageFrom
- Scripting_Phone_MBMessageDate
- Scripting_Phone_MBMarkUnRead
- Scripting_Phone_MBMarkRead
- Scripting_Phone_MBGetLoggedIn
**LINERecordStop**

**Purpose**

This function stops recording from the given phone line and saves the WAV information in the given file. **LINERecordStart** must have been called first.

**Parameters**

Parameter: **Line**  
Type: **Integer**  
Description: The phone line to record from.
Returns

None.

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMensajeName
Scripting_Phone_MBMensajeLength
Scripting_Phone_MBMensajeFrom
Scripting_Phone_MBMensajeDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetAddress
Scripting_Phone_MBGetAddress
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESetSendTones
Scripting_Phone_LINESetSendAT
Scripting_Phone_LINESetHasControl
Scripting_Phone_LINESetRingCount
Scripting_Phone_LINESetCallTimeout
Scripting_Phone_LINESetReset
Scripting_Phone_LINESetRecordStart
Scripting_Phone_LINESetMuteRings
Scripting_Phone_LINESetIsSpeaking
Scripting_Phone_LINESetHangup
Scripting_Phone_LINESetGetVoice
Scripting_Phone_LINESetGetDTMFString
Scripting_Phone_LINESetEnableDocument
Scripting_Phone_LINESetDocument
Scripting_Phone_LINESetLastCallerInfo
Scripting_Phone_LINESetLastVoiceCommand
Scripting_Phone_LINESetLastPhoneCID
Scripting_Phone_LINESetLocalCID
Scripting_Phone_LINERecordStart

LINERecordStart

Purpose

This function starts recording from the given phone line. A call must be in progress or the function will return an error string.

Parameters

Parameter: Line
Type: Integer
Description: The phone line to record from.

Parameter: Filename
Type: String
Description: The file name to save the recorded WAV file to. Use the CreateMessageFilename function to create a file that can be read by the HomeSeer Phone application. The message will appear in the HomeSeer Phone message list.

Returns

Return value: Call status
Type: String
Description: Returns an empty string if the call succeeded or an error string if it failed.

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBMGetLoggedin
Scripting_Phone_MBWGetDefault
Scripting_Phone_MBGGetByName
Scripting_Phone_MBGGet
Scripting_Phone_MBFristUnreadMessage
Scripting_Phone_MBFristReadMessage
Scripting_Phone_MBDelteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINENextsTime
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendDAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINEEnableMuteRings
Scripting_Phone_LINEIsSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_CreateContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount

Home > Scripting > Phone > Scripting_Phone_LINEMuteRings

Scripting_Phone_LINEMuteRings

**LINEMuteRings**

**Purpose**

This function sets the ring muting status for the given line. Ring muting is the ability to not pass the incoming ring to the phones inside your home. This feature is only available on hardware that supports it. This includes devices such as the Way2Call Hi-Phone device.

- This feature is not supported on the HomeSeer PCI Voice modem.
- After making this call, the new setting is saved. If the system is restarted, it will use the new setting.

**Parameters**

Parameter: **Line**
Type: Integer
Description: The phone line to retrieve the voice from.

Parameter: **Mode**
Type: Integer
Description: The mute mode to set. Modes are:

- 0 = No muting.
- 1 = Mute only the rings before caller ID is detected. After the second ring, muting is disabled, even if Caller ID is not detected.
- 2 = Mutes all rings. Ring signal is never passed to internal phones.

**Returns**

None.
Examples

```
sub main()
    hsp.LINEMuteRings 1,0       ' disable muting
    hsp.LINEMuteRings 1,1       ' enable muting until Caller ID is detected
end sub
```

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedin
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByName
Scripting_Phone_MBGet
Scripting_Phone_MBfirstUnreadMessage
Scripting_Phone_MBfirstReadMessage
Scripting_Phone_MBdeleteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBcancelPendingNotifications
Scripting_Phone_MBanswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEIsSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFFrom
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDialed
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_LINEIsSpeaking

**Purpose**

This function returns the status of the text-to-speech on the given line. This function may be used to determine if speech is in progress.

**Parameters**

- **Parameter:** Index  
  **Type:** Integer  
  **Description:** The phone line to disconnect.

**Returns**

- **Return value:** Speech status  
  **Type:** Boolean  
  **Description:** Returns TRUE if the text-to-speech is currently speaking a phrase or FALSE if it's not.

**See also**

- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MBNew
- Scripting_Phone_MBMessageTime
- Scripting_Phone_MBMessageName
- Scripting_Phone_MBMessageLength
- Scripting_Phone_MBMessageFrom
- Scripting_Phone_MBMessageDate
- Scripting_Phone_MBMarkUnRead
- Scripting_Phone_MBMarkRead
- Scripting_Phone_MBGetLoggedIn
- Scripting_Phone_MBGetDefault
- Scripting_Phone_MBGetByIndex
- Scripting_Phone_MBGet
- Scripting_Phone_MBFirstUnreadMessage
- Scripting_Phone_MBFirstReadMessage
- Scripting_Phone_MBDeleteMessage
- Scripting_Phone_MBCount
- Scripting_Phone_MBCancelPendingNotifications
- Scripting_Phone_MBAntwerMode
- Scripting_Phone_MailboxClass
- Scripting_Phone_LINEStopSpeaking
- Scripting_Phone_LINEStatus
- Scripting_Phone_LINESetVoice
- Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINESetScriptHasControl
Scripting_Phone_LINESetRingCount
Scripting_Phone_LINESetCallTimeout
Scripting_Phone_LINESetReset
Scripting_Phone_LINESetRecordStop
Scripting_Phone_LINESetRecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_CONTACTClass
Scripting_Phone_CLEARLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount

Scripting_Phone_LINEHangup

**LINEHangup**

**Purpose**

This function disconnects the current call and hangs up the line.

**Parameters**

Parameter: **Line**
Type: **Integer**
Description: The phone line to disconnect.

**Returns**

None.

**Examples**

```plaintext
hsp.LINEHangUp 1
```

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByNumber
Scripting_Phone_MBGet
Scripting_Phone_MBCount
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByNumber
Scripting_Phone_MBGet
**Scripting_Phone_LINEGetVoice**

**Purpose**

This function returns the name of the text-to-speech voice currently in use on the given line. This function may be used with hsp.LINESetVoice to temporarily set a new voice then restore it back to default.

**Parameters**

Parameter: `Line`  
Type: **Integer**  
Description: The phone line to retrieve the voice from.

**Returns**

Return value: **Voice name**  
Type: **String**  
Description: A string that is the name of the currently selected voice.

**See also**

Scripting_Phone_LINEClearDTMF  
Scripting_Phone_WaitMS  
Scripting_Phone_StopListening  
Scripting_Phone_StartListening  
Scripting_Phone_Speak  
Scripting_Phone.LinesetSpeaker  
Scripting_Phone_RestoreSettings  
Scripting_Phone_MBSort  
Scripting_Phone_MBSave  
Scripting_Phone_MBNextUnreadMessage  
Scripting_Phone_MBNextReadMessage  
Scripting_Phone_MBNew  
Scripting_Phone_MBMessageTime  
Scripting_Phone_MBMessageName  
Scripting_Phone_MBMessageLength  
Scripting_Phone_MBMessageFrom  
Scripting_Phone_MBMessageDate  
Scripting_Phone_MBMarkUnRead  
Scripting_Phone_MBMarkRead  
Scripting_Phone_MBGetLoggedIN  
Scripting_Phone_MBGetDefault  
Scripting_Phone_MBGetByName  
Scripting_Phone_MBGet  
Scripting_Phone_MBFirstUnreadMessage  
Scripting_Phone_MBFirstReadMessage  
Scripting_Phone_MBDeleteMessage  
Scripting_Phone_MBCancelPendingNotifications  
Scripting_Phone_MBAnswerMode  
Scripting_Phone_MailboxClass  
Scripting_Phone_LINEStopSpeaking  
Scripting_Phone_LINEStatus  
Scripting_Phone_LINESetVoice  
Scripting_Phone_LINESetRingsCurrent  
Scripting_Phone_LINESetSpeakingSpeed  
Scripting_Phone_LINESetRings  
Scripting_Phone_LINESetGreeting  
Scripting_Phone_LINESetCIDNumber  
Scripting_Phone_LINESetCIDName  
Scripting_Phone_LINESetCIDInfo  
Scripting_Phone_LINESetAnswerMode  
Scripting_Phone_LINESendTones  
Scripting_Phone_LINESendAT  
Scripting_Phone_LINEScriptHasControl  
Scripting_Phone_LINESetRingCount  
Scripting_Phone_LINESetCallTimeout  
Scripting_Phone_LINEReset  
Scripting_Phone_LINERecordStop  
Scripting_Phone_LINERecordStart  
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINEIsSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCrrd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_ContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount

Scripting_Phone_LINEGetDTMFString

**LINEGetDTMFString**

**Purpose**

This function returns a string that is the DTMF (touch-tone) digits received. This is a buffer, and every new touch-tone detected will be added to the buffer. To clear the buffer, call **LINEClearDTMF**.

The "#" and "*" keys return the characters "#" and "*", respectively.

**Parameters**

Parameter: Line
Type: Integer
Description: The phone line to get the buffer from.

**Returns**

Return value: DTMF digits
Type: String

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Purpose

This function returns the number of DTMF keys that have been detected. Use this function to check if any keys on the phone keypad have been pressed.
Parameters

Parameter: Line
Type: Integer
Description: The phone line to check.

Returns

Return value: DTMF value
Type: Integer
Description: The count of the number of DTMF keys that have been detected

Examples

dim digit_count

digit_count = hsp.LINEGetDTMFCount(1)

See also
Scripting_Phone_INLINEClearDTMF
Scripting_Phone_INLINEClear
Scripting_Phone_INLINEStopListening
Scripting_Phone_INLINEStartListening
Scripting_Phone_INLINESpeak
Scripting_Phone_INLINESetSpeaker
Scripting_Phone_INLINEStopListening
Scripting_Phone_INLINEStartListening
Scripting_Phone_INLINESpeak
Scripting_Phone_INLINESetSpeaker
Scripting_Phone_INLINECount
Scripting_Phone_INLINEDial
Scripting_Phone_INLINEAnswerSpeaker
Scripting_Phone_INLINEAnswerLocal

HomeSeer HS3 - End User Documentation
Scripting_Phone_LINEEnableSpeakerPhone

LINEEnableSpeakerPhone

Purpose

This function sends the commands to the telephone hardware to enable speakerphone operation. This command must be issued when the telephone hardware is already connected to the phone line. (e.g. After answering)

Parameters

Parameter: Line
Type: Integer
Description: The phone line to access.

Returns

Parameter: Status
Type: Integer
Description: The status of the call - 0 = Failed, 1 = Success.

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedin
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByUserName
Scripting_Phone_MBGet
Scripting_Phone_MBBNewMessage
Scripting_Phone_MBBNewReadMessage
Scripting_Phone_MBBDeleteMessage
Scripting_Phone_MBBMarkUnRead
Scripting_Phone_LINEDisableSpeakerPhone

**Purpose**

This function disables the speakerphone operation on the telephone hardware.

**Parameters**

Parameter: **Line**
Type: **Integer**
Description: The phone line to access.

**Returns**

Parameter: **Status**
Type: **Integer**
Description: The status of the call - 0 = Failed, 1 = Success.

See also:
- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MBNew
- Scripting_Phone_MBMessageTime
- Scripting_Phone_MBMessageName
- Scripting_Phone_MBMessageLength
- Scripting_Phone_MBMessageFrom
- Scripting_Phone_MBMessageDate
- Scripting_Phone_MBMarkUnRead
- Scripting_Phone_MBMarkRead
- Scripting_Phone_MBGetLoggedin
- Scripting_Phone_MBGetDefault
- Scripting_Phone_MBGetByIndex
- Scripting_Phone_MBGet
- Scripting_Phone_MBfirstUnreadMessage
- Scripting_Phone_MBfirstReadMessage
- Scripting_Phone_MBCount
- Scripting_Phone_MBCancelPendingNotifications
- Scripting_Phone_MBAnswerMode
- Scripting_Phone_MailboxClass
- Scripting_Phone_LINEStopSpeaking
- Scripting_Phone_LINEStatus
- Scripting_Phone_LINESetVoice
- Scripting_Phone_LINESetRingsCurrent
- Scripting_Phone_LINESetSpeakingSpeed
- Scripting_Phone_LINESetRings
- Scripting_Phone_LINESetGreeting
- Scripting_Phone_LINESetCIDNumber
- Scripting_Phone_LINESetCIDName
- Scripting_Phone_LINESetCIDInfo
- Scripting_Phone_LINESetAnswerMode
- Scripting_Phone_LINESendTones
- Scripting_Phone_LINESendAT
- Scripting_Phone_LINEScriptHasControl
- Scripting_Phone_LINERingCount
- Scripting_Phone_LINEResetCallTimeout
- Scripting_Phone_LINEReset
- Scripting_Phone_LINEReorderStop
- Scripting_Phone_LINEReorderStart
- Scripting_Phone_LINEMuteRings
- Scripting_Phone_LINELsSpeaking
- Scripting_Phone_LINExHangup
- Scripting_Phone_LINEGetVoice
- Scripting_Phone_LINEGetDTMFString
- Scripting_Phone_LINEGetDTMFCount
- Scripting_Phone_LINEEnableSpeakerPhone
- Scripting_Phone_LINENumber
- Scripting_Phone_LINEDial
- Scripting_Phone_LINEAnswerSpeakerPhone
- Scripting_Phone_LINEAnswerLocal
- Scripting_Phone_LINEAnswer
- Scripting_Phone_LastVoiceMailInfo
- Scripting_Phone_LastCallerInfo
- Scripting_Phone_HIPSetCallWaitingLED
- Scripting_Phone_HIPSendLocalCID
- Scripting_Phone_HIPCmd
- Scripting_Phone_HandsetOnHook
- Scripting_Phone_GetLastVoiceCommand
- Scripting_Phone_CreateMessageFilename
- Scripting_Phone_ContactClass
- Scripting_Phone_ClearLastVoiceCommand
- Scripting_Phone_CIDNumber
Scripting_Phone_LINECount

**NumLines**

**Purpose**

This function returns a count of the number of phone lines configured in the system.

**Parameters**

None.

**Returns**

Return value: **Number**  
Type: **Integer**

See also

- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MBNew
- Scripting_Phone_MBMessageTime
- Scripting_Phone_MBMessageName
- Scripting_Phone_MBMessageLength
- Scripting_Phone_MBMessageFrom
- Scripting_Phone_MBMessageDate
- Scripting_Phone_MBMarkUnRead
- Scripting_Phone_MBMarkRead
- Scripting_Phone_MBGetLoggedIn
- Scripting_Phone_MBGetDefault
- Scripting_Phone_MBGetByName
- Scripting_Phone_MBGet
- Scripting_Phone_MBFirstUnreadMessage
- Scripting_Phone_MBFirstReadMessage
- Scripting_Phone_MBDeleteMessage
- Scripting_Phone_MBCount
- Scripting_Phone_MBCancelPendingNotifications
- Scripting_Phone_MBAnterMode
- Scripting_Phone_MailboxClass
- Scripting_Phone_LINESetVoice
- Scripting_Phone_LINESetRingsCurrent
- Scripting_Phone_LINESetRings
- Scripting_Phone_LINESetGreeting
- Scripting_Phone_LINESetCIDNumber
- Scripting_Phone_LINESetCIDName
- Scripting_Phone_LINESetCIDInfo
- Scripting_Phone_LINESetAnswerMode
- Scripting_Phone_LINESendTones
- Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINEDial

**Purpose**

This function dials the phone number given. If the line is currently in use, the error LINE_INUSE is returned.

- Voice modems can't detect when the calling party has actually answered the phone. This function will return the LINE_CONNECTED status when the modem actually starts ringing the line. If the line is busy, the LINE_BUSY error is returned. Note that the LINE_BUSY status can only be returned if the "nowait" parameter is set to FALSE. If "nowait" is set to TRY, the function returns immediately with the LINE_CONNECTED status. This is useful for quick dialing when you don't want to check for the line being busy.

The `hangup` parameter is used with the HomeSeer Phone switch. If this parameter is TRUE, the line is hung up immediately after dialing is complete. This will cause the switch box to reconnect the local phone.

**Parameters**

- **Parameter:** Line
  - **Type:** Integer
  - **Description:** The phone line to access.

- **Parameter:** Number
  - **Type:** String
  - **Description:** The phone number to dial. Note that the Windows dialing properties are used to alter the phone number. If the number given includes the area code, and the area code matches the one listed in the dialing properties, it is removed.

- **Parameter:** Hangup
  - **Type:** Boolean
  - **Description:** If TRUE, causes the modem to hang up immediately after dialing the number.

- **Parameter:** Nowait
  - **Type:** Boolean
  - **Description:** If TRUE, the call returns immediately regardless if the remote party has answered. This is useful if you are connected to a PBX system and you know the PBX answers immediately. This parameter is optional, and if omitted, the system assumes a FALSE value.

  If FALSE, the system will wait up to 8 seconds for the remote party to connect. Since voice modems do not notify the system when a connection is
made, the delay gives time for a connection. You may have to add more of a delay in your script.

**Returns**

Return value: **Status**
Type: **Long (.NET Integer)**
Description: Returns a value to indicate the status of the line:

- 0 = LINE_IDLE
- 1 = LINE_OFFERING
- 2 = LINE_RINGING
- 3 = LINE_CONNECTED
- 4 = LINE_INACTIVE
- 5 = LINE_BUSY
- 6 = LINE_INUSE
- 7 = LINE_TIMEOUT (for calling)
- 8 = LINE_ERROR (line error event)
- 9 = LINE_DIALING (dialing out)
- 10 = LINE_REORDER (fast busy, Hi-Phone only)

**Examples**

The following example dials a number and speaks over the phone.

```vbnet
sub main()
    dim r
    dim voice
    r=hsp.LINEDial(1,"555-1212",false,false)
    if r=5 then
        ' line is busy
        hs.WriteLog "Dial Error","Line busy"
        exit sub
    end if
    hsp.waitms 1000
    hsp.speak 1,"hello on the phone, how are you today?",true
    hsp.LINEHangup 1
end sub
```

See also

- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MBNew
- Scripting_Phone_MBMessageTime
- Scripting_Phone_MBMessageName
- Scripting_Phone_MBMessageLength
- Scripting_Phone_MBMessageFrom
- Scripting_Phone_MBMessageDate
- Scripting_Phone_MBMarkUnRead
- Scripting_Phone_MBMarkRead
- Scripting_Phone_MBGetLoggedin
- Scripting_Phone_MBGetDefault
- Scripting_Phone_MBGetByName
- Scripting_Phone_MBGet
- Scripting_Phone_MBFIRSTUnreadMessage
- Scripting_Phone_MBFIRSTReadMessage
- Scripting_Phone_MBDeleteMessage
- Scripting_Phone_MBCount
- Scripting_Phone_MBCancelPendingNotifications
- Scripting_Phone_MBSM_BAnswerMode
- Scripting_Phone_MailboxClass
- Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEAnswerSpeakerPhone

**Purpose**

This function forces HomeSeer Phone to answer an external call, and it sets up the telephone interface for SpeakerPhone operation. Use hsp.HandsetOnHook to detect when the user hangs up the phone.

**Parameters**

- **Parameter:** Line
  - **Type:** Integer
  - **Description:** The phone line to access.

**Returns**

- **Parameter:** Status
  - **Type:** Integer
  - **Description:** The status of the call - 0 = Failed, 1 = Success.

**Examples**
See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetsName
Scripting_Phone_MBGets
Scripting_Phone_MBGet
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBDeleteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnterMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINEIsSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_ContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_ContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_LINEAnswerLocal

LINEAnswerLocal

Purpose

This function forces HomeSeer Phone to answer an internal call. This switches the handset and the modem in and switches out the line. If you are going to take control using a script, call `hsp.LINEScriptHasControl 1, TRUE`. This will allow you to send text-to-speech (TTS) audio to the handset. Use `hsp.HandsetOnHook` to detect when the user hangs up the phone.

Parameters

Parameter: Line
Type: Integer
Description: The phone line to access.

Returns

None.

Examples

```
hsp.LINEScriptHasControl 1
hsp.LINEAnswerLocal 1
hsp.Speak 1, "hello on the handset", TRUE
hsp.LINEReset 1
```

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedln
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByName
Scripting_Phone_MBBGet
Scripting_Phone_MBFIRSTUnreadMessage
Scripting_Phone_MBFIRSTReadMessage
Scripting_Phone_MBBDeleteMessage
Scripting_Phone_MBBCount
**LINEAnswer**

**Purpose**

This function answers a call. The line must be in the offering or ringing state before calling this function.

**Parameters**

- **Parameter:** Line<br>
  **Type:** `Integer`<br>
  **Description:** The phone line to answer.

**Returns**

- **Return value:** Status<br>
  **Type:** `Integer`<br>
  **Description:** Returns a zero (0) if there's no error or a non-zero value if there is an error.
Scripting_Phone_LastVoiceMailInfo

LastVoiceMailInfo

Purpose

This function returns the information on the last voice mail message left in the system, using the format specified in the HomeSeer Phone "Last Voice Mail Message" format box in the Phone Settings screen.

Parameters

Parameter: XML information
Type: Boolean
Description: If TRUE, the output is in an XML format suitable for use with other applications.

Returns

Return value: Voicemail information
Type: String
Description: String formatted according to the format string specification or in XML format.

Examples

If your "Last Voice Mail Message" format string is this:

On #CallDate# at #CallTime# you received a call from #CallFrom# at #CallNumber#, and a message #Length# in length was left in #MailBox# mailbox.

Your returned string would appear similar to this:

On Thu, Jan 1, 2005 at 1:23 PM you received a call from Smith, John at 603-555-1234, and a message 0:27 in length was left in Joe's mailbox.

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBnextUnreadMessage
Scripting_Phone_MBnextReadMessage
Scripting_Phone_MBNNew
Scripting_Phone_MBBMessageTime
Scripting_Phone_MBBMessageName
Scripting_Phone_MBBMessageLength
Scripting_Phone_MBBMessageFrom
Scripting_Phone_MBBMessageDate
Scripting_Phone_MBBmarkUnRead
Scripting_Phone_MBBmarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByName
Scripting_Phone_MBGet
Scripting_Phone_MBfirstUnreadMessage
Scripting_Phone_MBfirstReadMessage
Scripting_Phone_MBDDeleteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAanswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LastCallerInfo

LastCallerInfo

Purpose

This function returns the information on the last phone call received on a specific HomeSeer Phone line number, using the format specified in the HomeSeer Phone "Last Caller Message" format box in the Phone Settings screen.

Parameters

Parameter: Line
Type: Integer (.NET Short)
Description: This is the line number that you wish to retrieve the call info from.

Returns

Return value: Call information
Type: String
Description: String formatted according to the format string specification.

Examples

If your "Last Caller Message" format string is this:

#CallerIDName# called at #CallerIDTimeDate# from #CallerIDNumber#
Smith, John called at 5/24/2005 11:32:41 AM from 603-555-1234

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByUserName
Scripting_Phone_MBGet
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBDelMessage
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAntiAnswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESetSendDTMF
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINeIsSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFSentence
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEGet
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_ContactClass
Scripting_Phone_Contact
Scripting_Phone_IDNumber
Scripting_Phone_IDName
Scripting_Phone_ARDSave
Scripting_Phone_HIPSetCallWaitingLED

**HIPSetCallWaitingLED**

**Purpose**

- This command will only work with the PRO edition of the HS2 software.

  Set the call waiting LED on a phone connected to the phone jack on the Way2Call Hi-Phone device. If the "led_on" parameter is TRUE the led is turned on, otherwise the led is turned off.

**Parameters**

- Parameter: Line
  - Type: Integer
- Parameter: led_on
  - Type: Boolean

**Returns**

None.

**See Also**

- HIPSendLocalCID
- HIPCmd

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_LINEGet
Scripting_Phone_LINEListen
Scripting_Phone_LINEStartListening
Scripting_Phone_LINESetMicrophone
Scripting_Phone_LINESetVoice
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_HIPSendLocalCID

HIPSendLocalCID

Purpose

- This command will only work with the PRO edition of the HS2 software.

Generate caller ID information to be displayed on the phones connected to the phone jack on the Way2Call Hi-Phone device.

Parameters

Parameter: Line
Type: Integer

Parameter: Name
Type: string

Parameter: Number
Type: string

Returns

None.
See Also

HIPSetCallWaitingLED
HIPCmd

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBsort
Scripting_Phone_MBsave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBmessageTime
Scripting_Phone_MBmessageName
Scripting_Phone_MBmessageLength
Scripting_Phone_MBmessageFrom
Scripting_Phone_MBmessageDate
Scripting_Phone_MBmarkUnRead
Scripting_Phone_MBmarkRead
Scripting_Phone_MGetLoggedln
Scripting_Phone_MGetDefault
Scripting_Phone_MGetByNumber
Scripting_Phone_MGet
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBDeleteMessage
Scripting_Phone_MBcount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnterMode
Scripting_Phone_MMailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetRingsSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESetDTMF
Scripting_Phone_LINESetAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINEIsSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_contactClass
Scripting_Phone_clearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRLoad
Scripting_Phone_HIPCmd

**HIPCmd**

**Purpose**

This function sends a Hi-Phone specific command to the Way2Call Hi-Phone device.

**Parameters**

- **Parameter:** Line
  - **Type:** Integer
- **Parameter:** Code
  - **Type:** Long
- **Parameter:** Data
  - **Type:** Long

**Returns**

None.

**See Also**

- HIPSendLocalCID
- HIPSetCallWaitingLED

See also

- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_setSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNew
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MBMessageTime
- Scripting_Phone_GBMessageName
- Scripting_Phone_GBMessageLength
- Scripting_Phone_GBMessageFrom
- Scripting_Phone_GBMessageDate
- Scripting_Phone_GBMarkUnRead
- Scripting_Phone_GBMarkRead
- Scripting_Phone_MBGetLoggedln
- Scripting_Phone_MBGetDefault
- Scripting_Phone_MBGetByName
- Scripting_Phone_MBGet
- Scripting_Phone_MBFirstUnreadMessage
- Scripting_Phone_MBFirstReadMessage
- Scripting_Phone_MBDeleteMessage
- Scripting_Phone_MBCount
- Scripting_Phone_MBCancelPendingNotifications
- Scripting_Phone_MBAnswerMode
- Scripting_Phone_MailboxClass
- Scripting_Phone_LINEStopSpeaking
- Scripting_Phone_LINEStatus
HandsetOnHook

Purpose

This function returns the status of the local handset. This is the handset connected to the phone jack on the modem device.

Parameters

Parameter: Line
Type: Integer
Description: The phone line to access.

Returns

Return value: Status of line
Type: Boolean
Description: Return TRUE if the local handset is on-hook (not in use) or FALSE if the handset is off-hook (in use).

Examples

dim hook_stat
hook_stat = hsp.HandsetOnHook(1)

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SettSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBSaveLoggedin
Scripting_Phone_MBSaveDefault
Scripting_Phone_MBSaveByName
Scripting_Phone_MBSave
Scripting_Phone_MBGet
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBDeleteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBSavePendingNotifications
Scripting_Phone_MBSaveAnswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetRingsSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESetTones
Scripting_Phone_LINESetAT
Scripting_Phone_LINESetScriptHasControl
Scripting_Phone_LINESetRingCount
Scripting_Phone_LINESetCallTimeout
Scripting_Phone_LINESet
Scripting_Phone_LINESetRecordStop
Scripting_Phone_LINESetRecordStart
Scripting_Phone_LINESetMuteRings
Scripting_Phone_LINESetSpeaking
Scripting_Phone_LINESetHangup
Scripting_Phone_LINESetDTMFS
Scripting_Phone_LINESetDTMFCount
Scripting_Phone_LINESetEnableSpeakerPhone
Scripting_Phone_LINESetDisableSpeakerPhone
Scripting_Phone_LINESetCount
Scripting_Phone_LINESetDial
Scripting_Phone_LINESetAnswerSpeakerPhone
Scripting_Phone_LINESetAnswerLocal
Scripting_Phone_LINESetAnswer
Scripting_Phone_LINESetLastVoiceMailInfo
Scripting_Phone_LINESetLastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_ContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_GetLastVoiceCommand

GetLastVoiceCommand

Purpose

This function returns the phrase last recognized over the specified phone line.

Parameters

Parameter: Line
Type: Integer
Description: The phone line to access.

Returns

Return value: Phrase
Type: String

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBSend
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLogged
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetBy
Scripting_Phone_MBGet
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBSendMailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINESetScriptHasControl
Scripting_Phone_LINESetRingCount
Scripting_Phone_LINESetCallTimeout
Scripting_Phone_LINESetReset
**Scripting_Phone_CreateMessageFilename**

**CreateMessageFilename**

**Purpose**

This function creates a file name for a voice message. The voice message is required to be formatted properly so it can be read by the application. The file name encodes who the message is for, the time it was received, etc.

**Parameters**

- **Parameter:** Username  
  **Type:** String  
  **Description:** The user name of the mailbox the message is to be sent to.

- **Parameter:** Number  
  **Type:** Integer  
  **Description:** The phone number of the caller, usually from Caller ID.

- **Parameter:** Name  
  **Type:** String  
  **Description:** The name of the caller, usually from Caller ID.

**Returns**

- **Return value:** Filename  
  **Type:** String  
  **Description:** A string that is the complete path to the file. This string may be passed to the LINERecordStart function to start recording a voice message.

**See also**

Scripting_Phone_LINEClearDTMF  
Scripting_Phone_WaitMS  
Scripting_Phone_StopListening  
Scripting_Phone_StartListening  
Scripting_Phone_Speak  
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBBNew
Scripting_Phone_MBBMessageTime
Scripting_Phone_MBBMessageName
Scripting_Phone_MBBMessageLength
Scripting_Phone_MBBMessageFrom
Scripting_Phone_MBBMessageDate
Scripting_Phone_MBBMarkUnRead
Scripting_Phone_MBBMarkRead
Scripting_Phone_MBBGetLoggedIn
Scripting_Phone_MBBGetDefault
Scripting_Phone_MBBGetByName
Scripting_Phone_MBBGet
Scripting_Phone_MBBFirstUnreadMessage
Scripting_Phone_MBBFirstReadMessage
Scripting_Phone_MBBDeleteMessage
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAnswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESetTones
Scripting_Phone_LINESetDT
Scripting_Phone_LINESetHasControl
Scripting_Phone_LINESetRingCount
Scripting_Phone_LINESetCallTimeout
Scripting_Phone_LINESetStop
Scripting_Phone_LINESetRecordStop
Scripting_Phone_LINESetRecordStart
Scripting_Phone_LINESetMuteRings
Scripting_Phone_LINESetSpeaking
Scripting_Phone_LINESetHangup
Scripting_Phone_LINESetGetVoice
Scripting_Phone_LINESetDTMFString
Scripting_Phone_LINESetDTMFCount
Scripting_Phone_LINESetEnableSpeakerPhone
Scripting_Phone_LINESetDisableSpeakerPhone
Scripting_Phone_LINESetCount
Scripting_Phone_LINESetDial
Scripting_Phone_LINESetAnswerSpeakerPhone
Scripting_Phone_LINESetAnswerLocal
Scripting_Phone_LINESetAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_ContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount

Home > Scripting > Phone > Scripting_Phone_ContactClass

Scripting_Phone_ContactClass
## Contact Class

### Contact Object

HomeSeer Phone keeps an internal address book that is loaded when the program is started. Each address book entry has numerous properties that can be read and set. The class name is contact, and has the following properties:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>announcement</td>
<td>The announcement to play when this caller calls. Either a text-to-speech string or full path to a wav file.</td>
</tr>
<tr>
<td>announce_local_wav</td>
<td>Text or wave file that will be played when this caller calls. The announcement is played through HomeSeer.</td>
</tr>
<tr>
<td>answer_rings</td>
<td>Set to TRUE to answer this call on the answer_rings_count, else FALSE to ignore</td>
</tr>
<tr>
<td>answer_rings_count</td>
<td>The number of rings to answer when this caller calls</td>
</tr>
<tr>
<td>business_phone</td>
<td>Business phone number</td>
</tr>
<tr>
<td>business_phone_2</td>
<td>2nd Business phone number</td>
</tr>
<tr>
<td>cell_phone</td>
<td>Cell phone number</td>
</tr>
<tr>
<td>cell_phone_2</td>
<td>2nd Cell phone number</td>
</tr>
<tr>
<td>custom1</td>
<td>A custom field</td>
</tr>
<tr>
<td>custom2</td>
<td>A custom field</td>
</tr>
<tr>
<td>email_address</td>
<td>email address</td>
</tr>
<tr>
<td>email_address_2</td>
<td>2nd email address</td>
</tr>
<tr>
<td>email_address_3</td>
<td>3rd email address</td>
</tr>
<tr>
<td>FIRST</td>
<td>First name</td>
</tr>
<tr>
<td>LAST</td>
<td>Last name</td>
</tr>
<tr>
<td>company_name</td>
<td>Company name</td>
</tr>
<tr>
<td>home_phone</td>
<td>Home phone number</td>
</tr>
<tr>
<td>home_phone_2</td>
<td>2nd Home phone number</td>
</tr>
<tr>
<td>fax_home</td>
<td>Home fax phone number</td>
</tr>
<tr>
<td>fax_work</td>
<td>Work fax phone number</td>
</tr>
<tr>
<td>pager_phone</td>
<td>Pager phone number</td>
</tr>
<tr>
<td>pager_pin</td>
<td>Pager PIN</td>
</tr>
<tr>
<td>cid_group_category</td>
<td>Currently not used in HomeSeer Phone, this field can be used by scripts to control answering and voicemail functions on groups of address book entries. e.g. Enter “Family” here for all family members, and then a script can control allowing the phones to ring when a family member calls.</td>
</tr>
<tr>
<td>home_address_1 through home_address_3</td>
<td>Three home address fields for (typically) street address information.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>home_city</td>
<td>Home city name</td>
</tr>
<tr>
<td>home_state_province</td>
<td>Home state or province</td>
</tr>
<tr>
<td>home_postal_code</td>
<td>Home postal code for an address</td>
</tr>
<tr>
<td>home_country</td>
<td>Home country name for an address</td>
</tr>
<tr>
<td>business_address_1</td>
<td>Three business address fields for (typically) street address information.</td>
</tr>
<tr>
<td>business_address_3</td>
<td></td>
</tr>
<tr>
<td>business_city</td>
<td>Business city name</td>
</tr>
<tr>
<td>business_state_province</td>
<td>Business state or province</td>
</tr>
<tr>
<td>business_postal_code</td>
<td>Business postal code for an address</td>
</tr>
<tr>
<td>business_country</td>
<td>Business country name for an address</td>
</tr>
<tr>
<td>cid_name</td>
<td>Name to match in the caller ID name field when a call arrives. Some phone systems may out a special string in this field like “marketing”. If this field is present and matches the caller ID name field, the address entry will match the call. This field is the “Name Matching” field on the “Phone (CID Matching)” tab in the address book.</td>
</tr>
<tr>
<td>EnableRingPattern</td>
<td>Supported only on the Hi-Phone device. If set to TRUE, and an incoming call matches this address book entry, the phones in the home will ring with pattern as set in the RingON, RingOFF, RingDelay properties.</td>
</tr>
<tr>
<td>RingON, RingOFF, RingDelay</td>
<td>Hi-Phone only. These properties specify the ring pattern when the EnableRingPattern property is set to TRUE. The times are in 1/10 of a second.</td>
</tr>
<tr>
<td>cidflags</td>
<td>cidflags bit definitions:</td>
</tr>
<tr>
<td></td>
<td>CT_CIDFLAGS_BLOCKED = 1            &quot;callers with this CID are blocked&quot;</td>
</tr>
<tr>
<td></td>
<td>CT_CIDFLAGS_ANNOUNCE = 2          &quot;this ID is announced&quot;</td>
</tr>
<tr>
<td></td>
<td>CT_CIDFLAGS_SPARE1 = 4            &quot;spare, not used&quot;</td>
</tr>
<tr>
<td></td>
<td>CT_CIDFLAGS_SPEC_ANN = 8          &quot;play special announcement to caller&quot;</td>
</tr>
<tr>
<td></td>
<td>CT_CIDFLAGS_POPUP = 4H10          &quot;pop up window with CID info&quot;</td>
</tr>
<tr>
<td></td>
<td>CT_CIDFLAGS_PRIVATE = 6H20        &quot;entry matches private CID calls&quot;</td>
</tr>
<tr>
<td></td>
<td>CT_CIDFLAGS_OUTOFAREA = 6H40     &quot;entry matches out of area calls&quot;</td>
</tr>
<tr>
<td>Misc1 through Misc6</td>
<td>Miscellaneous string fields for use by scripts to store information. These fields are saved in the address book file but do not appear in the user interface.</td>
</tr>
<tr>
<td>MiscNum1 through MiscNum4</td>
<td>Miscellaneous long integer fields for use by scripts to store information. These fields are saved in the address book file but do not appear in the user interface.</td>
</tr>
<tr>
<td>flags</td>
<td>flags bit definitions:</td>
</tr>
<tr>
<td></td>
<td>CT_FLAGS_VRENABLED = 1            &quot;address book entry is enabled as voice command&quot;</td>
</tr>
<tr>
<td></td>
<td>CT_FLAGS_HANGUP = 2              &quot;hang up after speaking announcement&quot;</td>
</tr>
</tbody>
</table>

See also
- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
**Purpose**

This function clears out the last voice command recognized to an empty string.

**Parameters**

Parameter: **Line**  
Type: **Integer**  
Description: The phone line to access.

**Returns**

None.

See also

- Scripting_Phone_LINEClearDTMF
- Scripting_Phone_WaitMS
- Scripting_Phone_StopListening
- Scripting_Phone_StartListening
- Scripting_Phone_Speak
- Scripting_Phone_SetSpeaker
- Scripting_Phone_RestoreSettings
- Scripting_Phone_MBSort
- Scripting_Phone_MBSave
- Scripting_Phone_MBNextUnreadMessage
- Scripting_Phone_MBNextReadMessage
- Scripting_Phone_MBNew
- Scripting_Phone_MBMessageTime
- Scripting_Phone_MBMessageName
- Scripting_Phone_MBMessageLength
- Scripting_Phone_MBMessageFrom
- Scripting_Phone_MBMessageDate
- Scripting_Phone_MBMarkUnRead
- Scripting_Phone_MBMarkRead
- Scripting_Phone_MGetLoggedln
- Scripting_Phone_MGetDefault
- Scripting_Phone_MGetByName
- Scripting_Phone_MGet
- Scripting_Phone_MFirstUnreadMessage
- Scripting_Phone_MFirstReadMessage
- Scripting_Phone_MDeleteMessage
- Scripting_Phone_MCount
- Scripting_Phone_MBCancelPendingNotifications
- Scripting_Phone_MBAnswerMode
- Scripting_Phone_MailboxClass
- Scripting_Phone_LINEStopSpeaking
- Scripting_Phone_LINEStatus
- Scripting_Phone_LINESetVoice
- Scripting_Phone_LINESetRingsCurrent
- Scripting_Phone_LINESetSpeakingSpeed
- Scripting_Phone_LINESetRings
- Scripting_Phone_LINESetGreeting
- Scripting_Phone_LINESetCIDNumber
- Scripting_Phone_LINESetCIDName
- Scripting_Phone_LINESetCIDInfo
- Scripting_Phone_LINESetAnswerMode
- Scripting_Phone_LINESendTones
- Scripting_Phone_LINESendAT
- Scripting_Phone_LINEScriptHasControl
- Scripting_Phone_LINEResetCallTimeout
- Scripting_Phone_LINEResetCount
- Scripting_Phone_LINEReset
- Scripting_Phone_LINERecordStop
- Scripting_Phone_LINERecordStart
- Scripting_Phone_LINEMuteRings
- Scripting_Phone_LINEIsSpeaking
- Scripting_Phone_LINEHangup
- Scripting_Phone_LINEGetVoice
- Scripting_Phone_LINEGetDTMFString
- Scripting_Phone_LINEGetDTMFCount
- Scripting_Phone_LINEEnableSpeakerPhone
- Scripting_Phone_LINEDisableSpeakerPhone
- Scripting_Phone_LINECount
- Scripting_Phone_LINEDialed
- Scripting_Phone_LINEAnswerSpeakerPhone
- Scripting_Phone_LINEAnswerLocal
- Scripting_Phone_LINEAnswer
- Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_CIDNumber

**Purpose**

This function returns the Caller ID number returned from the last (or current) call. This is reset when a new call arrives. The number will only be available if you have the Caller ID service. It may also be some other string like Private or Out or Area if the call was blocked.

**Parameters**

Parameter: Line
Type: Integer
Description: The phone line to retrieve the information from.

**Returns**

Return value: Phone number
Type: String
Description: A string that is the callers phone number, or an empty string if the information was not available.

**Examples**

' get the Caller ID number

dim cnumber

cnumber = hsp.CIDNumber(1)

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBSort
Scripting_Phone_MBGetLogin
Scripting_Phone_MBSetDefault
Scripting_Phone_MBGetDefault
This function returns the Caller ID name returned from the last (or current) call. This is reset when a new call arrives. The name will only be available if your Caller ID service supplies names.

**Parameters**

- **Parameter:** Line  
  **Type:** Integer  
  **Description:** The phone line to retrieve the information from.
Returns

Return value: Name
Type: String
Description: The caller's name or an empty string if the information was not available.

Examples

'get the Caller ID name

dim cname

cname = hsp.CIDName(1)

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_Sett Speaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessagetime
Scripting_Phone_MBMessagelength
Scripting_Phone_MBMessagename
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedin
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetAddress
Scripting_Phone_MBGetByName
Scripting_Phone_MBGet
Scripting_Phone_MBFIRSTunreadMessage
Scripting_Phone_MBFIRSTreadMessage
Scripting_Phone_MBDel etaMessage
Scripting_Phone_MBCound
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBA nswerMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpea kingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINERingCount
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINEISpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEDial
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_ADRSave

**ADRSave**

**Purpose**

This function saves all configured address book information. This is useful if a script modifies any properties of an address book entry.

**Parameters**

None.

**Returns**

None.

See also

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByName
Scripting_Phone_MBGet
Scripting_Phone_MBFirstUnreadMessage
Scripting_Phone_MBFirstReadMessage
Scripting_Phone_MBDelteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAwetMode
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINESetRings
Scripting_Phone_ADRNew

ADRNew

Purpose

This function returns a reference to a new address book entry (contact).

Parameters

None.

Returns

Return value: Contact class
Type: Object

See also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLogged
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetAddress
Scripting_Phone_MBGet
Scripting_Phone_MBFIRSTUnreadMessage
Scripting_Phone_MBFIRSTReadMessage
Scripting_Phone_MBDelMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
Scripting_Phone_MBAsterMode
Scripting_Phone_MBMboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINESendTones
Scripting_Phone_LINESendAT
Scripting_Phone_LINEScriptHasControl
Scripting_Phone_LINESetRingCount
Scripting_Phone_LINESetCallTimeout
Scripting_Phone_LINESetReset
Scripting_Phone_LINESetRecordStop
Scripting_Phone_LINESetRecordStart
Scripting_Phone_LINESetMuteRings
Scripting_Phone_LINESetSpeaking
Scripting_Phone_LINESetHangup
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetDtmfString
Scripting_Phone_LINESetDtmfCount
Scripting_Phone_LINESetEnableSpeakerPhone
Scripting_Phone_LINESetDisableSpeakerPhone
Scripting_Phone_LINESetCount
Scripting_Phone_LINESetDial
Scripting_Phone_LINESetAnswerSpeakerPhone
Scripting_Phone_LINESetAnswerLocal
Scripting_Phone_LINESetAnswer
Scripting_Phone_LINESetLastVoiceMailInfo
Scripting_Phone_LINESetLastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_GetLastVoiceCommand
Scripting_Phone_CreateMessageFilename
Scripting_Phone_ContactClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
Scripting_Phone_ADRCount

Scripting_Phone_ADRGet
ADRGet

Purpose

This function returns the reference to an address book entry (contact).

Parameters

Parameter: Index
Type: Integer
Description: The index number of the address book entry to get.

Returns

Return value: Contact class
Type: Object

See also

Scripting Phone LINEClearDTMF
Scripting Phone WaitMS
Scripting Phone StopListening
Scripting Phone StartListening
Scripting Phone Speak
Scripting Phone SetSpeaker
Scripting Phone RestoreSettings
Scripting Phone MSave
Scripting Phone MBnextUnreadMessage
Scripting Phone MBnextReadMessage
Scripting Phone MBNew
Scripting Phone MBMessageTime
Scripting Phone MBMessageName
Scripting Phone MBMessageLength
Scripting Phone MBMessageFrom
Scripting Phone MBMessageDate
Scripting Phone MBMarkUnRead
Scripting Phone MBMarkRead
Scripting Phone MBGetLoggedin
Scripting Phone MBGetDefault
Scripting Phone MBGetByName
Scripting Phone MBGet
Scripting Phone MBFirstUnreadMessage
Scripting Phone MBFirstReadMessage
Scripting Phone MBDeleteMessage
Scripting Phone MBCount
Scripting Phone MBCancelPendingNotifications
Scripting Phone MBAnswerMode
Scripting Phone MailboxClass
Scripting Phone LINEStopSpeaking
Scripting Phone LINEStatus
Scripting Phone LINERing
Scripting Phone LINERingCurrent
Scripting Phone LINESetRings
Scripting Phone LINESetRingsCurrent
Scripting Phone LINESetSpeakingSpeed
Scripting Phone LINESetRings
Scripting Phone LINESetRinging
Scripting Phone LINESetCIDNumber
Scripting Phone LINESetCIDName
Scripting Phone LINESetCIDInfo
Scripting Phone LINESetAnswerMode
Scripting Phone LINEReSet
Scripting Phone LINEReSetTones
Scripting Phone LINECount
Scripting Phone LINEEnd
Scripting Phone LINEHasControl
Scripting Phone LINERingCount
Scripting Phone LINEResetCallTimeout
Scripting Phone LINEReset
Scripting Phone LINERecordStop
Scripting Phone LINERecordStart
Scripting Phone LINEMuteRings
Scripting Phone LINEIsTalking
Scripting Phone LINEHangup
Scripting Phone LINEGetVoice
Scripting Phone LINEGetDTMFString
Scripting Phone LINEGetDTMFCount
Scripting Phone LINEEnableSpeakerPhone
Scripting Phone LINEDisableSpeakerPhone
Scripting Phone LINECount
Scripting Phone LINEDial
Scripting Phone LINEAnswerSpeakerPhone
Scripting_Phone_ADRDelete

**ADRDelete**

**Purpose**

This function deletes an address book entry given its index. Retrieve the proper index by calling hsp.ADRCount then hsp.ADRGet to search for the proper index.

- You can't delete the private and out-of-area address book entries.

**Parameters**

- **Parameter:** Index
  - **Type:** Integer
  - **Description:** The index number of the address book entry to delete.

**Returns**

None.

**See also**

Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedln
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByName
Scripting_Phone_MBGet
Scripting_Phone_MBFIRSTUnreadMessage
Scripting_Phone_MBFIRSTReadMessage
Scripting_Phone_MBDeleteMessage
Scripting_Phone_MBCount
Scripting_Phone_MBCancelPendingNotifications
scripting_phone_mbanwermode
scripting_phone_mailboxclass
scripting_phone_linesetstopspeaking
scripting_phone_linesetstatus
scripting_phone_linesetvoice
scripting_phone_linesettsetringscurrent
scripting_phone_linesettsetspeaking-speed
scripting_phone_linesettsetrings
scripting_phone_linesettsetgreeting
scripting_phone_linesettsetcidnumber
scripting_phone_linesettsetcidname
scripting_phone_linesettsetcidinfo
scripting_phone_linesettsetanswermode
scripting_phone_linesettsettones
scripting_phone_linesettsetat
scripting_phone_linesettscripthascontrol
scripting_phone_linesettsetringcount
scripting_phone_linesettresetcalltimeout
scripting_phone_linesettreset
scripting_phone_linesettrecordstop
scripting_phone_linesettrecordstart
scripting_phone_linesettmuterings
scripting_phone_linesettisspeaking
scripting_phone_linesetthangup
scripting_phone_linesettgetvoice
scripting_phone_linesettgetdtmffstring
scripting_phone_linesettgetdtmffcount
scripting_phone_linesett_enable_speakerphone
scripting_phone_linesett_disable_speakerphone
scripting_phone_linesettncount
scripting_phone_linesettndial
scripting_phone_linesettanswerspeakerphone
scripting_phone_linesettanswerlocal
scripting_phone_linesettanswer
scripting_phone_linesettlastvoiceseconding
scripting_phone_linesettlastcallertinfo
scripting_phone_hipsetcallwaitingled
scripting_phone_hipsetlocalcid
scripting_phone_hipcmd
scripting_phone_handsetonhook
scripting_phone_getlastvoicecommand
scripting_phone_createmessagefilename
scripting_phone_contactclass
scripting_phone_clearlastvoicecommand
scripting_phone_cidnumber
scripting_phone_cidname
scripting_phone_adrsave
scripting_phone_adrsave
scripting_phone_adrget
scripting_phone_adrcount

home > scripting > phone > scripting_phone_adrcount

scripting_phone_adrcount

adrcount

purpose
this function returns the number of entries that are in the address book.

parameters
none.

returns
return value: address book entries
type: integer

see also
Scripting_Phone_LINEClearDTMF
Scripting_Phone_WaitMS
Scripting_Phone_StopListening
Scripting_Phone_StartListening
Scripting_Phone_Speak
Scripting_Phone_SetSpeaker
Scripting_Phone_RestoreSettings
Scripting_Phone_MBSort
Scripting_Phone_MBSave
Scripting_Phone_MBNextUnreadMessage
Scripting_Phone_MBNextReadMessage
Scripting_Phone_MBNew
Scripting_Phone_MBMessageTime
Scripting_Phone_MBMessageName
Scripting_Phone_MBMessageLength
Scripting_Phone_MBMessageFrom
Scripting_Phone_MBMessageDate
Scripting_Phone_MBMarkUnRead
Scripting_Phone_MBMarkRead
Scripting_Phone_MBGetLoggedIn
Scripting_Phone_MBGetDefault
Scripting_Phone_MBGetByName
Scripting_Phone_MailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINESetRingsCurrent
Scripting_Phone_LINESetSpeakingSpeed
Scripting_Phone_LINESetRings
Scripting_Phone_LINESetGreeting
Scripting_Phone_LINESetCIDNumber
Scripting_Phone_LINESetCIDName
Scripting_Phone_LINESetCIDInfo
Scripting_Phone_LINESetAnswerMode
Scripting_Phone_LINEMailboxClass
Scripting_Phone_LINEStopSpeaking
Scripting_Phone_LINEStatus
Scripting_Phone_LINESetVoice
Scripting_Phone_LINEResetCallTimeout
Scripting_Phone_LINEReset
Scripting_Phone_LINERecordStop
Scripting_Phone_LINERecordStart
Scripting_Phone_LINEMuteRings
Scripting_Phone_LINEIsSpeaking
Scripting_Phone_LINEHangup
Scripting_Phone_LINEGetVoice
Scripting_Phone_LINEGetDTMFString
Scripting_Phone_LINEGetDTMFCount
Scripting_Phone_LINEEnableSpeakerPhone
Scripting_Phone_LINEDisableSpeakerPhone
Scripting_Phone_LINECount
Scripting_Phone_LINEAnswerSpeakerPhone
Scripting_Phone_LINEAnswerLocal
Scripting_Phone_LINEAnswer
Scripting_Phone_LastVoiceMailInfo
Scripting_Phone_LastCallerInfo
Scripting_Phone_HIPSetCallWaitingLED
Scripting_Phone_HIPSendLocalCID
Scripting_Phone_HIPCmd
Scripting_Phone_HandsetOnHook
Scripting_Phone_HIPClearLastVoiceCommand
Scripting_Phone_HIPCreateMessageFilename
Scripting_Phone_HIPClass
Scripting_Phone_ClearLastVoiceCommand
Scripting_Phone_CIDNumber
Scripting_Phone_CIDName
Scripting_Phone_ADRSave
Scripting_Phone_ADRNew
Scripting_Phone_ADRGet
Scripting_Phone_ADRDelete
GetScriptPath

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

See also
IsScriptRunning
RunScript
RunScriptFunc
ScriptsRunning
WaitEvents
WaitSecs
IsScriptRunning

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

See also
GetScriptPath
RunScript
RunScriptFunc
ScriptsRunning
WaitEvents
WaitSecs

RunScript

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.

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: Boolean
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.

See also
GetScriptPath
IsScriptRunning
RunScriptFunc
ScriptsRunning
WaitEvents
WaitSecs

---

**RunScriptFunc**

**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.

See also
GetScriptPath
IsScriptRunning
ScriptsRunning

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.

See also

GetScriptPath
IsScriptRunning
RunScript
RunScriptFunc
WaitEvents
WaitSecs

WaitEvents

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.")
WaitSecs

**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**

```vba
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
```

See also
GetScriptPath
IsScriptRunning
RunScript
RunScriptFunc
ScriptsRunning
WaitEvents
Modifying Voice Recognition Commands

In This Section

AddVoiceCommand
ClearAllVoiceCommands

See also
Getting Last Voice Command Information
Controlling Speaker Clients

AddVoiceCommand

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
Example

The following script will read your E-mail messages.

```vbscript
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, "")
  Next

  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
```
```vbnet
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
```

See also
ClearAllVoiceCommands

---

**ClearAllVoiceCommands**

**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.

See also
AddVoiceCommand

---

**Getting Last Voice Command Information**

In This Section
GetLastVRCollection
GetLastVRInfo
LastCommandSelected
LastVoiceCommand
LastVoiceCommandHost
LastVoiceCommandInstance
LastVoiceCommandPhone
LastVoiceCommandRaw

See also
Modifying Voice Recognition Commands
Controlling Speaker Clients

GetLastVRCollection

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

See also
GetLastVRInfo
LastCommandSelected
LastVoiceCommand
LastVoiceCommandHost
LastVoiceCommandInstance
LastVoiceCommandPhone
LastVoiceCommandRaw

clsLastVR

HomeSeer HS3 - End User Documentation
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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw</td>
<td>String</td>
<td>This is the raw voice command as it was heard and recognized by the VR engine.</td>
</tr>
<tr>
<td>Parsed</td>
<td>String</td>
<td>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.</td>
</tr>
<tr>
<td>Host</td>
<td>String</td>
<td>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.</td>
</tr>
<tr>
<td>Instance</td>
<td>String</td>
<td>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.</td>
</tr>
<tr>
<td>VRTime</td>
<td>Date</td>
<td>This is the date/time the phrase was recognized.</td>
</tr>
<tr>
<td>ID</td>
<td>Integer</td>
<td>This is the voice recognition context ID number that was matched for the recognized phrase.</td>
</tr>
</tbody>
</table>

**See Also:**
- GetLastVRInfo
- GetLastVRCollection
- LastCommandSelected
- LastVoiceCommand
- LastVoiceCommandPhone
- LastVoiceCommandHost
- LastVoiceCommandInstance
- LastVoiceCommandRaw

**See also**

Home > Scripting > Speech Recognition > Getting Last Voice Command Information > GetLastVRInfo

## GetLastVRInfo

### 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...
clsLastVR

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:

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw</td>
<td>String</td>
<td>This is the raw voice command as it was heard and recognized by the VR engine.</td>
</tr>
<tr>
<td>Parsed</td>
<td>String</td>
<td>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.</td>
</tr>
<tr>
<td>Host</td>
<td>String</td>
<td>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.</td>
</tr>
<tr>
<td>Instance</td>
<td>String</td>
<td>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.</td>
</tr>
<tr>
<td>VRTime</td>
<td>Date</td>
<td>This is the date/time the phrase was recognized.</td>
</tr>
<tr>
<td>ID</td>
<td>Integer</td>
<td>This is the voice recognition context ID number that was matched for the recognized phrase.</td>
</tr>
</tbody>
</table>
LastCommandSelected

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

LastVoiceCommand

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

See also
- GetLastVRCollection
- GetLastVRInfo
- LastVoiceCommand
- LastVoiceCommandPhone
- LastVoiceCommandHost
- LastVoiceCommandInstance
- LastVoiceCommandRaw
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:

```plaintext
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:

```plaintext
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.

```vbs
sub main()
    dim v
    v=hs.LastVoiceCommand
    msgbox "I heard ",v
end sub
```

**See Also:**

- clsLastVR
- GetLastVRInfo
- GetLastVRCollection
- LastCommandSelected
- LastVoiceCommandPhone
- LastVoiceCommandHost
- LastVoiceCommandInstance
- LastVoiceCommandRaw

See also
- GetLastVRCollection
- GetLastVRInfo
- LastCommandSelected
- LastVoiceCommandHost
- LastVoiceCommandInstance
- LastVoiceCommandPhone
- LastVoiceCommandRaw

---

**LastVoiceCommandHost**

**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

See also
- GetLastVRCollection
- GetLastVRInfo
- LastCommandSelected
- LastVoiceCommand
- LastVoiceCommandInstance
- LastVoiceCommandPhone
- LastVoiceCommandRaw

LastVoiceCommandInstance

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. This 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
- LastVoiceCommandInstance
- LastVoiceCommandRaw

See also
- GetLastVRCollection
- GetLastVRInfo
- LastCommandSelected
- LastVoiceCommand
- LastVoiceCommandHost
- LastVoiceCommandPhone
- LastVoiceCommandRaw
LastVoiceCommandPhone

**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
- LastCommand
- LastVoiceCommand
- LastVoiceCommandPhone
- LastVoiceCommandHost
- LastVoiceCommandInstance
- LastVoiceCommandRaw

See also

- GetLastVRCollection
- GetLastVRInfo
- LastCommandSelected
- LastCommand
- LastVoiceCommand
- LastVoiceCommandHost
- LastVoiceCommandInstance
- LastVoiceCommandRaw
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

See also

- GetLastVRCollection
- GetLastVRInfo
- LastCommandSelected
- LastVoiceCommand
- LastVoiceCommandHost
- LastVoiceCommandInstance
- LastVoiceCommandPhone

Controlling Speaker Clients

**In This Section**

- GetListenStatus
- ListenMode
- ListenForCommands
- SetSpeaker
- StartListen
- StopListen

See also

- Modifying Voice Recognition Commands
- Getting Last Voice Command Information
GetListenStatus

**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.

See also

ListenMode
ListenForCommands
SetSpeaker
StartListen
StopListen

ListenMode

**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

See also

GetListenStatus
ListenForCommands
SetSpeaker
StartListen
StopListen

ListenForCommands
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

See also
GetListenStatus
ListenMode
StartListen
StopListen

SetSpeaker
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.

See also
GetListenStatus
ListenMode
ListenForCommands
StartListen
StopListen
StartListen

**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.

See also

- GetListenStatus
- ListenMode
- ListenForCommands
- SetSpeaker
- StopListen

StopListen

**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.

See also

- GetListenStatus
- ListenMode
- ListenForCommands
- SetSpeaker
- StartListen

Strings, Global Variables, and Encryption
Global Variables

In This Section

CreateVar
DeleteVar
GetVar
SaveVar

See also
Encryption
Counters
Timers

Home > Scripting > Strings, Global Variables, and Encryption > Global Variables

Global Variables

CreateVar

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

DeleteVar

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.

GetVar

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")

See also
CreateVar
DeleteVar
SaveVar

SaveVar

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)

See also
CreateVar
DeleteVar
GetVar

Encryption

In This Section

EncryptString
EncryptStringEx
DecryptString
EncryptString

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(P 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 encryers 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

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

DecryptString

Purpose

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

See also
EncryptString
EncryptStringEx
Counters

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.

See also

- Global Variables
- Encryption
- Timers

CounterValue

**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")```  

See also

- CounterReset
- CounterIncrement
- CounterDecrement

CounterReset

**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**

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

See also  
- CounterValue  
- CounterIncrement  
- CounterDecrement

---

**CounterIncrement**

**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**

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

See also  
- CounterValue  
- CounterReset  
- CounterDecrement

---

**CounterDecrement**

**Purpose**

Decrements 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 decrement.

**Returns**

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

**Example**

```csharp
hs.CounterDecrement("mycounter")
```

See also  
- CounterValue  
- CounterReset  
- CounterIncrement

---

**TimerValue**

**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**

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

See also
TimerReset

**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**

```plaintext
hs.TimerReset("mytimer")
```

See also

- TimerValue

---

**Time and Calendar**

**In This Section**

- Time Related
- Calendar Related

See also

- About Scripts
- Applications and Plugins
- Computer
- Devices
- Email
- Events
- Internet
- Phone
- Scripts
- Speech Recognition
- Strings, Global Variables, and Encryption
- Text-To-Speech and Media
Time Related

In This Section

- LocalTimeZone
- SolarNoon
- Sunrise
- SunriseDt
- Sunset
- SunsetDt
- TimeZoneName

See also

Calendar Related

LocalTimeZone

**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**

```plaintext
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.
```

See also

- SolarNoon
- Sunrise
- SunriseDt
- Sunset
- SunsetDt
- TimeZoneName

SolarNoon
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

```vbs
sub main()
    dim t

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

end sub
```

See also

LocalTimeZone
Sunrise
SunriseDt
Sunset
SunsetDt
TimeZoneName

---

Sunrise

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

```vbs
sub main()
    dim t

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

end sub
```

See also

LocalTimeZone
Sunrise
SunriseDt
Sunset
SunsetDt
TimeZoneName
SunriseDt

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

See also
LocalTimeZone
SolarNoon
SunriseD
Sunset
SunsetDt
TimeZoneName

Sunset

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**

```vba
sub main()
    dim t
    t=hs.Sunset
    msgbox "Sunset is at " & t
end sub
```

See also
LocalTimeZone  
SolarNoon  
Sunrise  
SunriseDt  
SunsetDt  
TimeZoneName

---

**SunsetDt**

**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**

```vba
sub main()
    dim t
    t=hs.SunsetDt
    msgbox "Sunset is at " & FormatDateTime(t, vbLongTime)
end sub
```

See also
LocalTimeZone  
SolarNoon  
SolarNoon  
Sunrise  
SunriseDt  
TimeZoneName

---

Home > Scripting > Time and Calendar > Time Related > SunsetDt

---

HomeSeer HS3 - End User Documentation
TimeZoneName

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

See also
LocalTimeZone
SolarNoon
Sunrise
SunriseDt
Sunset
SunsetDt

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

See also
Time Related
DaylightSavings

**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
```

See also
DaysInMonth
DaysLeftInMonth
DaysLeftInYear
EvenOddMonth
EvenOddDay
GetLastWeekday
GetSpecialDay
IsSpecialDay
IsWeekday
IsWeekend
Moon
Weekdays
WeekEndDays
WeekNumber
WeekNumberEx
WeeksLeftInYear
WeeksLeftInYearEx

DaysInMonth

**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**

```vbnet
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

---

**DaysLeftInMonth**

**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**

```vbnet
hS.writeLog("Info","There are " & hS.DaysLeftInMonth.ToString & " days left in the month.")
```

**See Also**

- DaysInMonth
- DaysLeftInYear
- WeekNumber
- WeeksLeftInYear
DaysLeftInYear

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

See Also

DaysInMonth
DaysLeftInMonth
WeekNumber
WeeksLeftInYear

See also

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

**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**

```vbnet
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

---

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**

```vbs
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

---

See Also:
- DaylightSavings
- DaysInMonth
- DaysLeftInMonth
- DaysLeftInYear
- EvenOddMonth
- GetLastWeekday
- GetSpecialDay
- IsSpecialDay
- IsWeekday
- IsWeekend
- Moon
- Weekdays
- WeekEndDays
- WeekNumber
- WeekNumberEx
- WeeksLeftInYear
- WeeksLeftInYearEx

---

GetLastWeekday

**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**

```vbs
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
See also
- DaylightSavings
- DaysInMonth
- DaysLeftInMonth
- DaysLeftInYear
- EvenOddMonth
- EvenOddDay
- GetSpecialDay
- IsSpecialDay
- IsWeekday
- IsWeekend
- Moon
- Weekdays
- WeekEndDays
- WeekNumber
- WeekNumberEx
- WeeksLeftInYear
- WeeksLeftInYearEx

GetSpecialDay

**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

See also
DaylightSavings
DaysInMonth
DaysLeftInMonth
DaysLeftInYear
EvenOddMonth
EvenOddDay
GetLastWeekday
IsSpecialDay
IsWeekday
IsWeekend
Moon
Weekdays
WeekEndDays
WeekNumber
WeekNumberEx
WeeksLeftInYear
WeeksLeftInYearEx

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:
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
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

See also
DaylightSavings
DaysInMonth
DaysLeftInMonth
DaysLeftInYear
EvenOddMonth
EvenOddDay
GetLastWeekday
GetSpecialDay
IsSpecialDay
IsWeekend
Moon
Weekdays
WeekEndDays
WeekNumber
WeekNumberEx
WeeksLeftInYear
WeeksLeftInYearEx

IsWeekend

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
IsWeekend
Weekdays
WeekendDays

See also
DaylightSavings
DaysInMonth
DaysLeftInMonth
DaysLeftInYear
EvenOddMonth
Moon

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

See also
DaylightSavings
DaysInMonth
DaysLeftInMonth
Weekdays

**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

See also

DaylightSavings
DaysInMonth
DaysLeftInMonth
DaysLeftInYear
EvenOddMonth
EvenOddDay
GetLastWeekday
GetSpecialDay
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**

```vbnet
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
- WeekEndDays

See also

- DaylightSavings
- DaysInMonth
- DaysLeftInMonth
- DaysLeftInYear
- EvenOddMonth
- EvenOddDay
- GetLastWeekday
- GetSpecialDay
- IsSpecialDay
- IsWeekday
- IsWeekend
- Moon
**WeekNumber**

**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

See also
- DaylightSavings
- DaysInMonth
- DaysLeftInMonth
- DaysLeftInYear
- EvenOddMonth
- EvenOddDay
- GetLastWeekday
- GetSpecialDay
- IsSpecialDay
- IsWeekday
- IsWeekend
- Moon
- Weekdays
- WeekEndDays
- WeekNumberEx
- 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:

<table>
<thead>
<tr>
<th>Week Mode Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>4</td>
<td>The first week of the year is determined by the first week with at least four days in the new year.</td>
</tr>
<tr>
<td>(Anything Else)</td>
<td>The first week of the year is determined by the first full week starting on Sunday in the new year.</td>
</tr>
</tbody>
</table>

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

See also

- DaylightSavings
- DaysInMonth
- DaysLeftInMonth
- DaysLeftInYear
- EvenOddMonth
- EvenOddDay
- GetLastWeekday
- GetSpecialDay
- IsSpecialDay
- IsWeekday
- IsWeekend
- Moon
- Weekdays
- WeekEndDays
- WeekNumber
- WeeksLeftInYear
- WeeksLeftInYearEx

**WeeksLeftInYear**

**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

See also

- DaylightSavings
- DaysInMonth
- DaysLeftInMonth
- DaysLeftInYear
- EvenOddMonth
- EvenOddDay
- GetLastWeekday
- GetSpecialDay
- IsSpecialDay
- IsWeekday
- IsWeekend
- Moon
- Weekdays
- WeekEndDays
- WeekNumber
- WeekNumberEx
- WeeksLeftInYearEx

**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:

<table>
<thead>
<tr>
<th>Week Mode Value</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>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.</td>
</tr>
<tr>
<td>4</td>
<td>The first week of the year is determined by the first week with at least four days in the new year.</td>
</tr>
<tr>
<td>(Anything Else)</td>
<td>The first week of the year is determined by the first full week starting on Sunday in the new year.</td>
</tr>
</tbody>
</table>
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

See also

- DaylightSavings
- DaysInMonth
- DaysLeftInMonth
- DaysLeftInYear
- EvenOddMonth
- EvenOddDay
- GetLastWeekday
- GetSpecialDay
- IsSpecialDay
- IsWeekday
- IsWeekend
- Moon
- Weekdays
- WeekEndDays
- WeekNumber
- WeekNumberEx
- WeeksLeftInYear

---

**Text-To-Speech and Media**

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/controlling 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
- SpeakerClientGlobalAudio
- MediaOnlyProcedures
- Text-to-SpeechOnlyProcedures

See also

- About Scripts
- Applications and Plugins
GetInstanceList

**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
```

See also
IsSpeakerBusy
SpeakToFile
Speaker Client Global Audio
Media Only Procedures
Text-to-Speech Only Procedures

IsSpeakerBusy

**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.
See also
GetInstanceList
SpeakToFile
Speaker Client Global Audio
Media Only Procedures
Text-to-Speech Only Procedures

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

See also
GetInstanceList
IsSpeakerBusy
Speaker Client Global Audio
Media Only Procedures
Text-to-Speech Only Procedures

Home > Scripting > Text-To-Speech and Media > SpeakToFile

Speaker Client Global Audio

In This Section
SetVolume
GetVolume
GetMuteStatus
GetPauseStatus
MuteAudio
PauseAudio
UnMuteAudio
UnPauseAudio

See also
GetInstanceList
IsSpeakerBusy
SpeakToFile
Media Only Procedures
Text-to-Speech Only Procedures

SetVolume

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.SetVolume 20, "Kitchen"
    hs.speak "I am speaking softer on the Kitchen computer than on the others.",TRUE
end sub

See also
GetVolume
GetMuteStatus
GetPauseStatus
MuteAudio
PauseAudio
UnMuteAudio
UnPauseAudio
GetVolume

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.

See also
SetVolume
GetMuteStatus
GetPauseStatus
MuteAudio
PauseAudio
UnMuteAudio
UnPauseAudio

GetMuteStatus

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.

See also
SetVolume
GetVolume
GetPauseStatus
MuteAudio
PauseAudio
UnMuteAudio
UnPauseAudio
GetPauseStatus

**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:

<table>
<thead>
<tr>
<th>Bit Values</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 1 = 0</td>
<td>No Wavefile Instance</td>
</tr>
<tr>
<td>Bit 1 = 1</td>
<td>Wavefile Present</td>
</tr>
<tr>
<td>Bit 2 = 0</td>
<td>Wavefile Paused</td>
</tr>
<tr>
<td>Bit 2 = 1</td>
<td>Wavefile Playing</td>
</tr>
<tr>
<td>Bit 3 = 0</td>
<td>TTS Present</td>
</tr>
<tr>
<td>Bit 3 = 1</td>
<td>No TTS Present</td>
</tr>
<tr>
<td>Bit 4 = 0</td>
<td>TTS is currently speaking</td>
</tr>
<tr>
<td>Bit 4 = 1</td>
<td>TTS is not speaking</td>
</tr>
</tbody>
</table>

See also
SetVolume
GetVolume
GetMuteStatus
MuteAudio
PauseAudio
UnMuteAudio
UnPauseAudio

MuteAudio

**Purpose**

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

**Parameters**
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.

See also
SetVolume
GetVolume
GetMuteStatus
GetPauseStatus
PauseAudio
UnMuteAudio
UnPauseAudio

UnMuteAudio

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.

See also
SetVolume
GetVolume
GetMuteStatus
GetPauseStatus
MuteAudio
UnMuteAudio
UnPauseAudio
UnPauseAudio

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.

See also

SetVolume
GetVolume
GetMuteStatus
GetPauseStatus
MuteAudio
PauseAudio
UnMuteAudio

Media Only Procedures

In This Section

MediaFilename
MediaPlay
MediaPause
MediaMute
MediaMuteStatus
MediaPlaying
MediaStop
MediaUnPause
MediaVolume
See also
GetInstanceList
IsSpeakerBusy
SpeakToFile
Speaker Client Global Audio
Text-to-Speech Only Procedures

MediaFilename

**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` (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.

See also
MediaPlay
MediaPause
MediaMute
MediaIsPlaying
MediaStop
MediaUnPause
MediaVolume

MediaPlay

**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.

See also
MediaFilename
MediaPause
MediaMute
MediaIsPlaying
MediaStop
MediaUnPause
MediaVolume

MediaPause

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.

See also
MediaFilename
MediaPlay
MediaMute
MediaIsPlaying
MediaStop
MediaUnPause
MediaVolume

MediaMute

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**

```plaintext
sub main()
  ' mute the Windows Media Player
  hs.MediaMute TRUE
end sub
```

See also
MediaFilename
MediaPlay
MediaPause
MediaIsPlaying
MediaStop
MediaUnPause
MediaVolume

---

**MediaIsPlaying**

**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.

See also
MediaFilename
MediaPlay
MediaPause
MediaIsPlaying
MediaStop
MediaUnPause
MediaVolume
MediaStop

**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.

See also
- MediaFilename
- MediaPlay
- MediaPause
- MediaMute
- MediaIsPlaying
- MediaUnPause
- MediaVolume

MediaUnPause

**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.

See also
- MediaFilename
- MediaPlay
- MediaPause
- MediaMute
- MediaIsPlaying
- MediaStop
- MediaVolume

MediaVolume
**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
```

See also

- MediaFilename
- MediaPlay
- MediaPause
- MediaMute
- MediaIsPlaying
- MediaStop
- MediaUnPause

**Text-to-Speech Only Procedures**

**In This Section**

- Speak
- SpeakEx
- SpeakProxy
- GetVoiceName
- MuteSpeech
- SetSpeakingSpeed
- SetVoice
- StopSpeaking
- PlayWavFile
- PlayWavFileVol

See also

- GetInstanceList
- IsSpeakerBusy
- SpeakToFile
- Speaker Client Global Audio

Media Only Procedures
Speak

**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**

```vbscript
Sub Main()
    ' speak and wait
    hs.speak "hello there", True
    hs.speak "Hello people in the kitchen.", True, "Kitchen:"'
End Sub
```

**See Also**


See also

- SpeakEx
- SpeakProxy
- GetVoiceName
- MuteSpeech
- SetSpeakingSpeed
- SetVoice
- StopSpeaking
- PlayWavFile
- PlayWavFileVol
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


See also

Speak
SpeakProxy
GetVoiceName
MuteSpeech
SetSpeakingSpeed
SetVoice
StopSpeaking
PlayWavFile
PlayWavFileVol

SpeakProxy

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.

See also
Speak
SpeakEx
GetVoiceName
MuteSpeech
SetSpeakingSpeed
SetVoice
StopSpeaking
PlayWavFile
PlayWavFileVol

GetVoiceName

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

See also
Speak
SpeakEx
SpeakProxy
MuteSpeech
SetSpeakingSpeed
SetVoice
StopSpeaking
PlayWavFile
PlayWavFileVol

MuteSpeech

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
```

See also  
Speak  
SpeakEx  
SpeakProxy  
GetVoiceName  
SetSpeakingSpeed  
SetVoice  
StopSpeaking  
PlayWavFile  
PlayWavFileVol

---

**SetSpeakingSpeed**

**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**
Sub Main()
    Dim iOldSpeed
    Dim iNothing
    hs.Speak "This is the rate at which I am currently speaking."
    iOldSpeed = hs.SetSpeakingSpeed(8)
    hs.Speak "Now I am talking very fast like I just drank two pots of coffee."
    iNothing = hs.SetSpeakingSpeed(iOldSpeed)
End Sub

See also
Speak
SpeakEx
SpeakProxy
GetVoiceName
MuteSpeech
SetVoice
StopSpeaking
PlayWavFile
PlayWavFileVol

SetVoice

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.

See also
Speak
SpeakEx
SpeakProxy
GetVoiceName
MuteSpeech
SetVoice
StopSpeaking
PlayWavFile
PlayWavFileVol
StopSpeaking

**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
hs.StopSpeaking
```

See also
- Speak
- SpeakEx
- SpeakProxy
- GetVoiceName
- MuteSpeech
- SetSpeakingSpeed
- SetVoice
- PlayWavFile
- PlayWavFileVol

---

PlayWavFile

**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.

See also
- Speak
- SpeakEx
- SpeakProxy
### PlayWavFileVol

#### 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. In 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.

#### See also
- Speak
- SpeakEx
- SpeakProxy
- GetVoiceName
- MuteSpeech
- SetSpeakingSpeed
- SetVoice
- StopSpeaking
- PlayWavFile